THE UNIVERSITY *of* EDINBURGH

Advanced Database Systems

Spring 2026

Q&A Session 2

If you require this document in an alternative format, such as large print or a coloured background, please contact milos.nikolic@ed.ac.uk

About This Session

Practice Worksheet 2 is now available on Learn

- We will work through some questions during this session

This slide deck includes animations of discussed algorithms

- External sorting
- Join algorithms

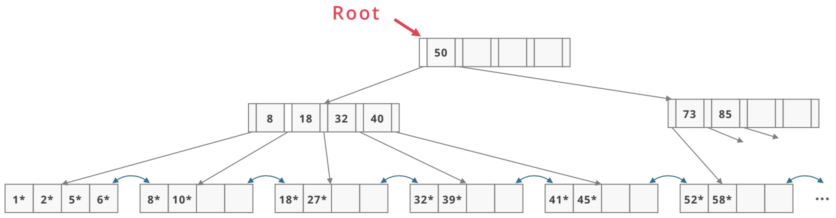
Main Purpose: To reinforce and strengthen your understanding

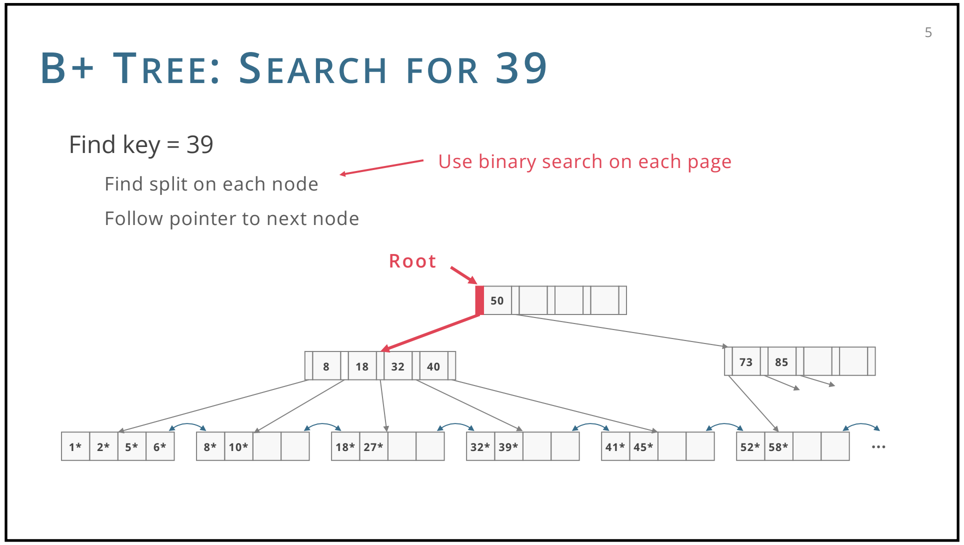
Question 1

B+ Tree: Search for 39

Find key = 39

- Find split on each node
- Follow pointer to next node





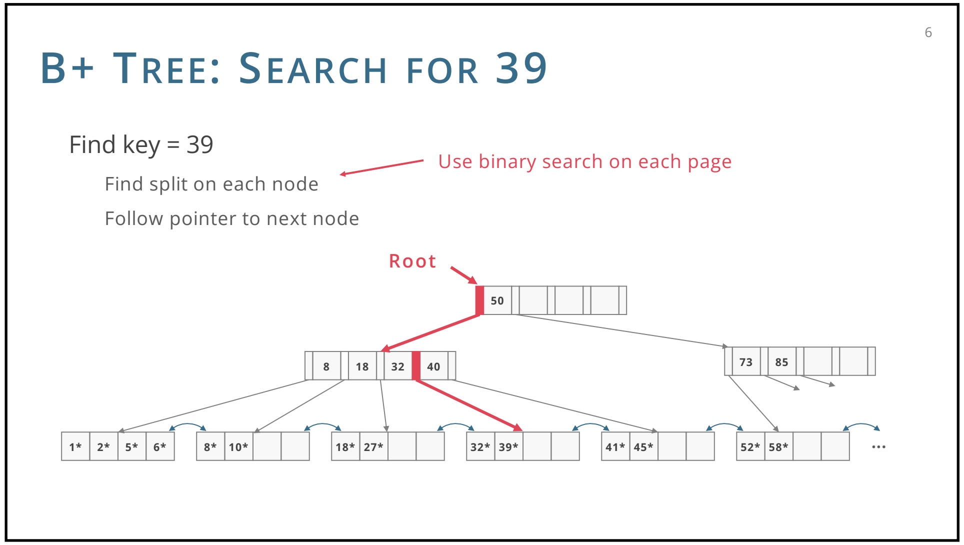
B+ Tree: Search for 39

Find key = 39

- Find split on each node
- Follow pointer to next node

Use binary search on each page

5



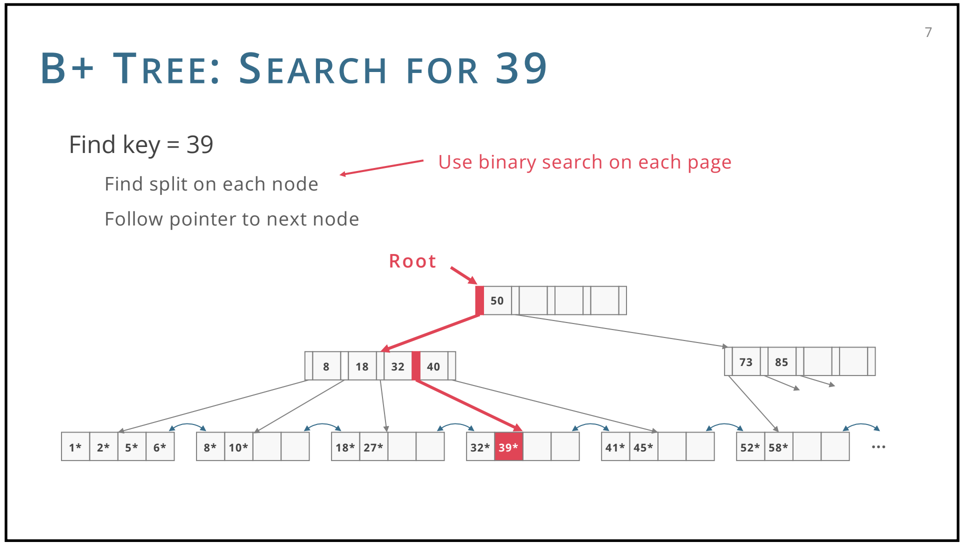
B+ Tree: Search for 39

Find key = 39

- Find split on each node
- Follow pointer to next node

Use binary search on each page

6



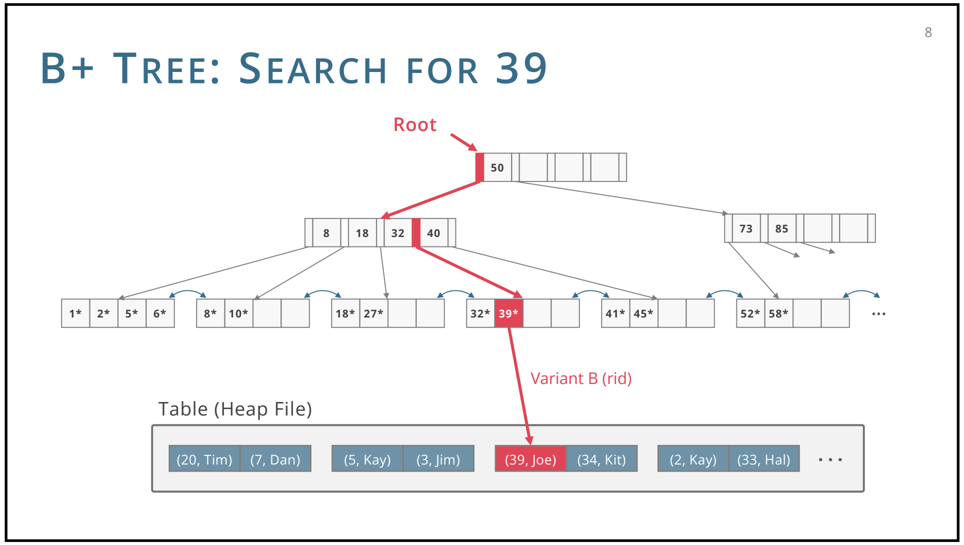
B+ Tree: Search for 39

Find key = 39

- Find split on each node
- Follow pointer to next node

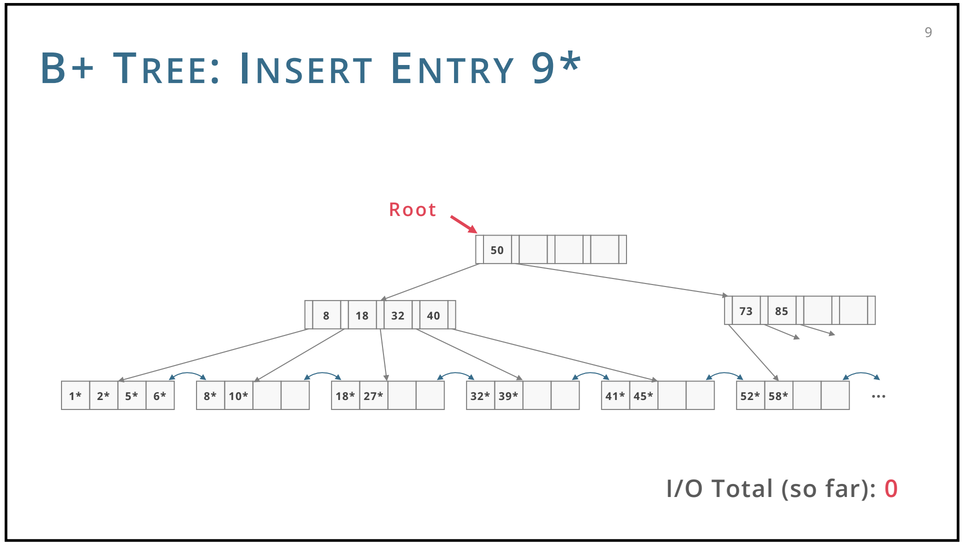
Use binary search on each page

7



B+ Tree: Search for 39

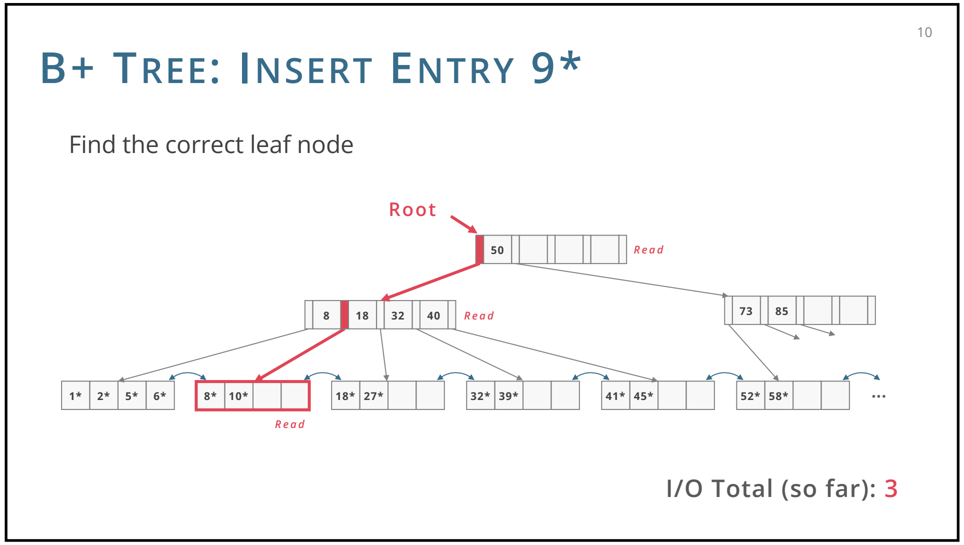
8



B+ Tree: Insert Entry 9*

I/O Total (so far): 0

9

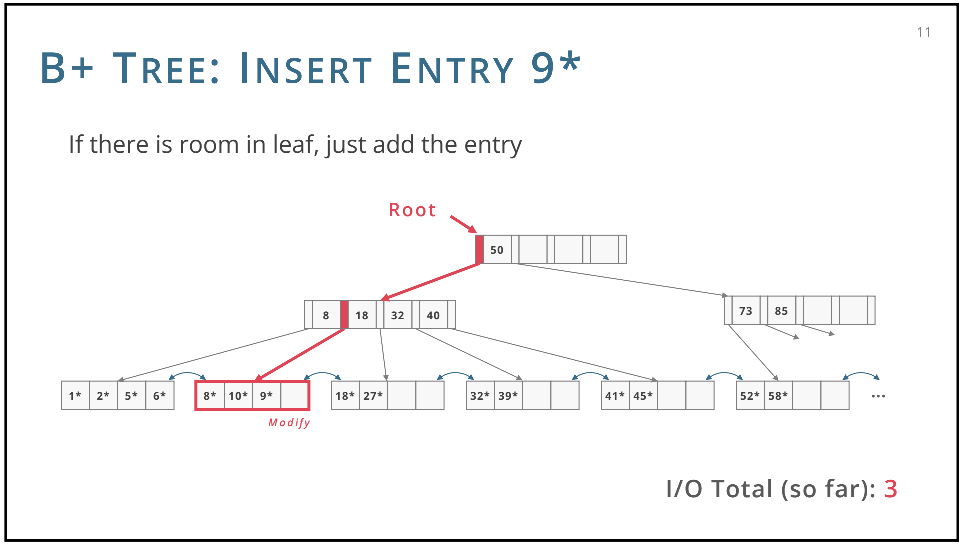


B+ Tree: Insert Entry 9*

Find the correct leaf node

I/O Total (so far): 3

10

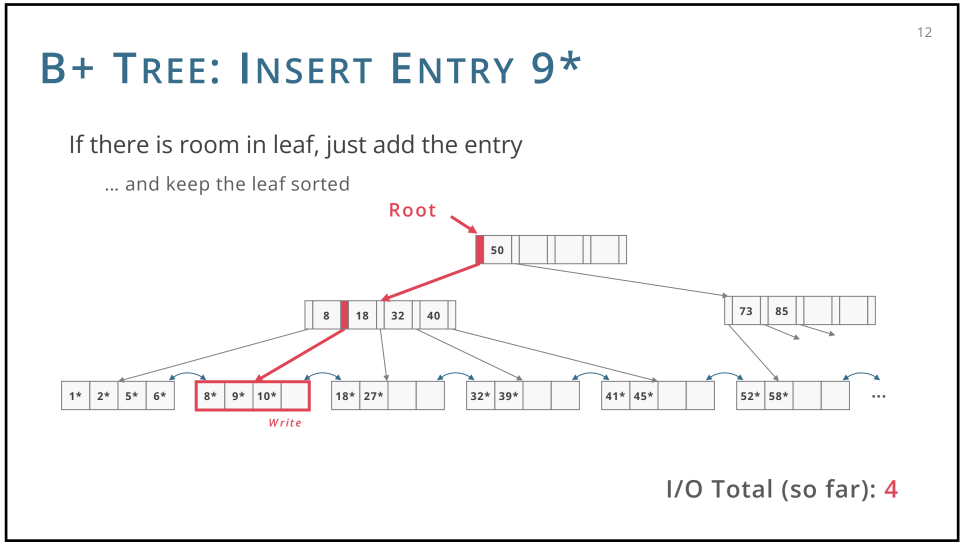


B+ Tree: Insert Entry 9*

If there is room in leaf, just add the entry

I/O Total (so far): 3

11



B+ Tree: Insert Entry 9*

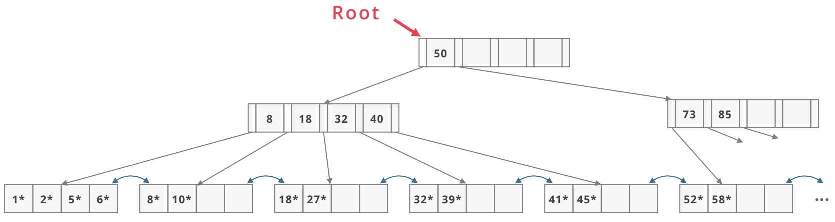
If there is room in leaf, just add the entry

… and keep the leaf sorted

I/O Total (so far): 4

12

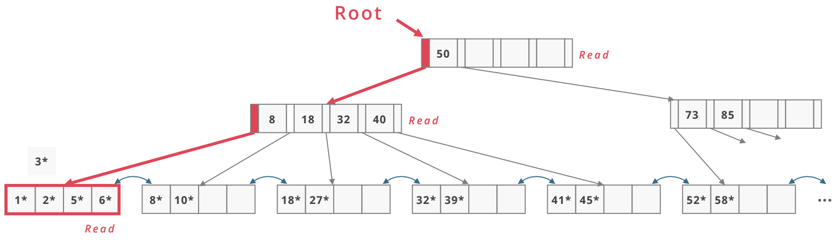
B+ TREE: INSERT ENTRY 3*



I/O Total (so far): 0

B+ TREE: INSERT ENTRY 3*

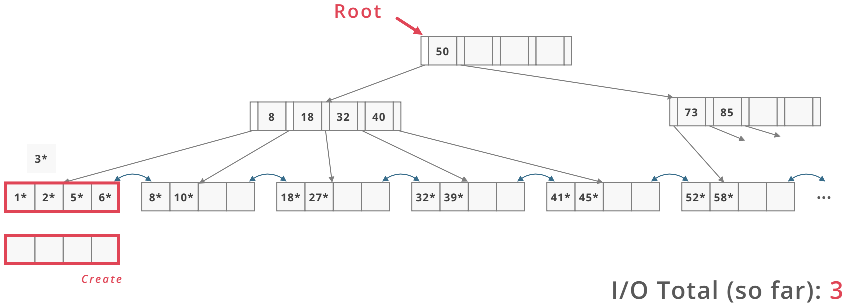
Find the correct leaf node



I/O Total (so far): 3

B+ TREE: INSERT ENTRY 3*

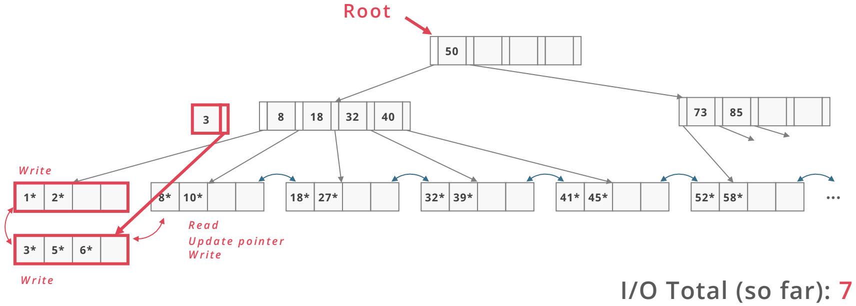
Split leaf if not enough room: into two leaves with d and d + 1 entries



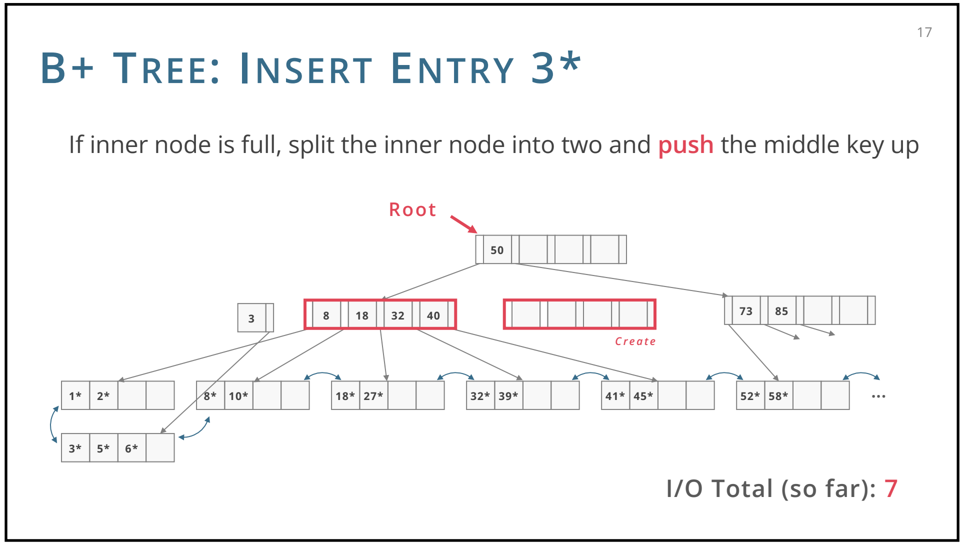
I/O Total (so far): 3

B+ TREE: INSERT ENTRY 3*

Copy up the middle key to inner node (since leaf nodes have data)



I/O Total (so far): 7

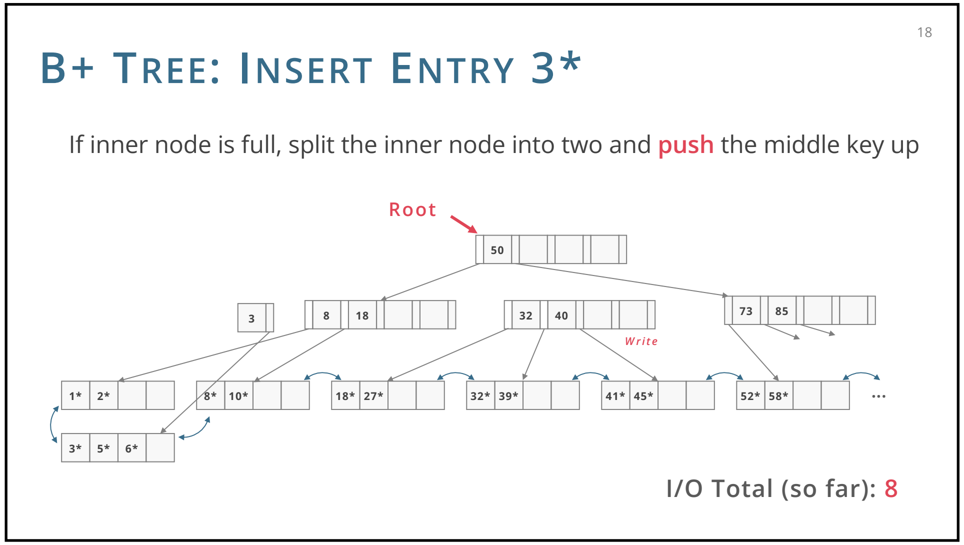


B+ Tree: Insert Entry 3*

If inner node is full, split the inner node into two and push the middle key up

I/O Total (so far): 7

17

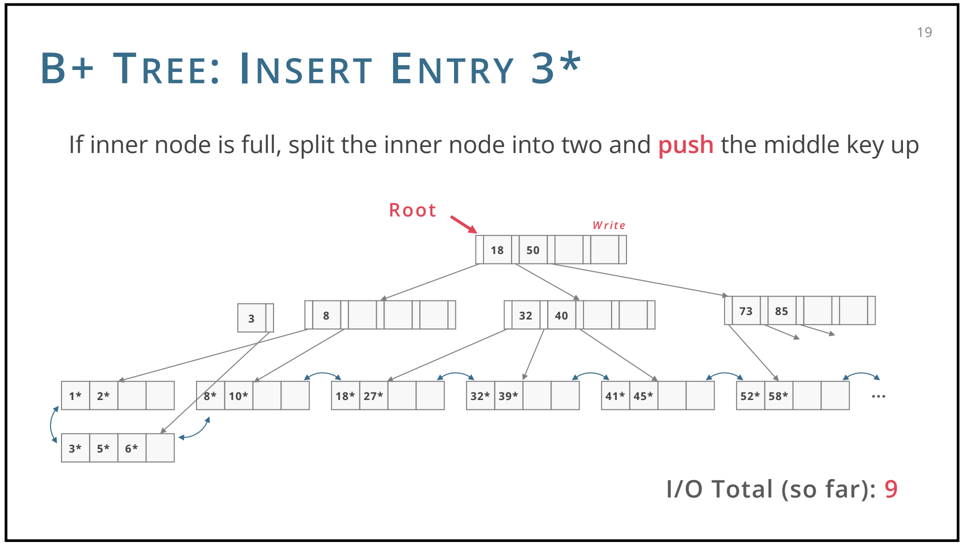


B+ Tree: Insert Entry 3*

If inner node is full, split the inner node into two and push the middle key up

I/O Total (so far): 8

18

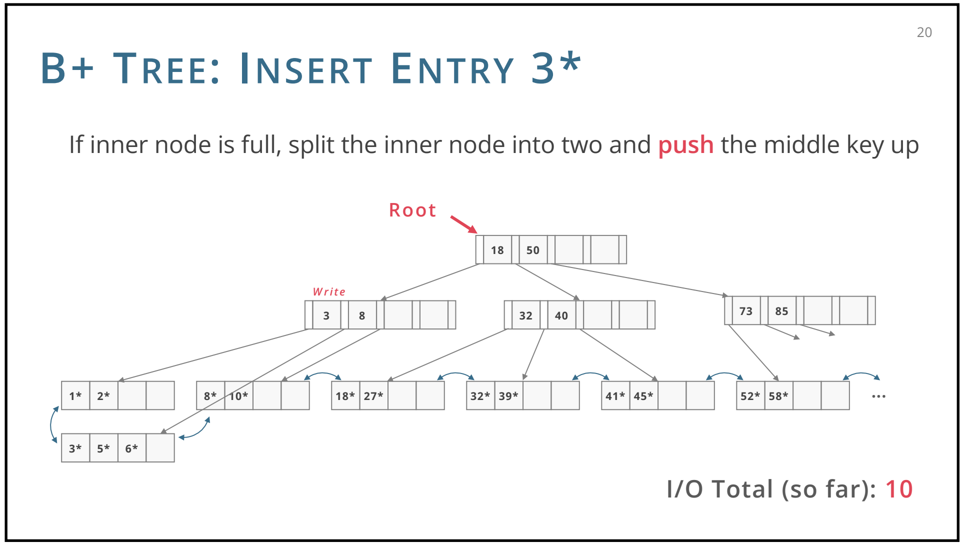


B+ Tree: Insert Entry 3*

If inner node is full, split the inner node into two and push the middle key up

I/O Total (so far): 9

19



B+ Tree: Insert Entry 3*

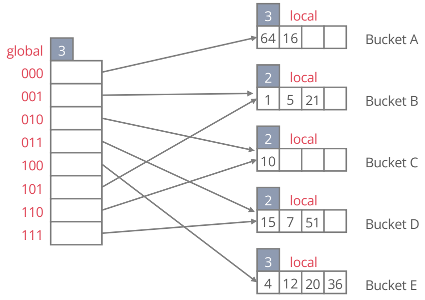
If inner node is full, split the inner node into two and push the middle key up

I/O Total (so far): 10

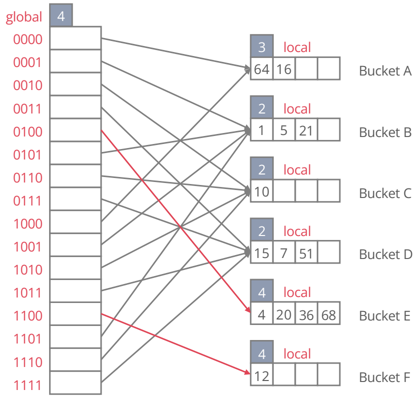
20

Question 2

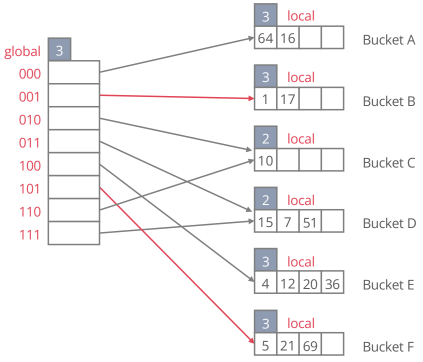
Extendible Hashing



Insert Entry with Hash Value 68



Insert Entries with Hash Values 17 & 69

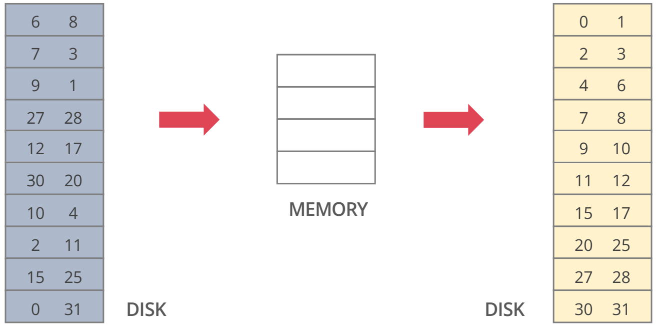


External Sorting

25

General External Merge Sort

Goal: sort records on 10 data pages using 4 buffer pages

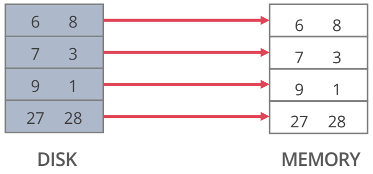


26

General External Merge Sort

Pass 0, Run 1

Read 4 pages into memory



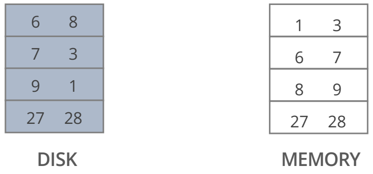
I/O Total (so far): 4

27

General External Merge Sort

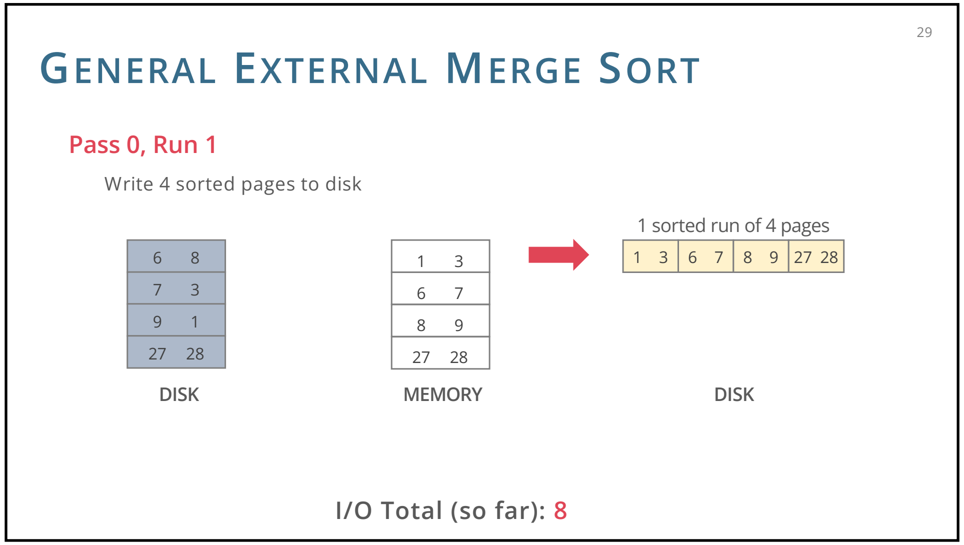
Pass 0, Run 1

In-memory sort



I/O Total (so far): 4

28

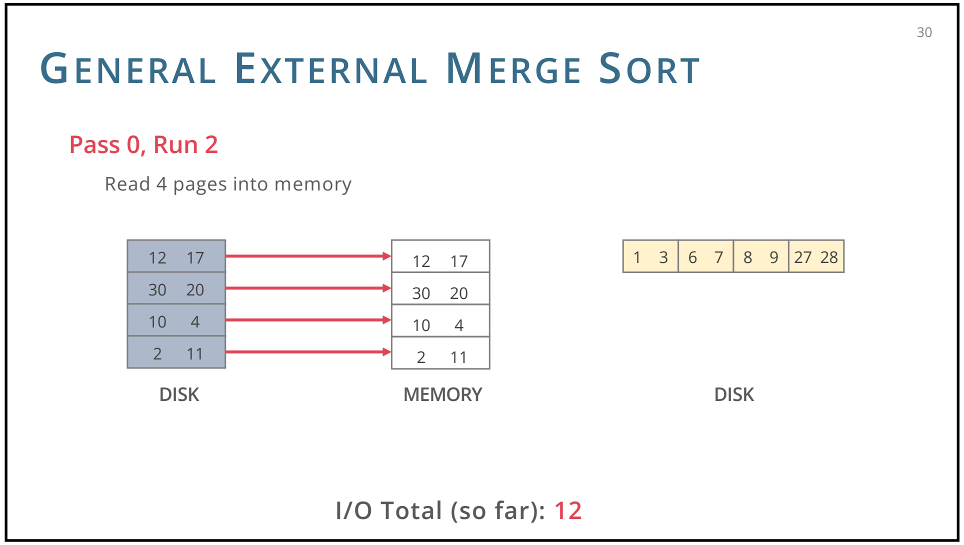


General External Merge Sort

Pass 0, Run 1

Write 4 sorted pages to disk

I/O Total (so far): 8

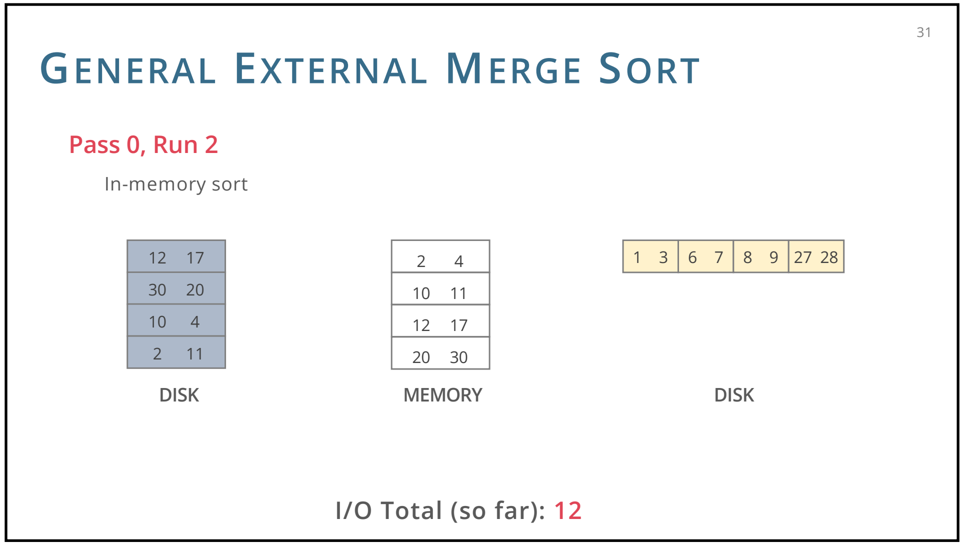


General External Merge Sort

Pass 0, Run 2

Read 4 pages into memory

I/O Total (so far): 12

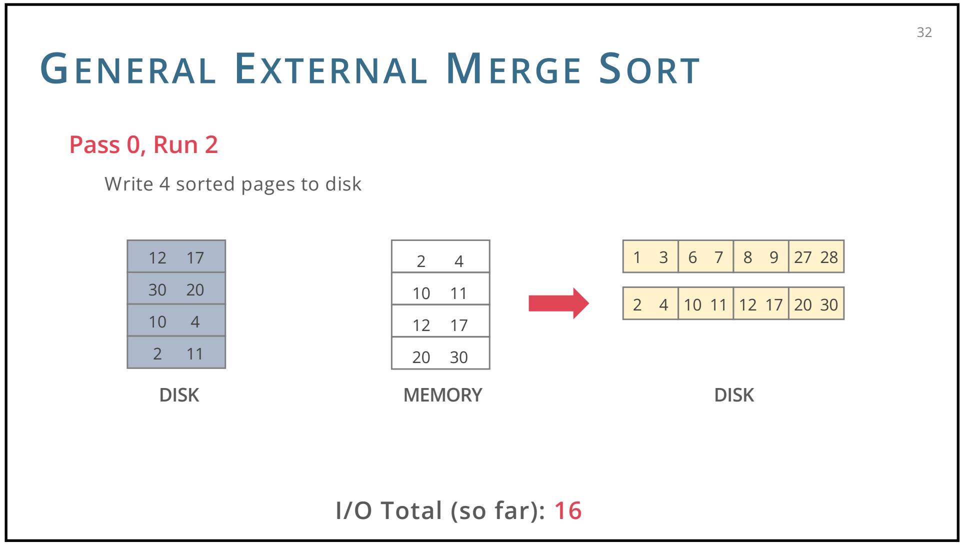


General External Merge Sort

Pass 0, Run 2

In-memory sort

I/O Total (so far): 12

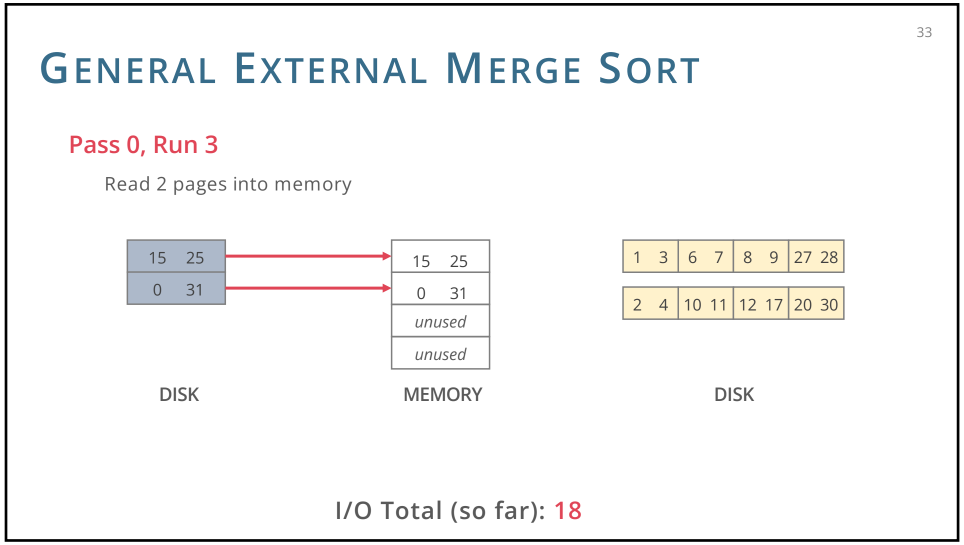


General External Merge Sort

Pass 0, Run 2

Write 4 sorted pages to disk

I/O Total (so far): 16



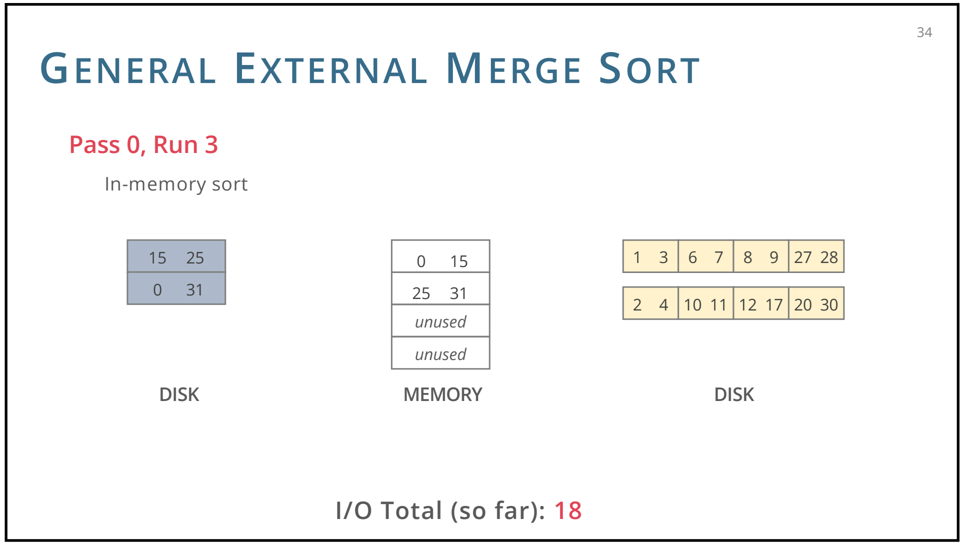
General External Merge Sort

Pass 0, Run 3

Read 2 pages into memory

I/O Total (so far): 18

33



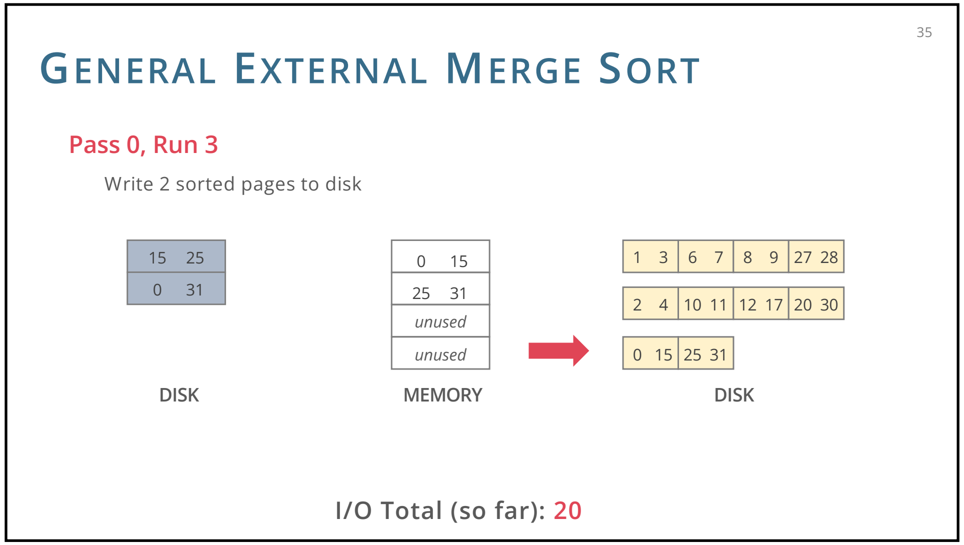
General External Merge Sort

Pass 0, Run 3

In-memory sort

I/O Total (so far): 18

34



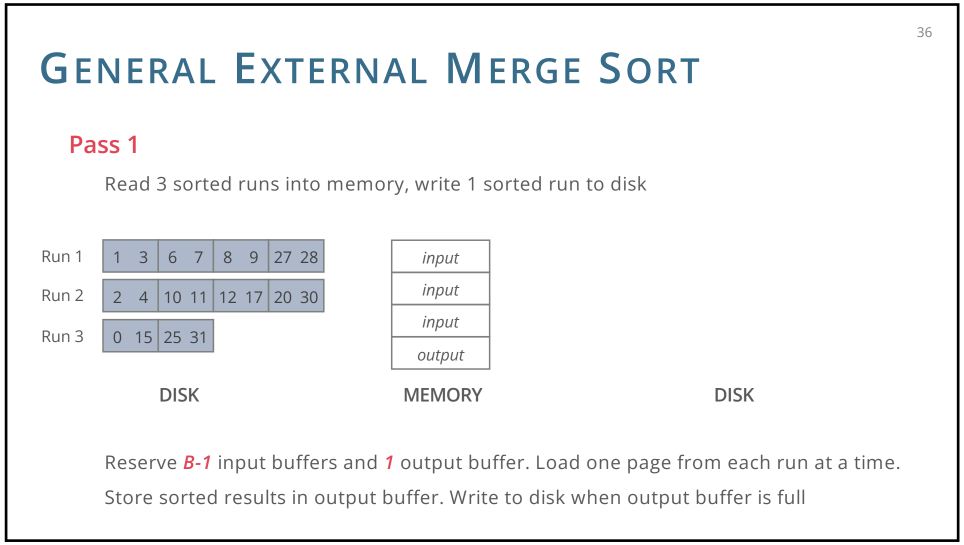
General External Merge Sort

Pass 0, Run 3

Write 2 sorted pages to disk

I/O Total (so far): 20

35



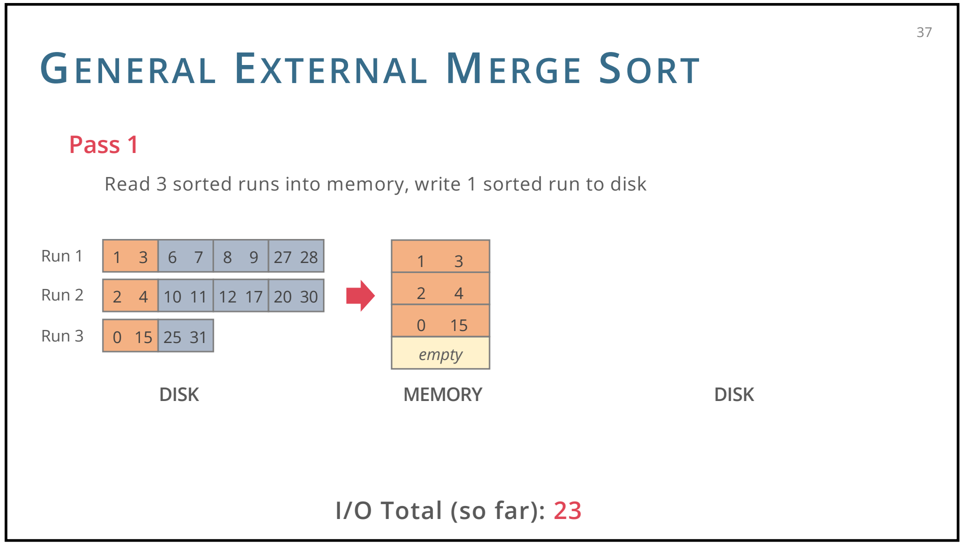
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Reserve *B-1* input buffers and *1* output buffer. Load one page from each run at a time. Store sorted results in output buffer. Write to disk when output buffer is full

36



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1: 1 3 | 6 7 | 8 9 | 27 28

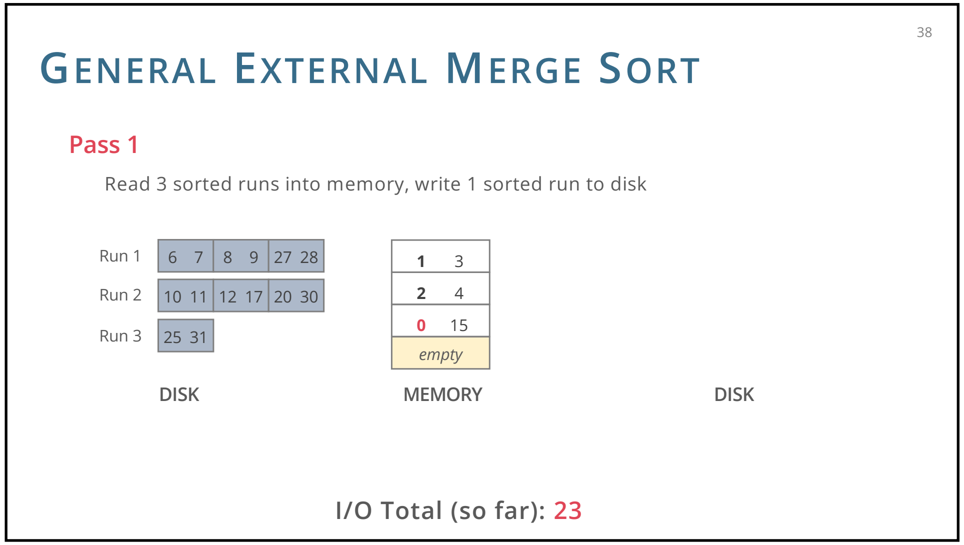
Run 2: 2 4 | 10 11 | 12 17 | 20 30

Run 3: 0 15 | 25 31

1 3
2 4
0 15
empty

DISK MEMORY DISK

I/O Total (so far): 23



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1: 6 7 | 8 9 | 27 28

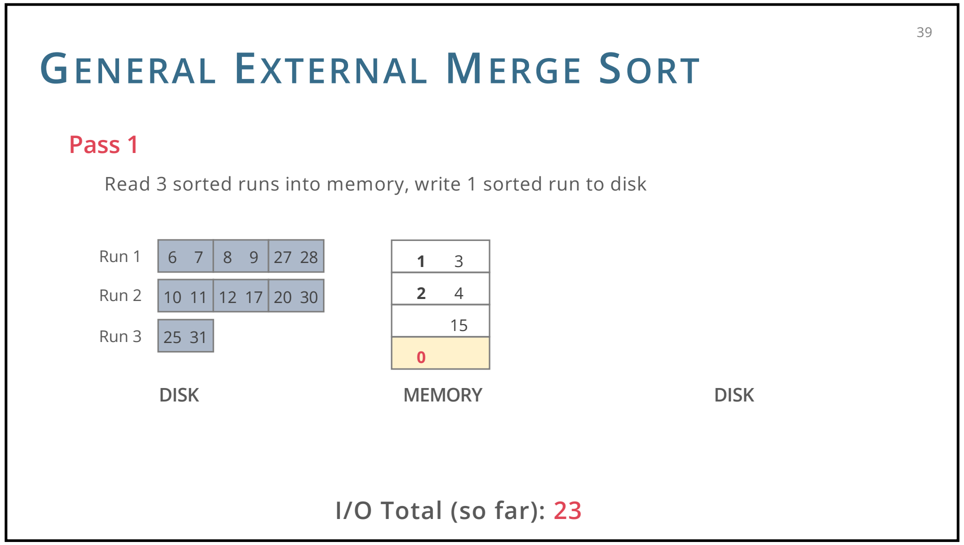
Run 2: 10 11 | 12 17 | 20 30

Run 3: 25 31

1 3
2 4
0 15
empty

DISK MEMORY DISK

I/O Total (so far): 23



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1: 6 7 | 8 9 | 27 28

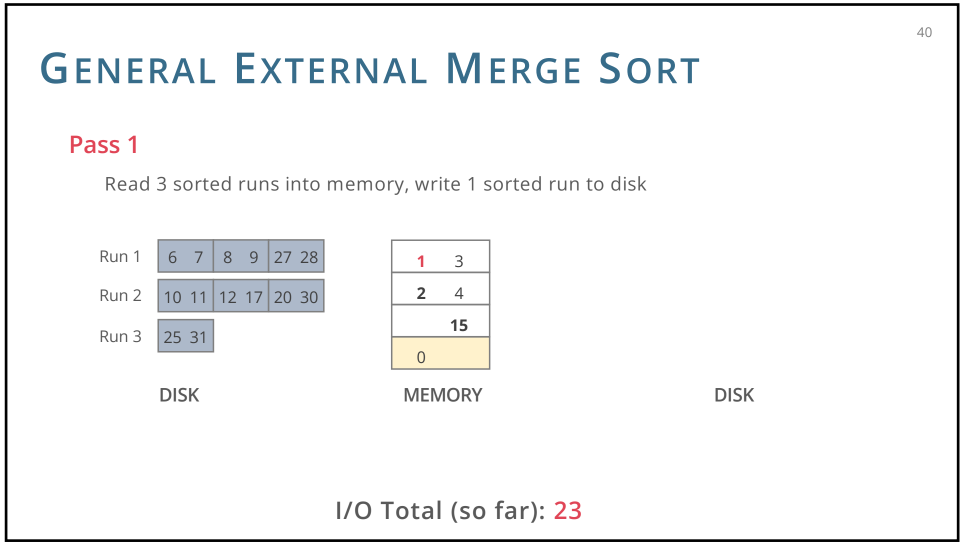
Run 2: 10 11 | 12 17 | 20 30

Run 3: 25 31

1 3
2 4
15
0

DISK MEMORY DISK

I/O Total (so far): 23



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1: 6 7 | 8 9 | 27 28

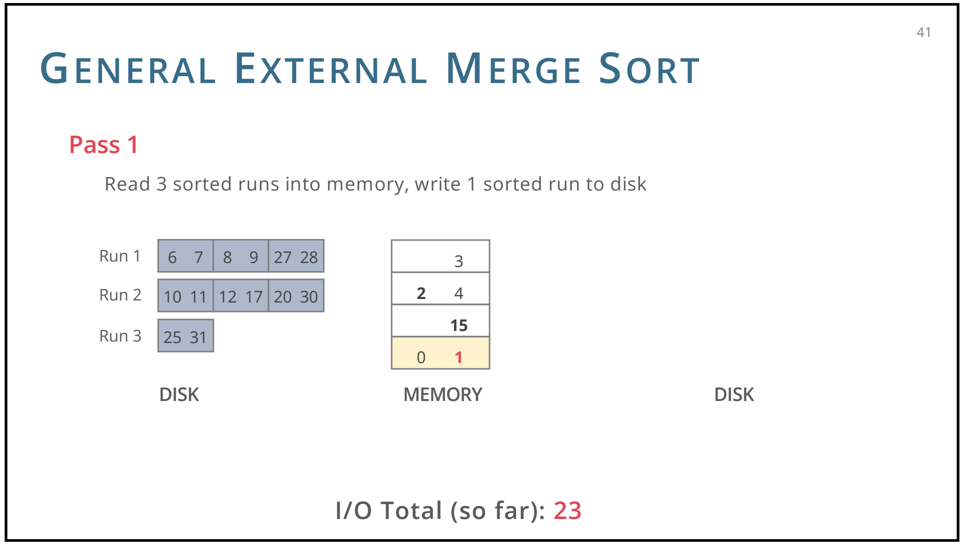
Run 2: 10 11 | 12 17 | 20 30

Run 3: 25 31

1 3
2 4
15
0

DISK MEMORY DISK

I/O Total (so far): 23



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 6 7 | 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

3

2 4

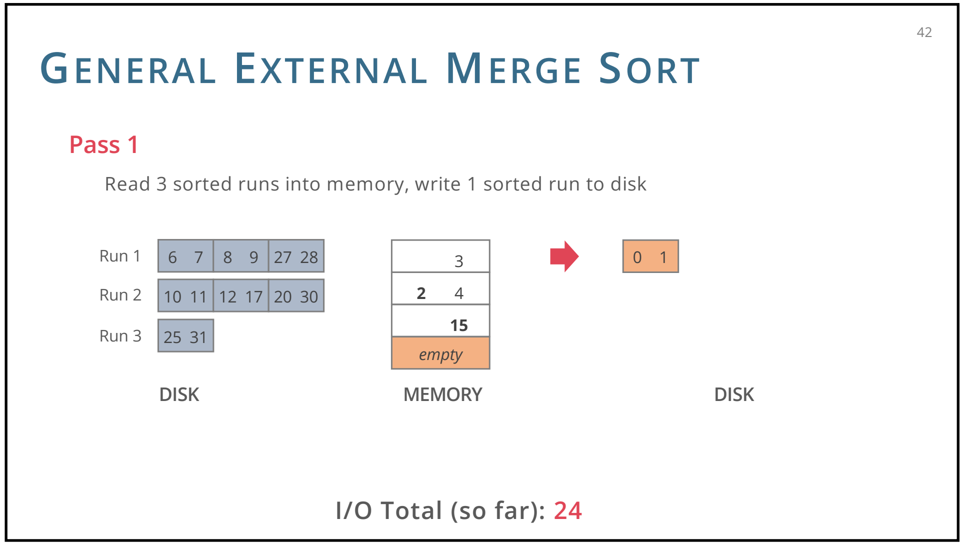
15

0 1

DISK MEMORY DISK

I/O Total (so far): 23

41



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 6 7 | 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

3

2 4

15

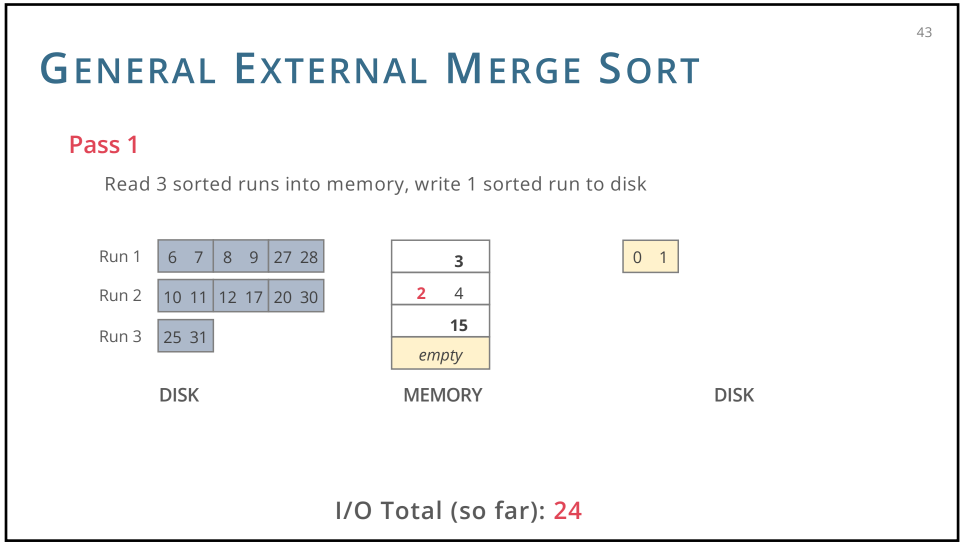
empty

0 1

DISK MEMORY DISK

I/O Total (so far): 24

42



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 6 7 | 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

3

2 4

15

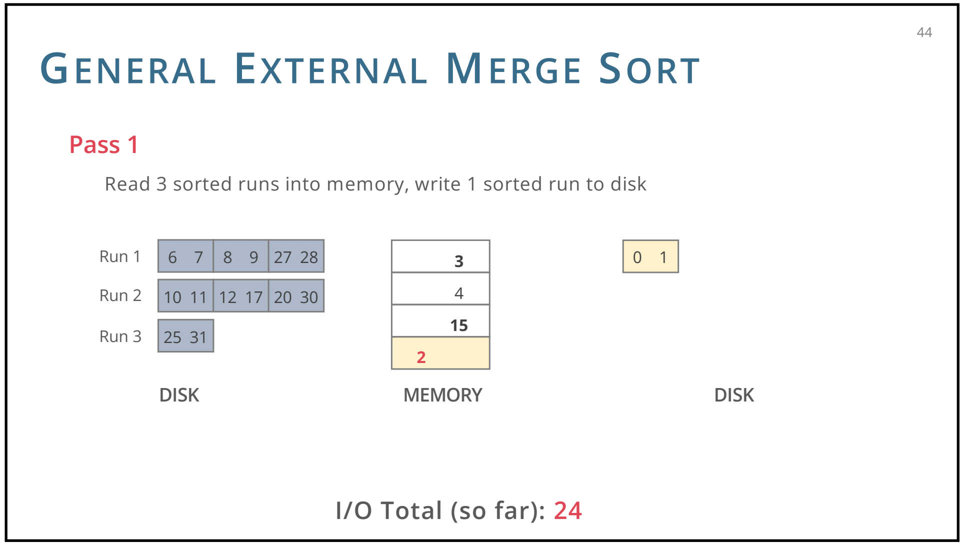
empty

0 1

DISK MEMORY DISK

I/O Total (so far): 24

43



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 6 7 | 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

3

4

15

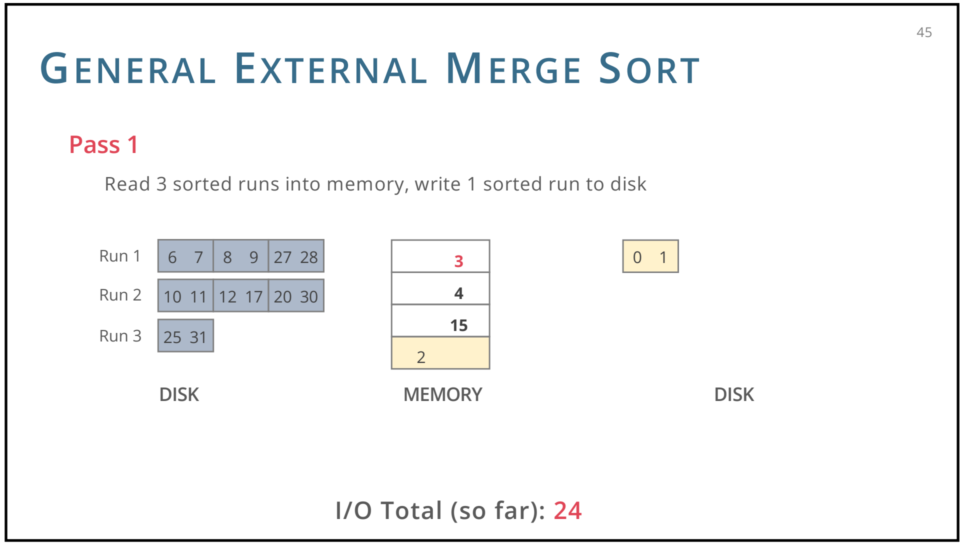
2

0 1

DISK MEMORY DISK

I/O Total (so far): 24

44



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 6 7 | 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

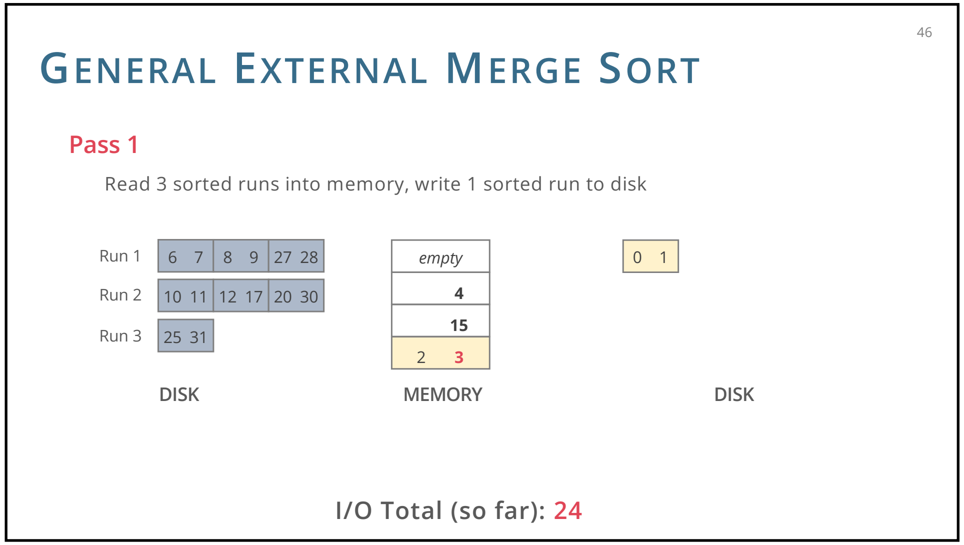
3
4
15
2

0 1

DISK MEMORY DISK

I/O Total (so far): 24

45



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 6 7 | 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

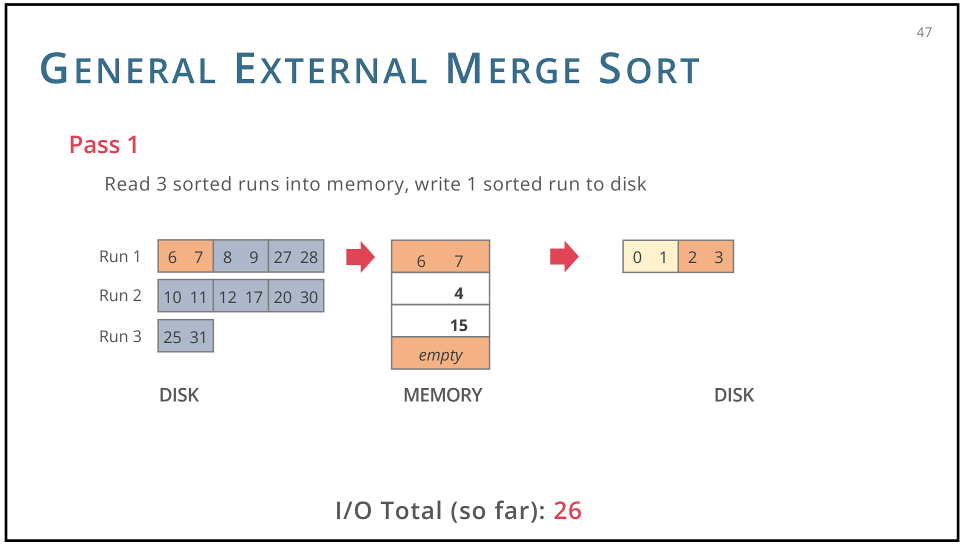
empty
4
15
2 3

0 1

DISK MEMORY DISK

I/O Total (so far): 24

46



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 6 7 | 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

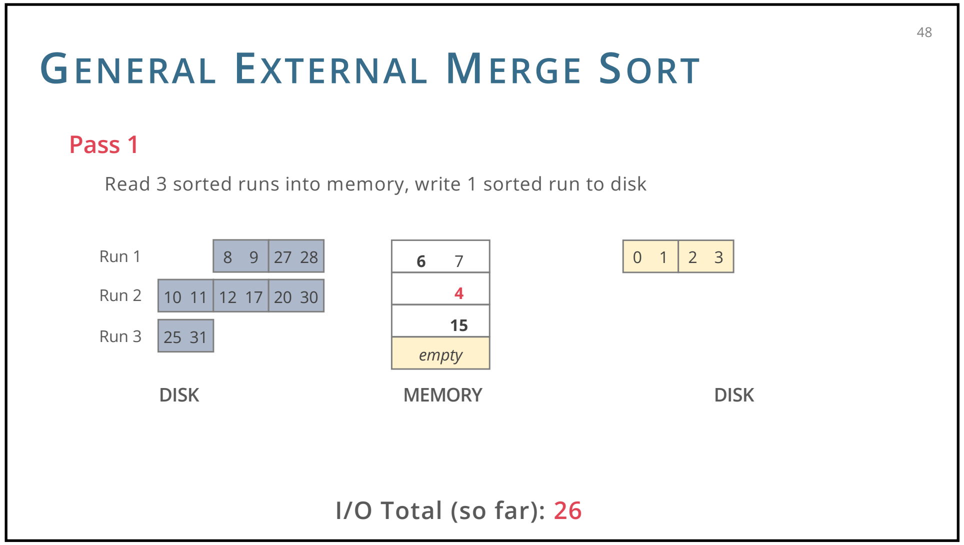
6 7
4
15
empty

0 1 | 2 3

DISK MEMORY DISK

I/O Total (so far): 26

47



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 8 9 | 27 28

Run 2 10 11 | 12 17 | 20 30

Run 3 25 31

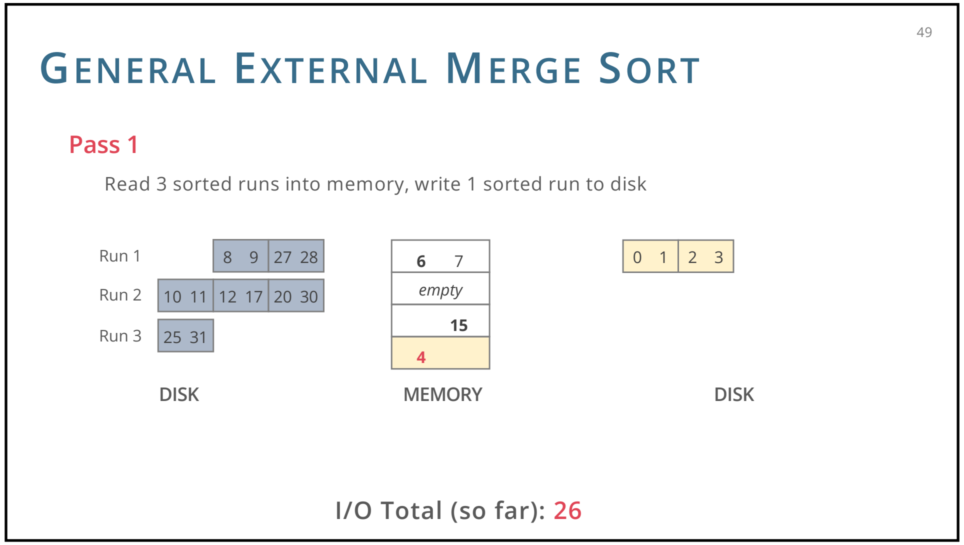
6 7
4
15
empty

0 1 | 2 3

DISK MEMORY DISK

I/O Total (so far): 26

48



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 8 9 27 28

Run 2 10 11 12 17 20 30

Run 3 25 31

DISK

6 7

empty

15

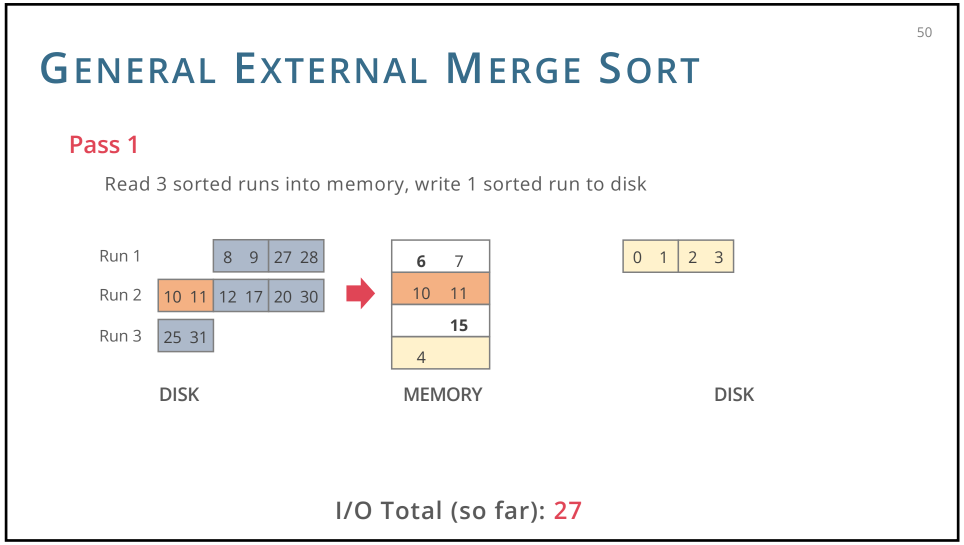
4

MEMORY

0 1 2 3

DISK

I/O Total (so far): 26



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 8 9 27 28

Run 2 10 11 12 17 20 30

Run 3 25 31

DISK

6 7

10 11

15

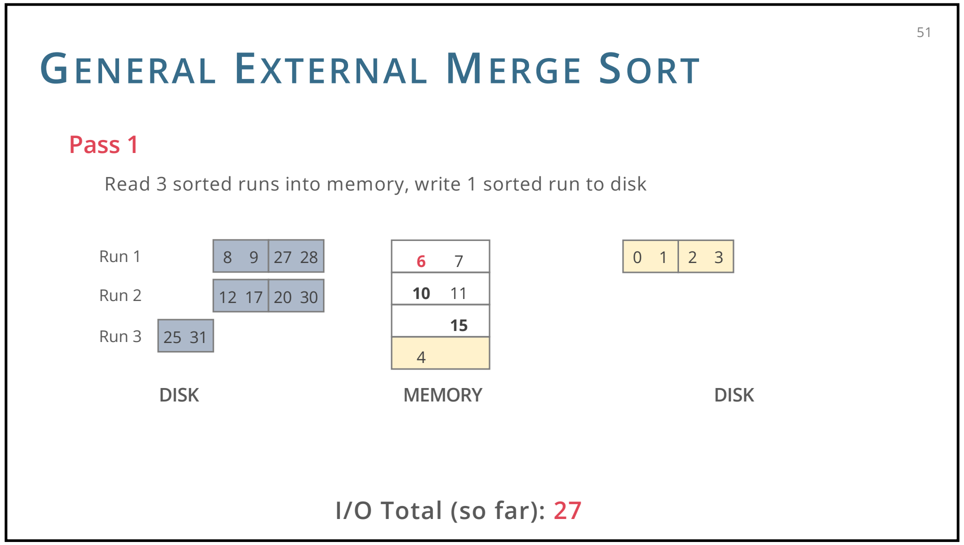
4

MEMORY

0 1 2 3

DISK

I/O Total (so far): 27



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 8 9 27 28

Run 2 12 17 20 30

Run 3 25 31

DISK

6 7

10 11

15

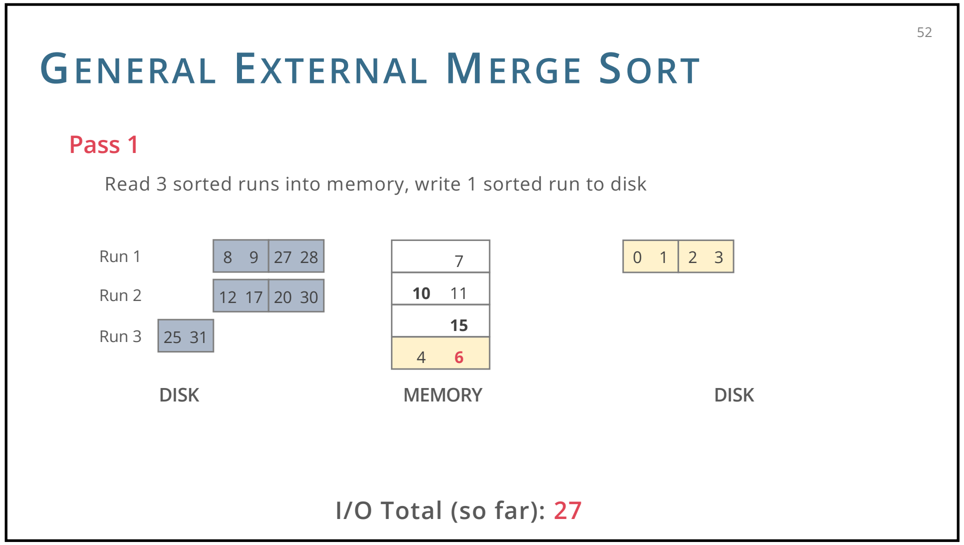
4

MEMORY

0 1 2 3

DISK

I/O Total (so far): 27



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 8 9 27 28

Run 2 12 17 20 30

Run 3 25 31

DISK

7

10 11

15

4 6

MEMORY

0 1 2 3

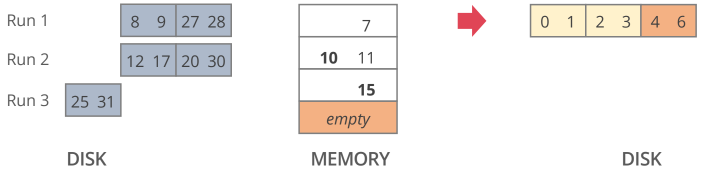
DISK

I/O Total (so far): 27

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



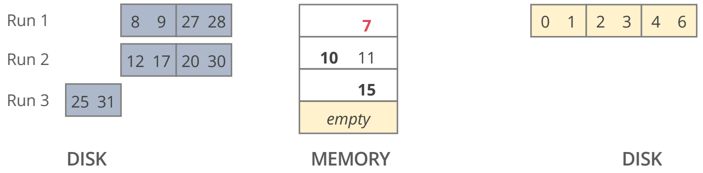
I/O Total (so far): 28

53

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



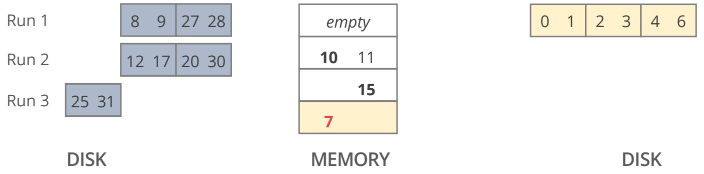
I/O Total (so far): 28

54

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



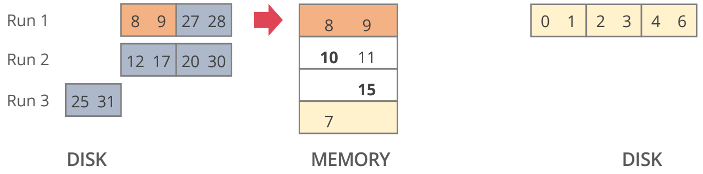
I/O Total (so far): 28

55

General External Merge Sort

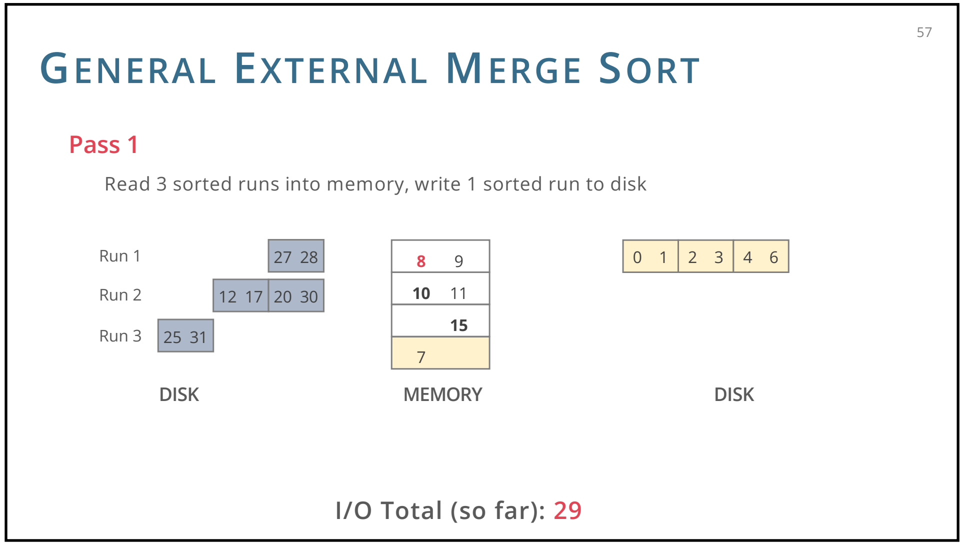
Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



I/O Total (so far): 29

56



57

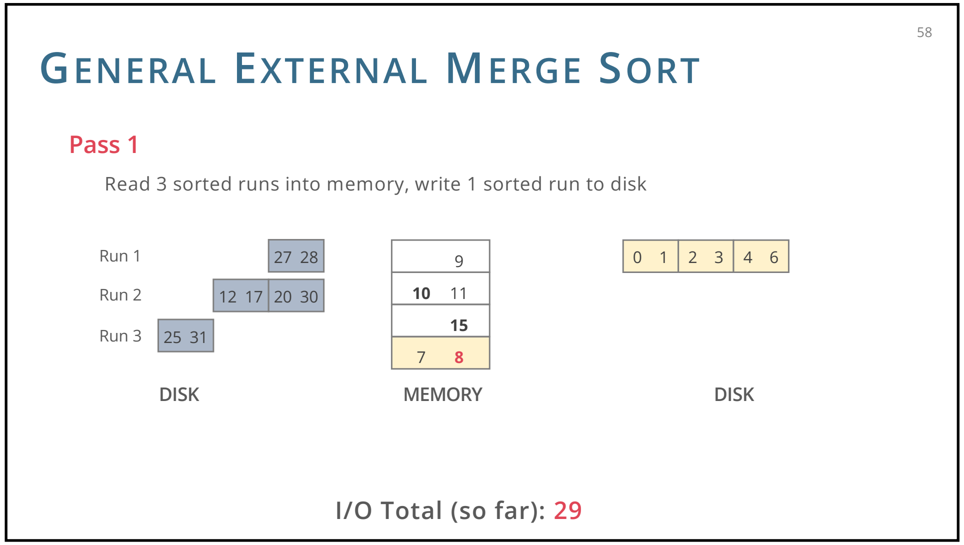
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 29

57



58

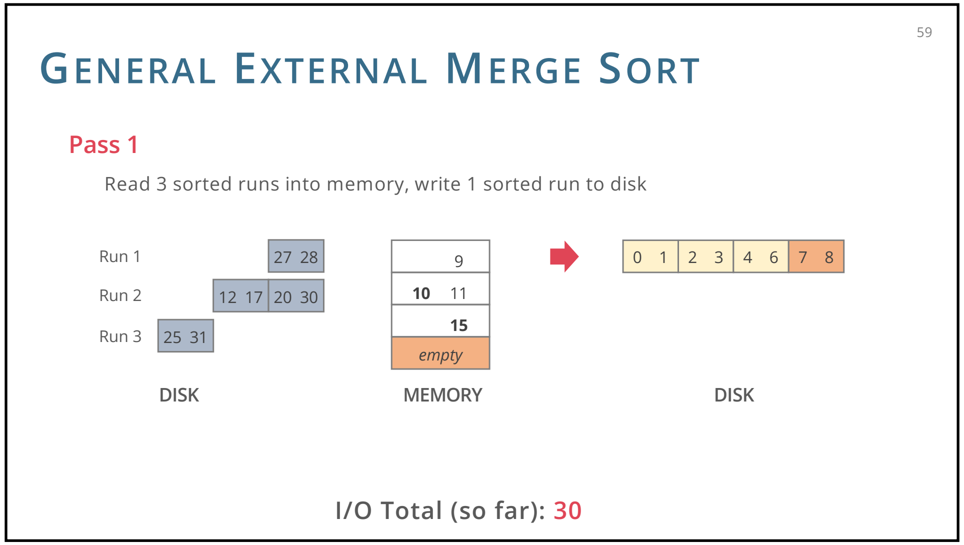
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 29

58



59

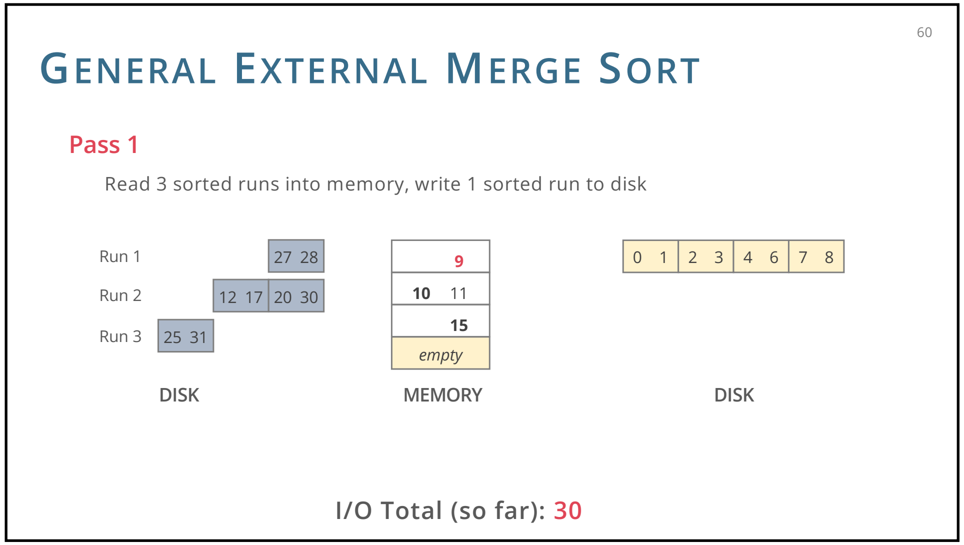
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 30

59



60

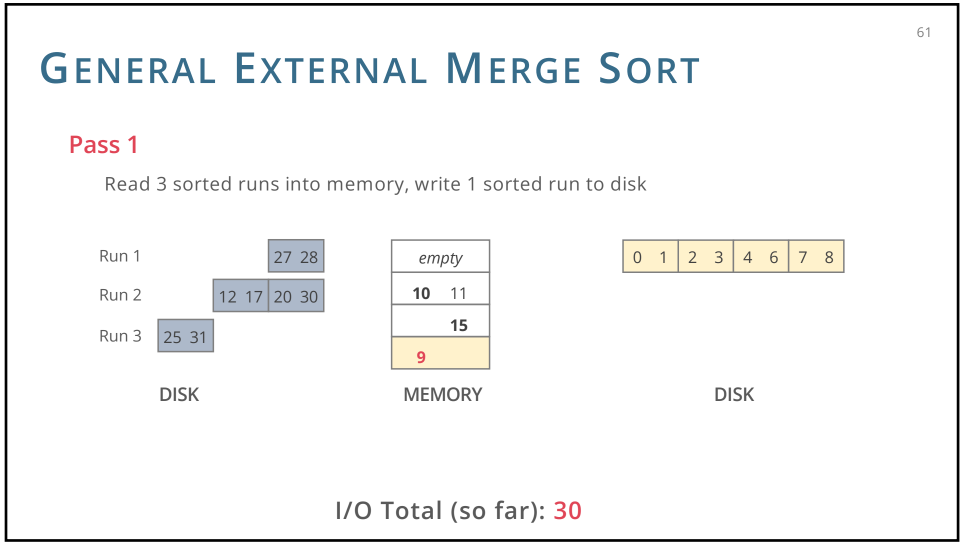
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 30

60



General External Merge Sort

Pass 1

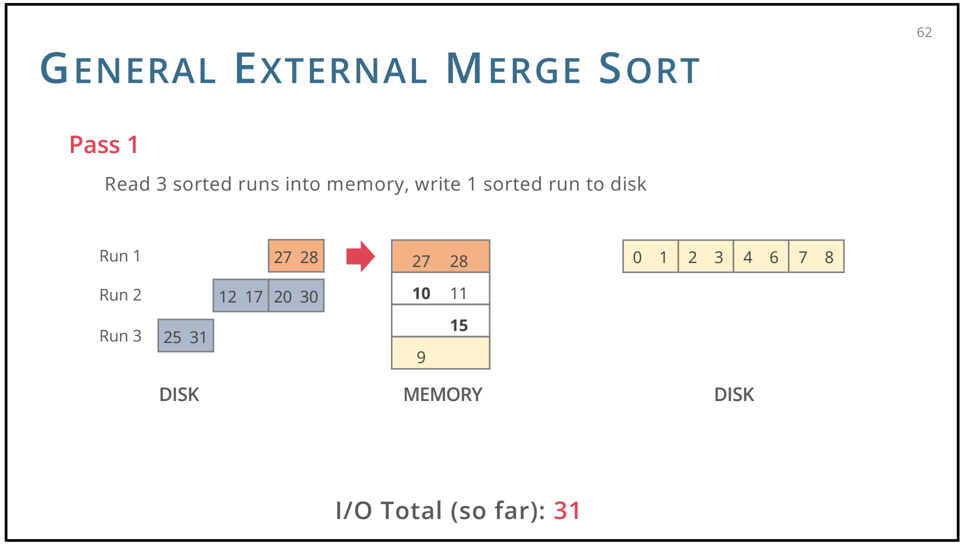
Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 27 28
Run 2 12 17 20 30
Run 3 25 31
DISK

empty
10 11
15
9
MEMORY

0 1 2 3 4 6 7 8
DISK

I/O Total (so far): 30



General External Merge Sort

Pass 1

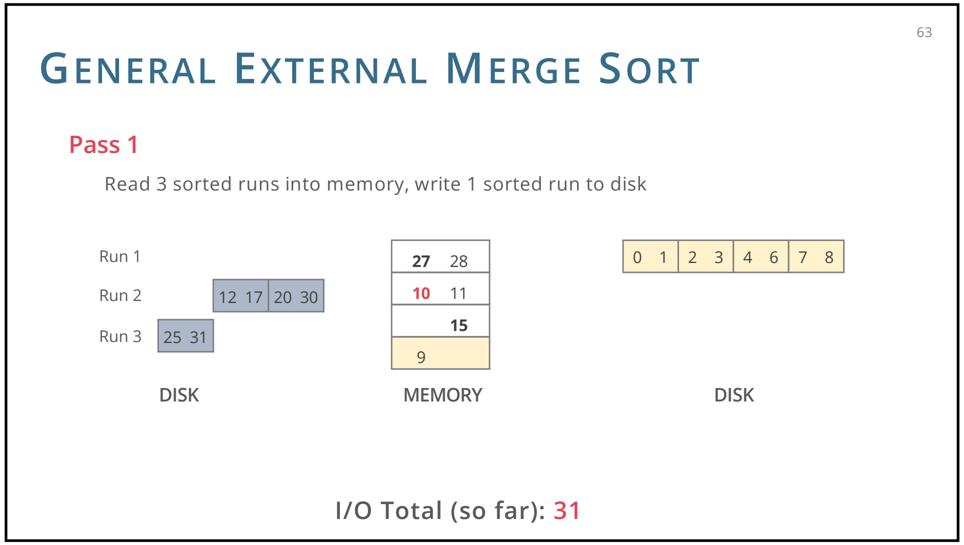
Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1 27 28
Run 2 12 17 20 30
Run 3 25 31
DISK

27 28
10 11
15
9
MEMORY

0 1 2 3 4 6 7 8
DISK

I/O Total (so far): 31



General External Merge Sort

Pass 1

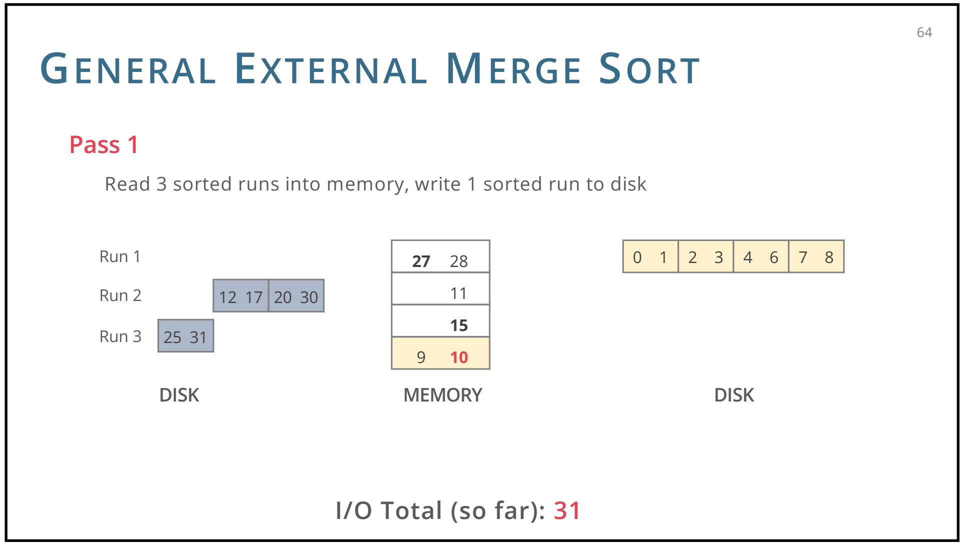
Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1
Run 2 12 17 20 30
Run 3 25 31
DISK

27 28
10 11
15
9
MEMORY

0 1 2 3 4 6 7 8
DISK

I/O Total (so far): 31



General External Merge Sort

Pass 1

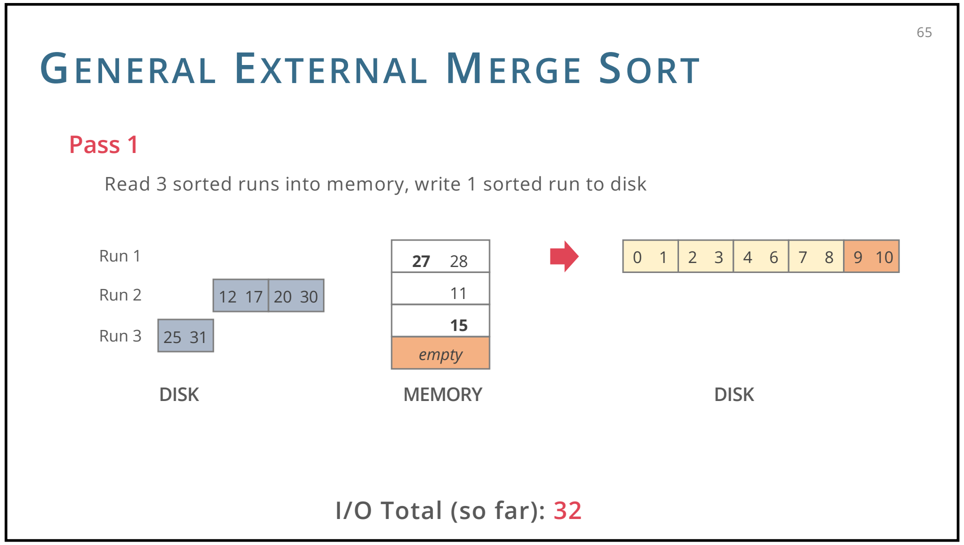
Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1
Run 2 12 17 20 30
Run 3 25 31
DISK

27 28
11
15
9 **10**
MEMORY

0 1 2 3 4 6 7 8
DISK

I/O Total (so far): 31



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 12 17 | 20 30

Run 3 25 31

DISK

27 28
11
15
empty

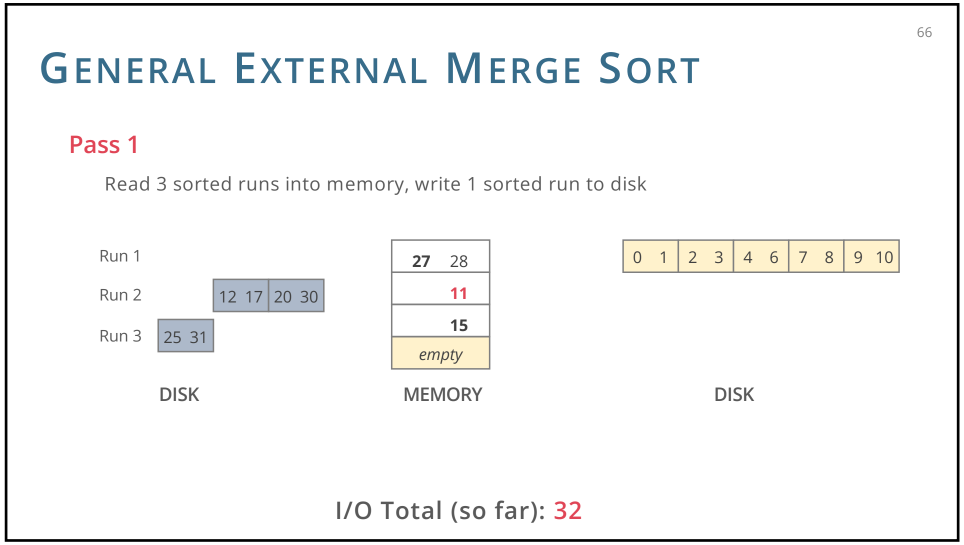
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

DISK

I/O Total (so far): 32

65



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 12 17 | 20 30

Run 3 25 31

DISK

27 28
11
15
empty

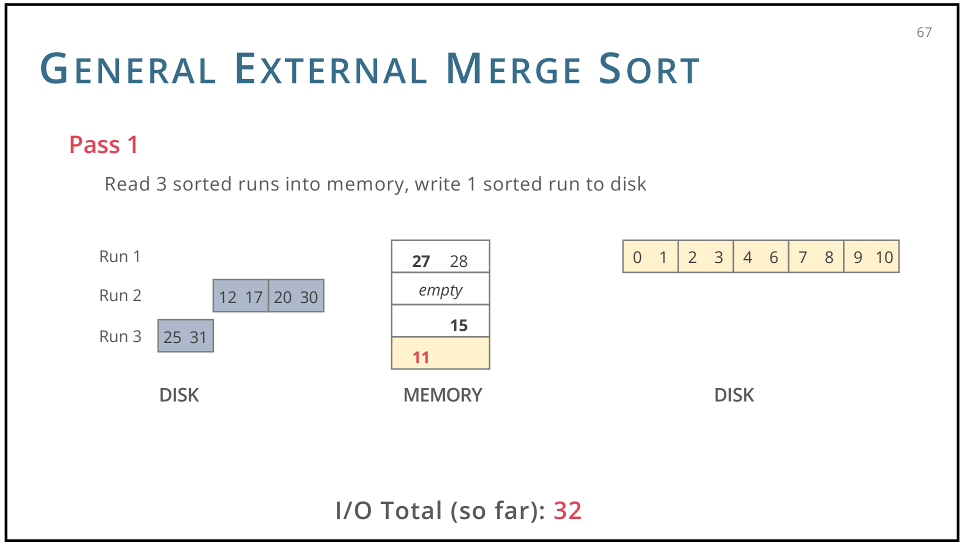
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

DISK

I/O Total (so far): 32

66



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 12 17 | 20 30

Run 3 25 31

DISK

27 28
empty
15
11

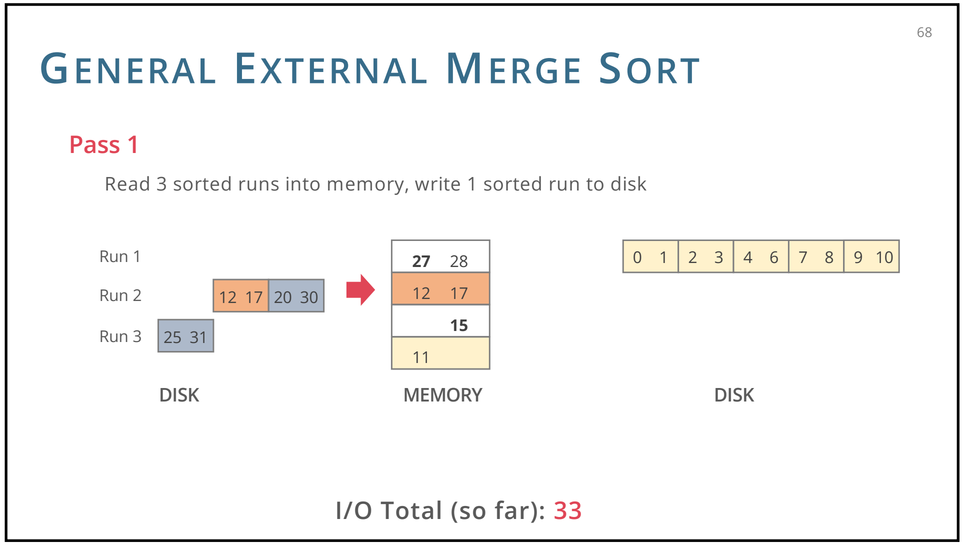
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

DISK

I/O Total (so far): 32

67



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 12 17 | 20 30

Run 3 25 31

DISK

27 28
12 17
15
11

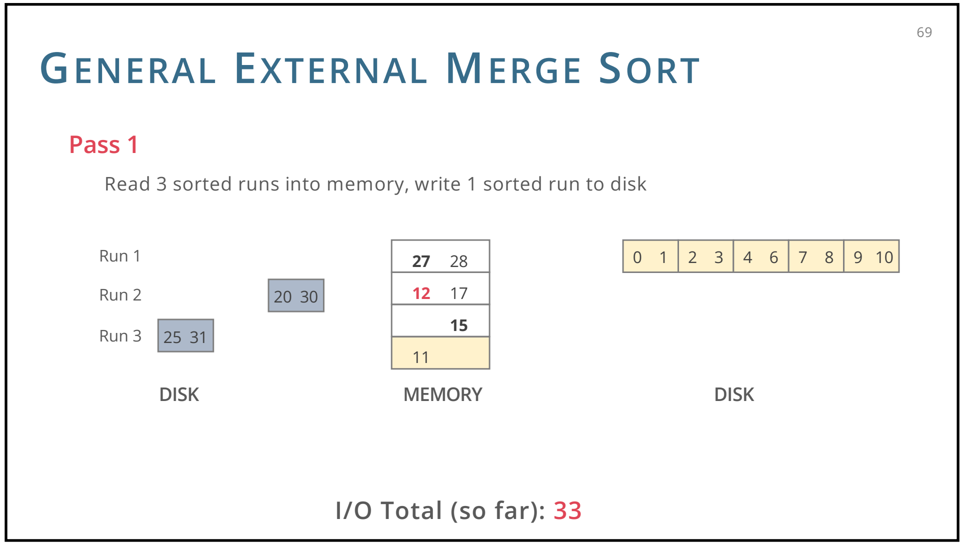
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

DISK

I/O Total (so far): 33

68



69

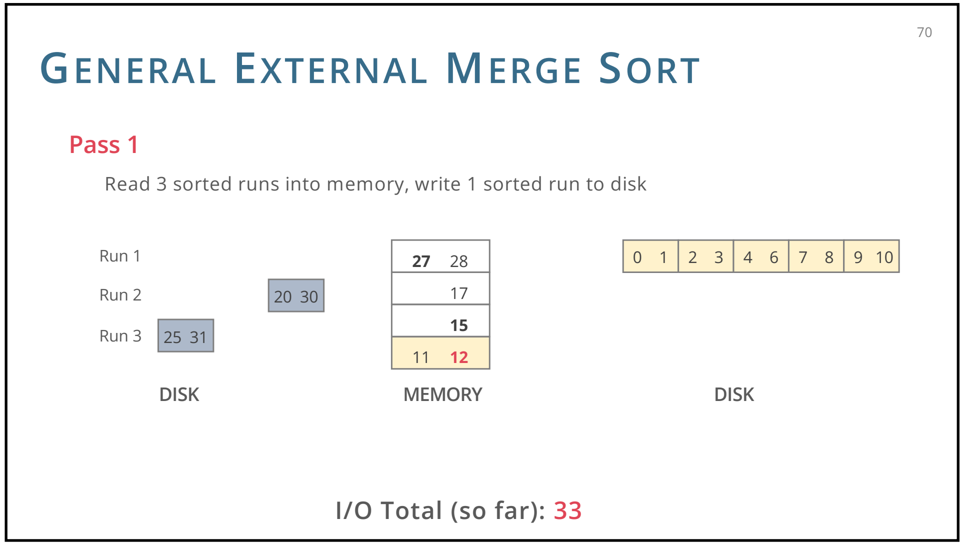
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 33

69



70

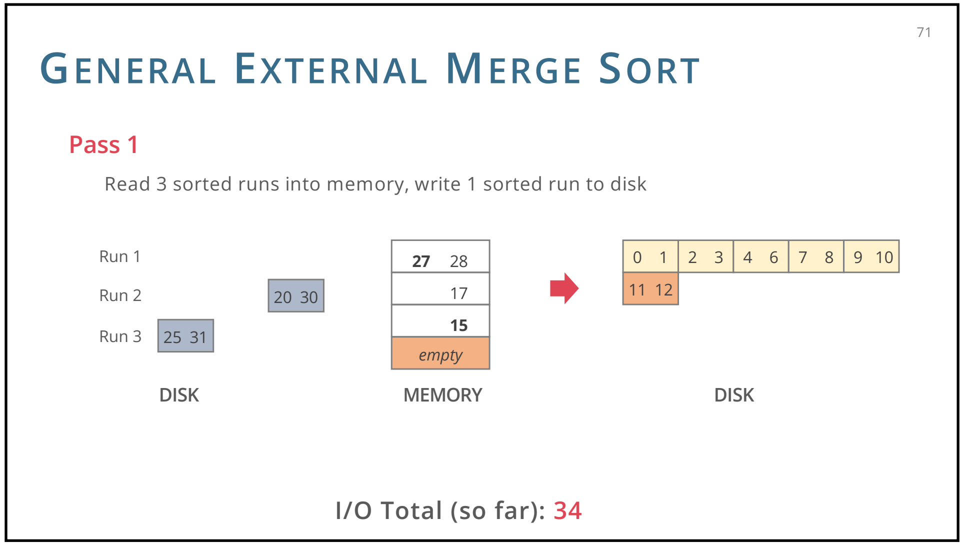
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 33

70



71

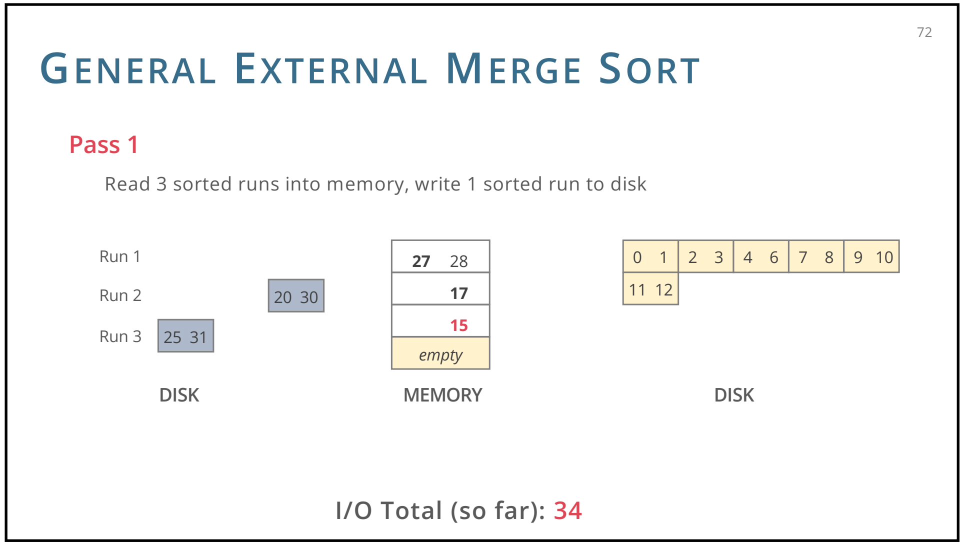
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 34

71



72

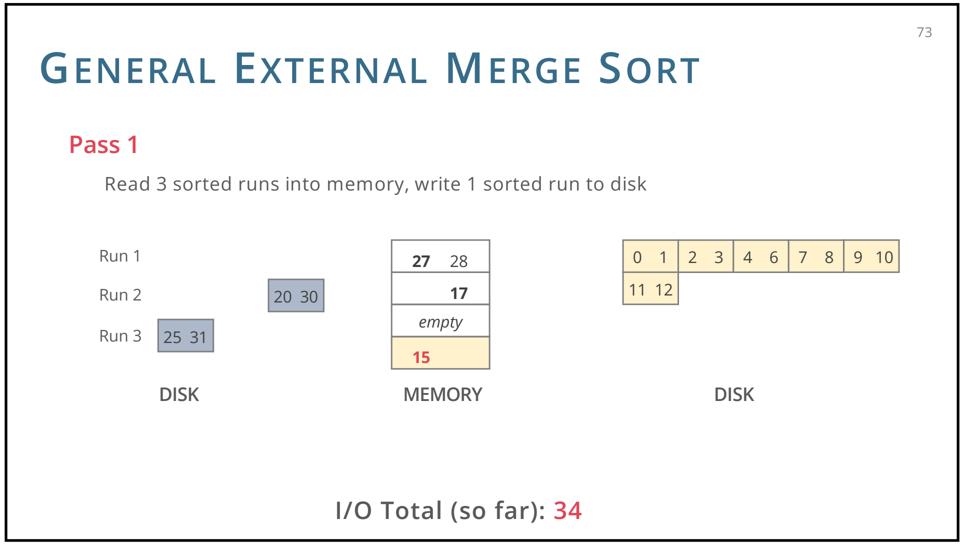
General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

I/O Total (so far): 34

72



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 20 30

Run 3 25 31

DISK

27 28

17

empty

15

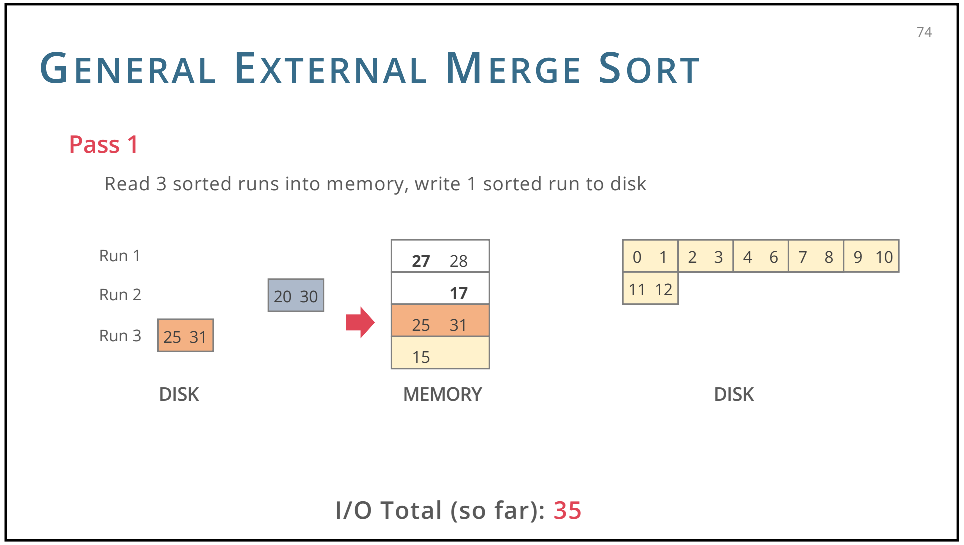
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

11 12

DISK

I/O Total (so far): 34



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 20 30

Run 3 25 31

DISK

27 28

17

25 31

15

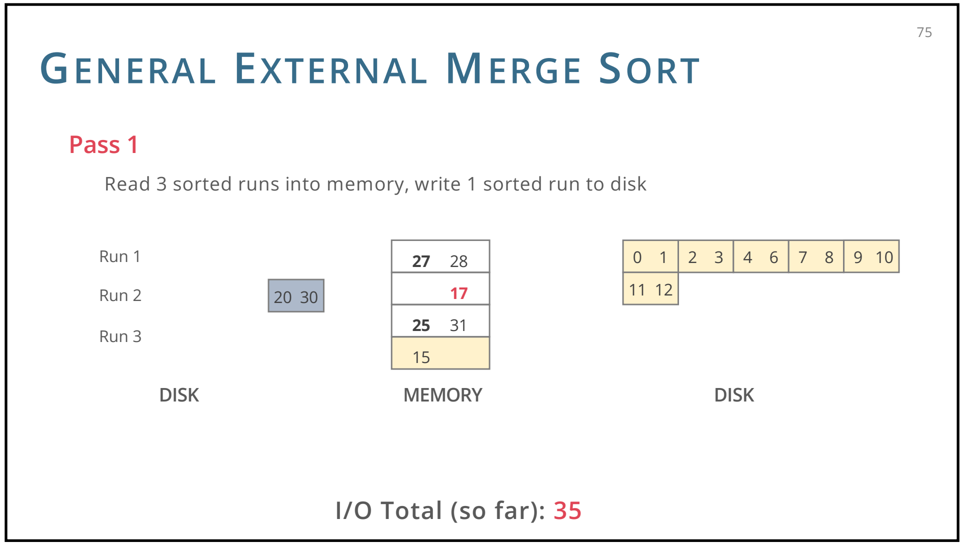
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

11 12

DISK

I/O Total (so far): 35



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 20 30

Run 3

DISK

27 28

17

25 31

15

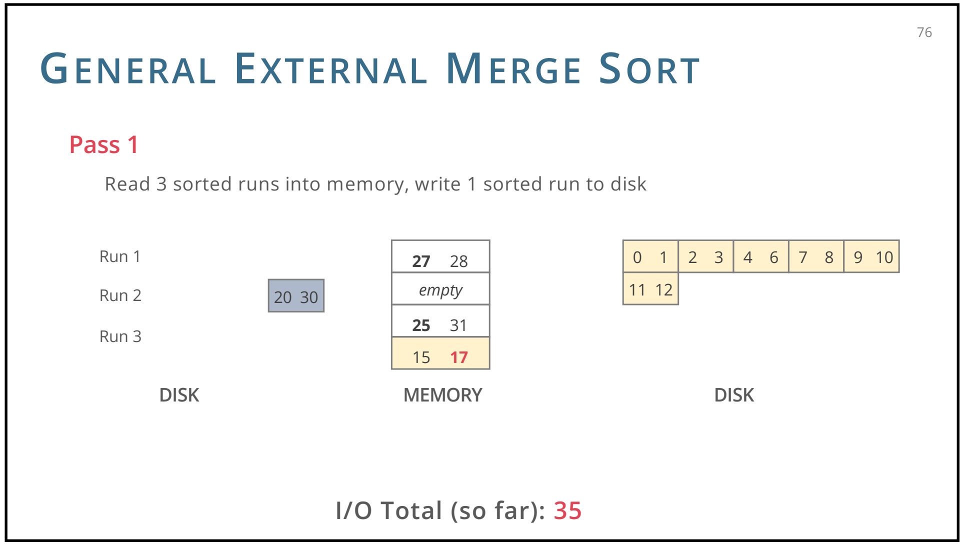
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

11 12

DISK

I/O Total (so far): 35



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 20 30

Run 3

DISK

27 28

empty

25 31

15 **17**

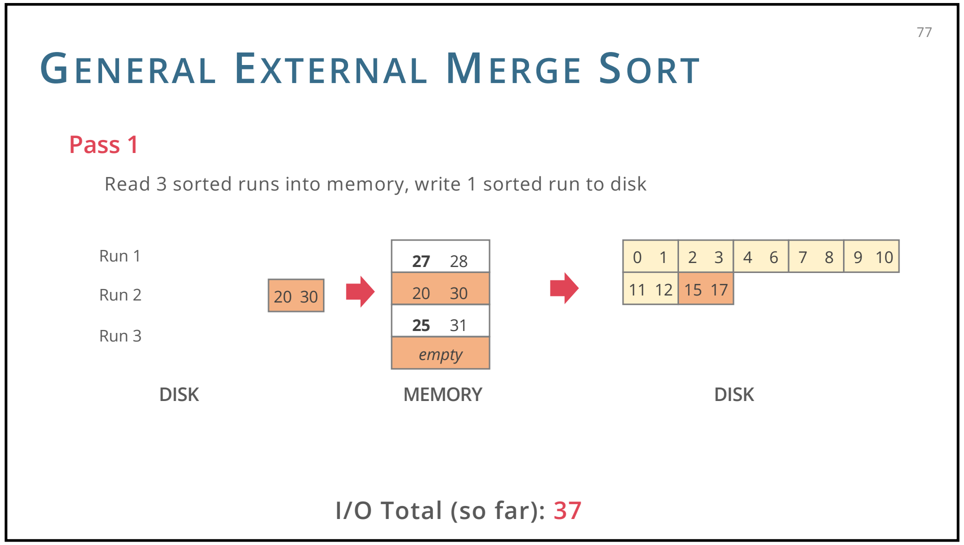
MEMORY

0 1 | 2 3 | 4 6 | 7 8 | 9 10

11 12

DISK

I/O Total (so far): 35



77

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2 20 30

Run 3

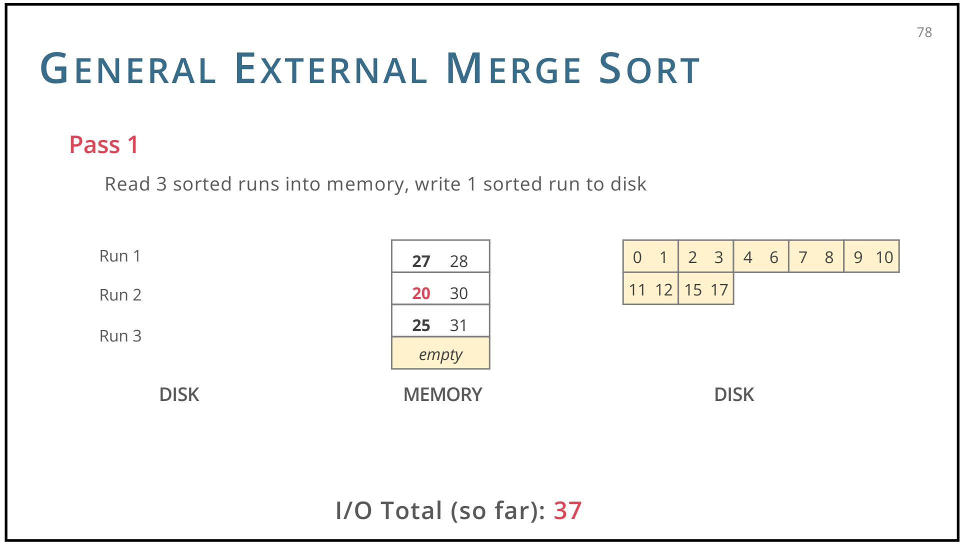
27	28
20	30
25	31
empty	

0 1	2 3	4 6	7 8	9 10
11 12	15 17			

DISK MEMORY DISK

I/O Total (so far): 37

77



78

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2

Run 3

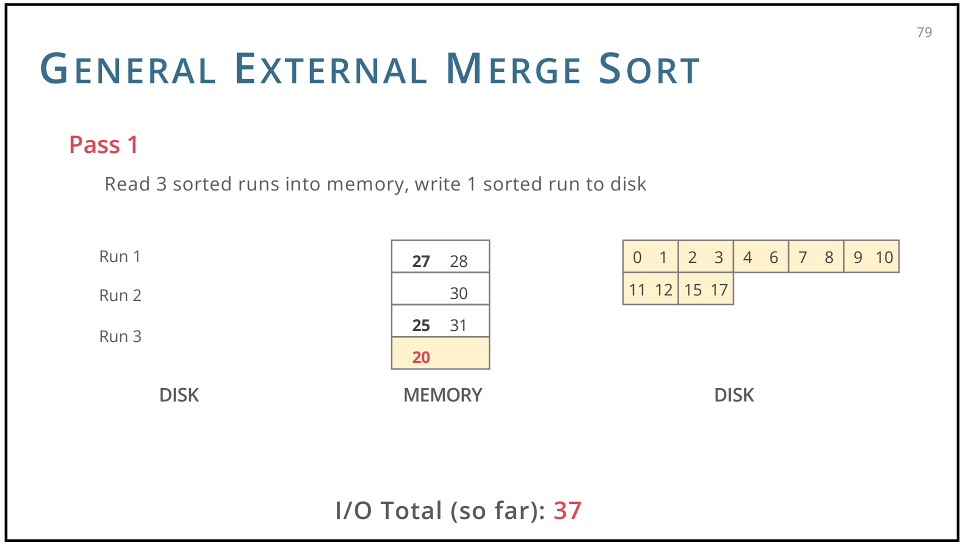
27	28
20	30
25	31
empty	

0 1	2 3	4 6	7 8	9 10
11 12	15 17			

DISK MEMORY DISK

I/O Total (so far): 37

78



79

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2

Run 3

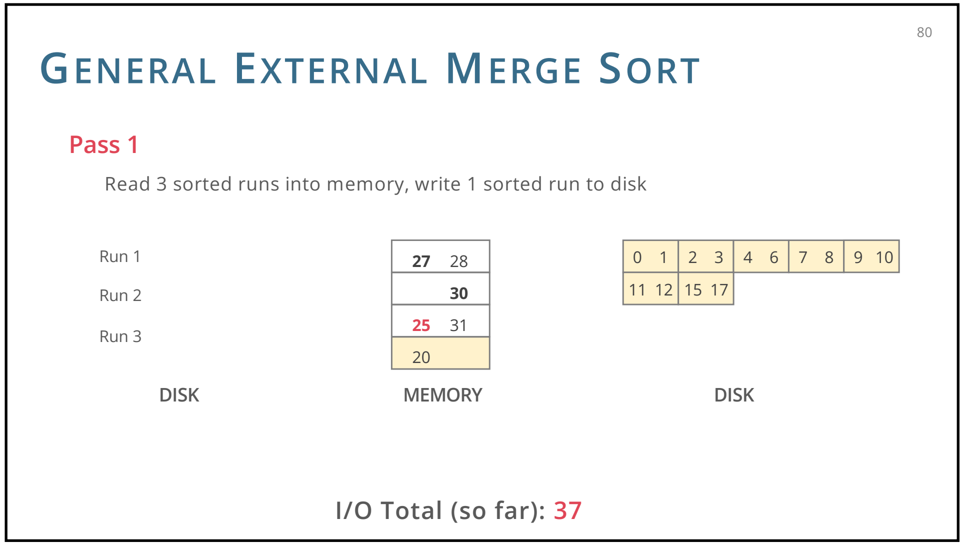
27	28
	30
25	31
20	

0 1	2 3	4 6	7 8	9 10
11 12	15 17			

DISK MEMORY DISK

I/O Total (so far): 37

79



80

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2

Run 3

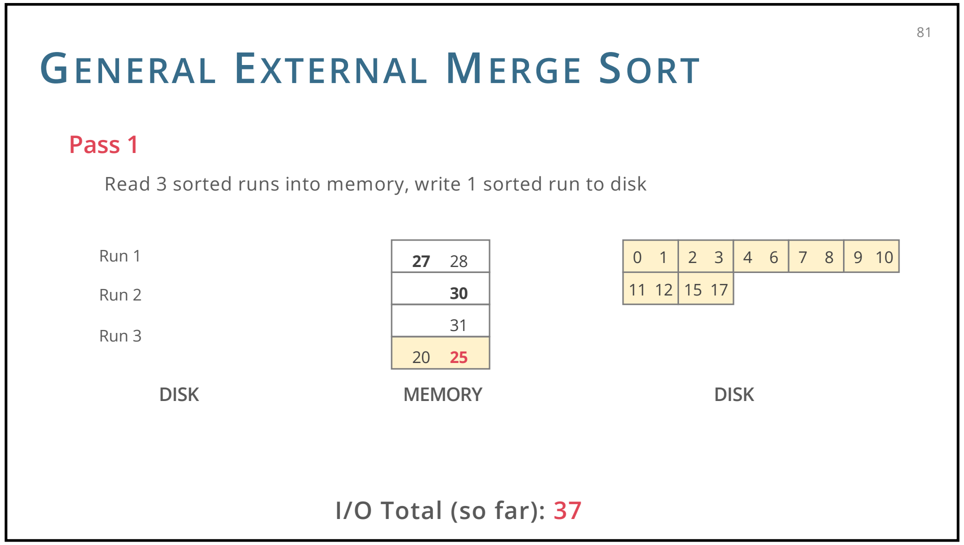
27	28
	30
25	31
20	

0 1	2 3	4 6	7 8	9 10
11 12	15 17			

DISK MEMORY DISK

I/O Total (so far): 37

80



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1
Run 2
Run 3

DISK

27 28
30
31
20 **25**

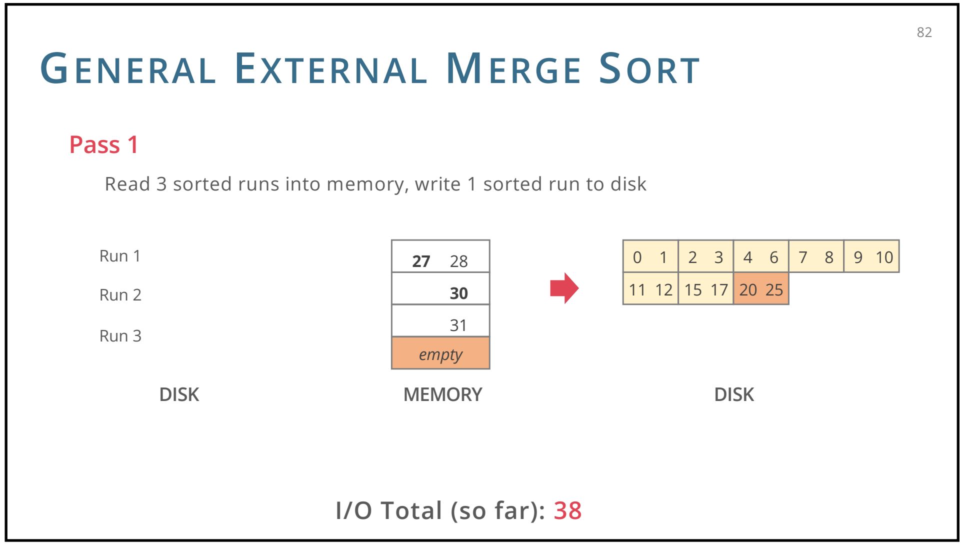
MEMORY

0 1	2 3	4 6	7 8	9 10
11 12	15 17			

DISK

I/O Total (so far): 37

81



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1
Run 2
Run 3

DISK

27 28
30
31
empty

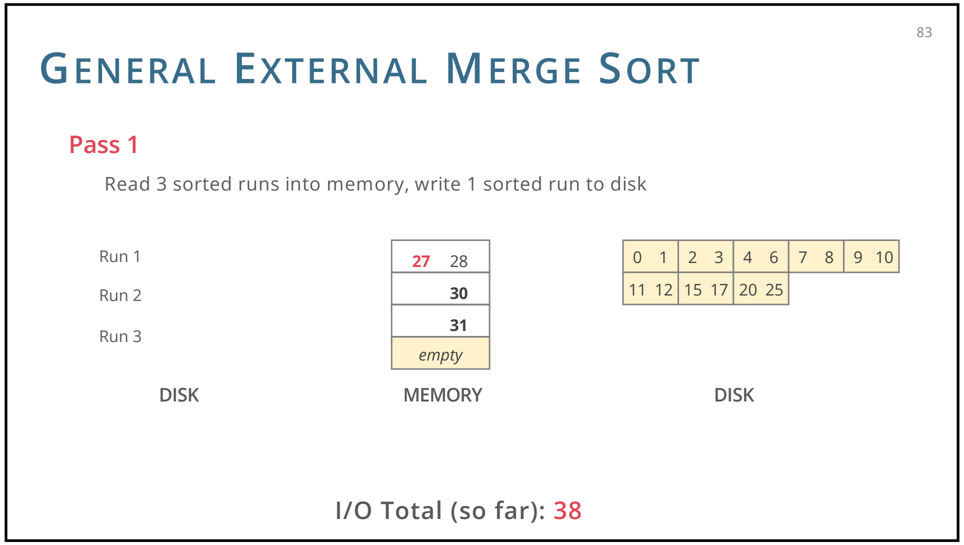
MEMORY

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25		

DISK

I/O Total (so far): 38

82



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1
Run 2
Run 3

DISK

27 28
30
31
empty

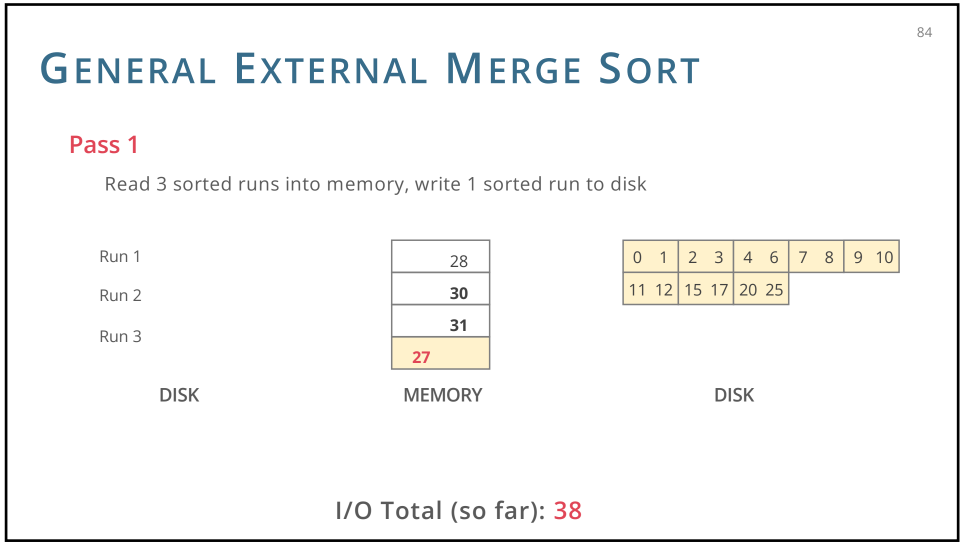
MEMORY

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25		

DISK

I/O Total (so far): 38

83



General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1
Run 2
Run 3

DISK

28
30
31
27

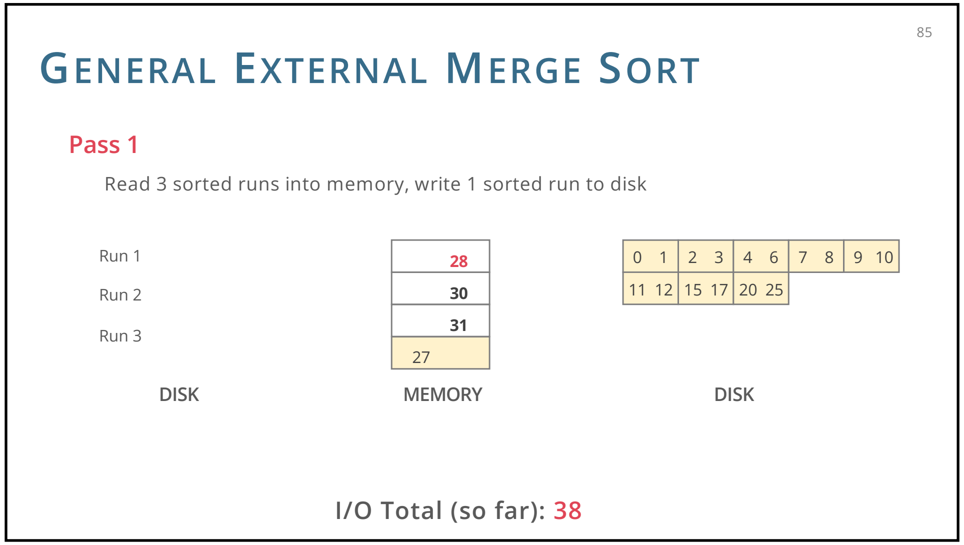
MEMORY

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25		

DISK

I/O Total (so far): 38

84



General External Merge Sort

85

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2

Run 3

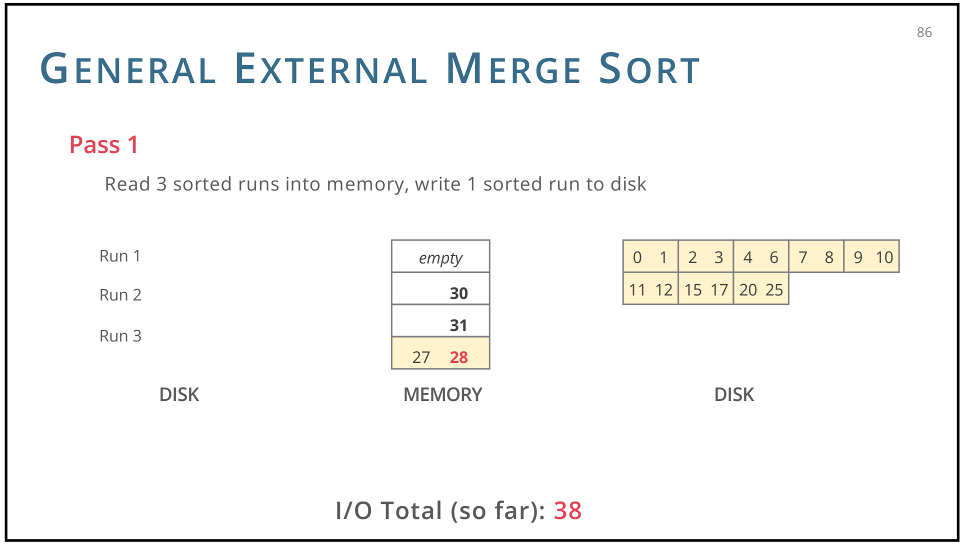
28
30
31
27

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25		

DISK MEMORY DISK

I/O Total (so far): 38

85



General External Merge Sort

86

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2

Run 3

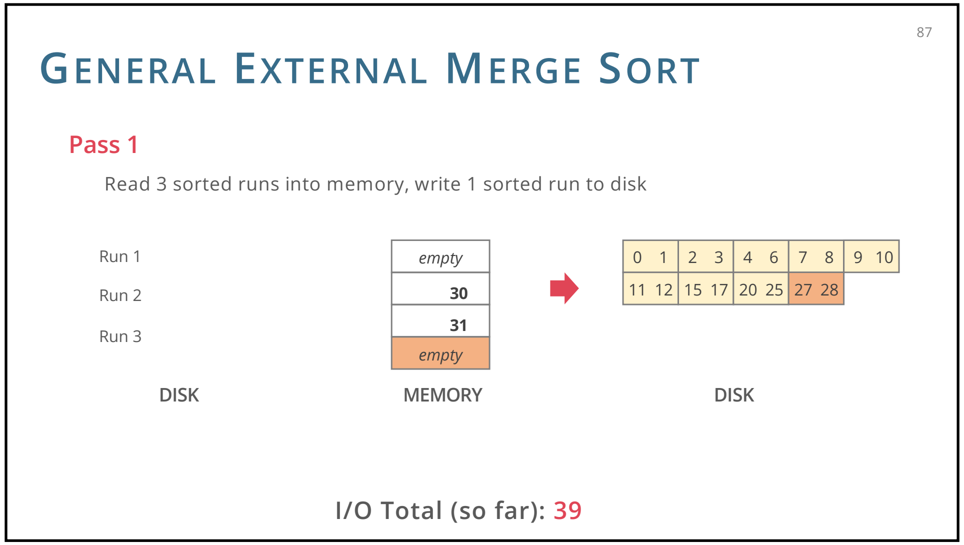
empty
30
31
27 28

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25		

DISK MEMORY DISK

I/O Total (so far): 38

86



General External Merge Sort

87

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2

Run 3

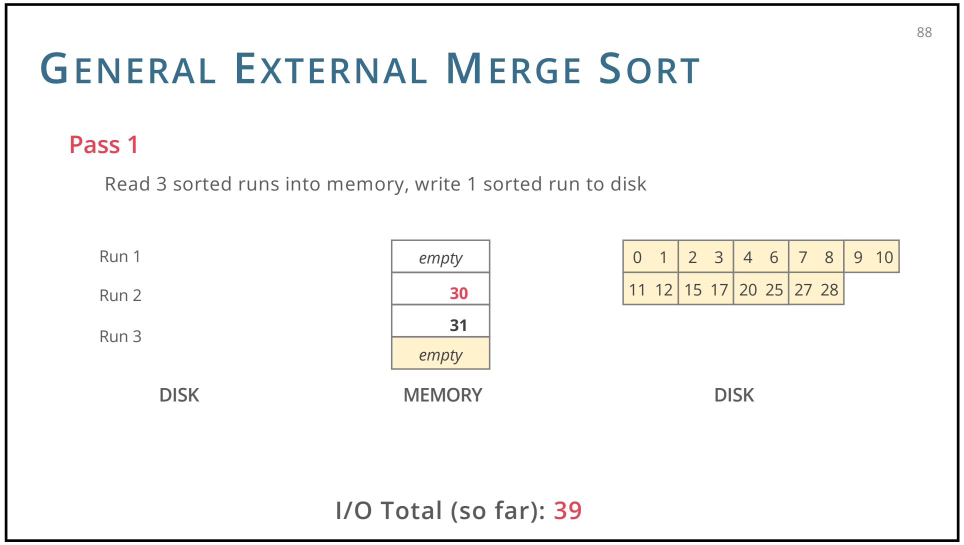
empty
30
31
empty

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25	27 28	

DISK MEMORY DISK

I/O Total (so far): 39

87



General External Merge Sort

88

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk

Run 1

Run 2

Run 3

empty
30
31
empty

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25	27 28	

DISK MEMORY DISK

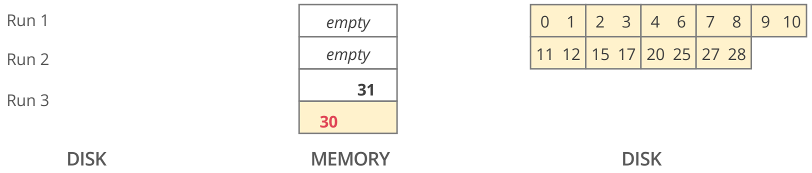
I/O Total (so far): 39

88

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



Run 1

Run 2

Run 3

DISK

MEMORY
empty
empty
31
30

DISK

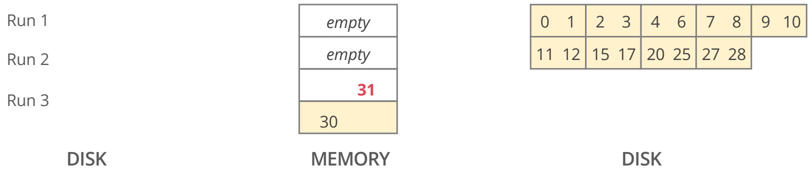
0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25	27 28	

I/O Total (so far): 39

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



Run 1

Run 2

Run 3

DISK

MEMORY
empty
empty
31
30

DISK

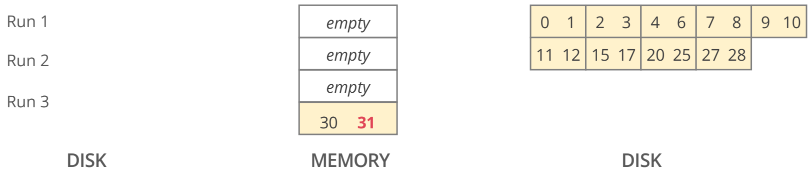
0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25	27 28	

I/O Total (so far): 39

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



Run 1

Run 2

Run 3

DISK

MEMORY
empty
empty
empty
30 31

DISK

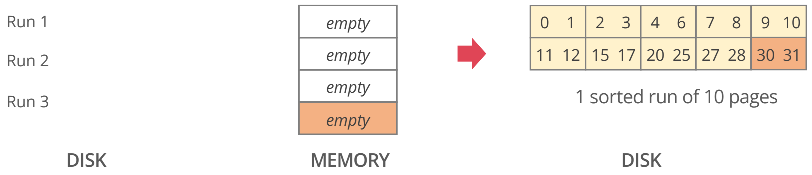
0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25	27 28	

I/O Total (so far): 39

General External Merge Sort

Pass 1

Read 3 sorted runs into memory, write 1 sorted run to disk



Run 1

Run 2

Run 3

DISK

MEMORY
empty
empty
empty
empty

DISK

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25	27 28	30 31

1 sorted run of 10 pages

I/O Total: 40

Sanity Check

N = 10, B = 4

Cost = $2N * (1 + \lceil \log_{B-1}(\lceil N/B \rceil) \rceil)$

$= 2 * 10 * (1 + \lceil \log_3(2.5) \rceil)$

$= 20 * (1 + 1)$

= 40 I/Os ✓

0 1	2 3	4 6	7 8	9 10
11 12	15 17	20 25	27 28	30 31

1 sorted run of 10 pages

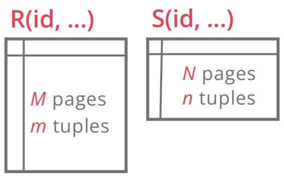
I/O Total: 40

Joins

Joins

Bit of notation:

- *M* = number of pages in *R*
- *m* = number of records in *R* (the **cardinality** of *R*)
- *N* = number of pages in *S*
- *n* = number of records in *S*



We typically exclude the final write's I/O cost

- Don't add the cost of writing the joined output to disk
  - We might decide to stream it to the next operator instead of materializing results!

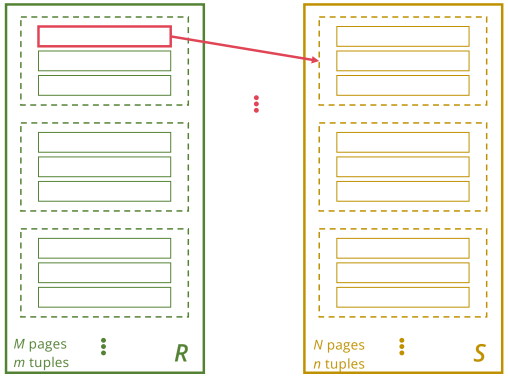
Simple Nested Loops Join (SNLJ)

Direct translation of the definition of join into code

- To perform $R \bowtie_\theta S$, take each tuple in R and scan through S to find the matching S-tuples!

```
foreach tuple r ∈ R:
  foreach tuple s ∈ S:
    if θ(r,s): output r joined with s
```

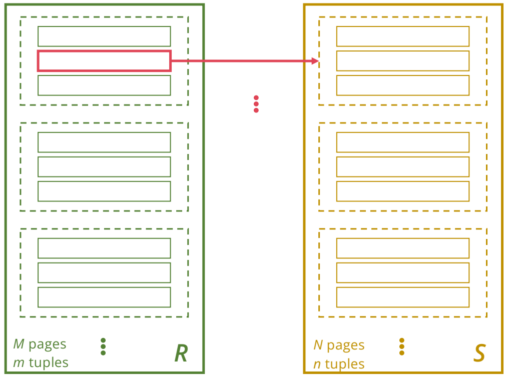
Simple Nested Loops Join (SNLJ)



First iteration of outer loop…

Compare with all tuples in S
… and add matches to result

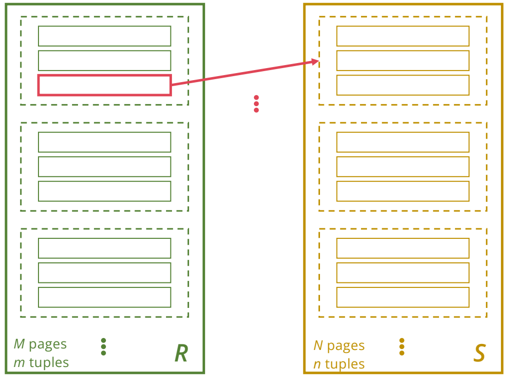
Simple Nested Loops Join (SNLJ)



Second iteration of outer loop…

Compare with all tuples in S
… and add matches to result

Simple Nested Loops Join (SNLJ)



Cost = $M + m \cdot N$

Flipping the order of R and S
can change the I/O cost

Which relation to use as outer?

Page Nested Loops Join (PNLJ)

Can we do better?

We scan S for every tuple in R,

… but we had to load an entire page of R into memory to get that tuple!

Instead of finding the tuples in S that match a tuple in R,

… do the check for all tuples in a page in R at once

Page Nested Loops Join (PNLJ)

SNLJ

```
foreach tuple r ∈ R:
  foreach tuple s ∈ S:
    if $\theta$(r,s): output r joined with s
```

Page Nested Loops Join (PNLJ)

SNLJ (but with page fetches written out explicitly)

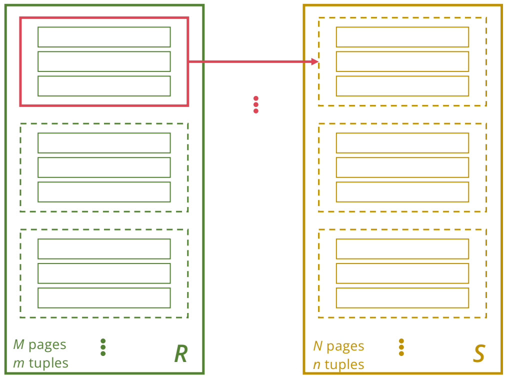
```
foreach page $P_R$ ∈ R:
  foreach tuple r ∈ $P_R$:
    foreach page $P_S$ ∈ S:
      foreach tuple s ∈ $P_S$:
        if $\theta$(r,s): output r joined with s
```

Page Nested Loops Join (PNLJ)

PNLJ

```
foreach page $P_R$ ∈ R:
  foreach page $P_S$ ∈ S:
    foreach tuple r ∈ $P_R$:          flipped loops
      foreach tuple s ∈ $P_S$:
        if $\theta$(r,s): output r joined with s
```

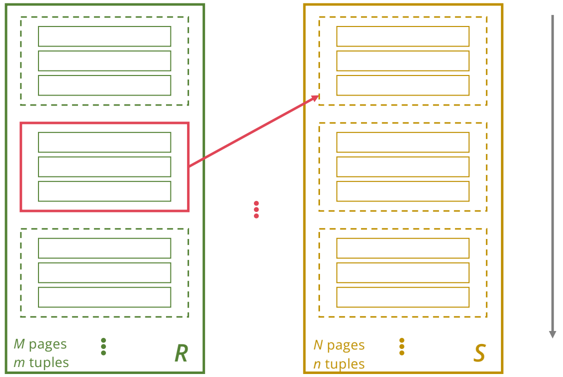
Page Nested Loops Join (PNLJ)



First iteration of outer loop...

Compare with all tuples in *S*
... and add matches to result

Page Nested Loops Join (PNLJ)

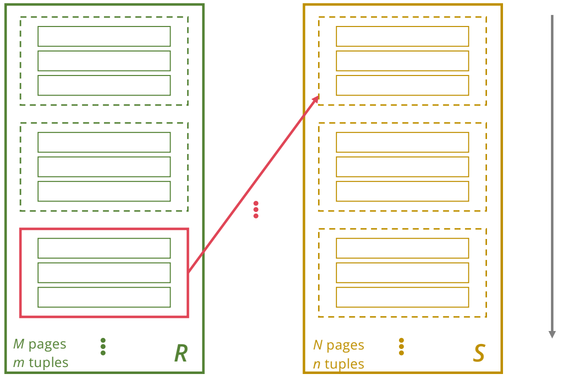


Second iteration of outer loop…

Compare with all tuples in *S*

… and add matches to result

Page Nested Loops Join (PNLJ)



Cost = $M + M \cdot N$

Block Nested Loops Join (BNLJ)

Can we do even better?

We only use three page of memory for PNLJ

… (one buffer for *R*, one buffer for *S*, one output buffer),

… but we usually have more memory!

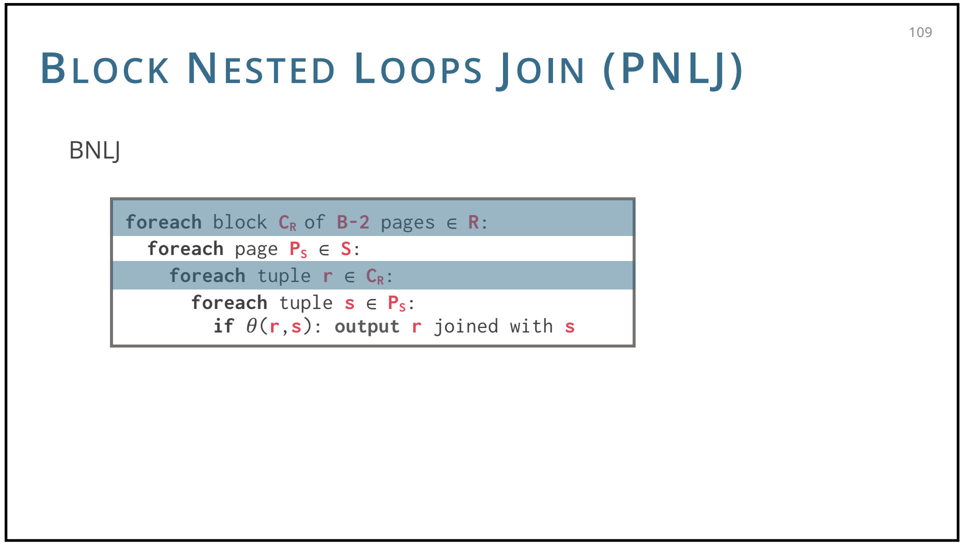
Instead of fetching one page of *R* at a time,

.. why not fetch as many pages of *R* as we can fit (*B - 2* pages)!

Block Nested Loops Join (PNLJ)

PNLJ

```
foreach page P_R ∈ R:
  foreach page P_S ∈ S:
    foreach tuple r ∈ P_R:
      foreach tuple s ∈ P_S:
        if θ(r,s): output r joined with s
```

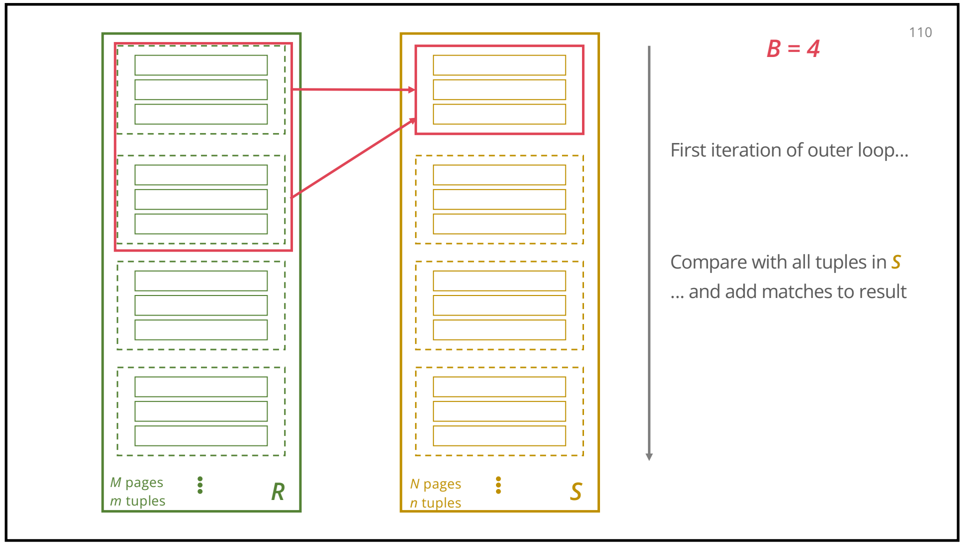


Block Nested Loops Join (PNLJ)

BNLJ

```
foreach block $C_R$ of B-2 pages ∈ R:
  foreach page $P_S$ ∈ S:
    foreach tuple r ∈ $C_R$:
      foreach tuple s ∈ $P_S$:
        if $θ$(r,s): output r joined with s
```

109

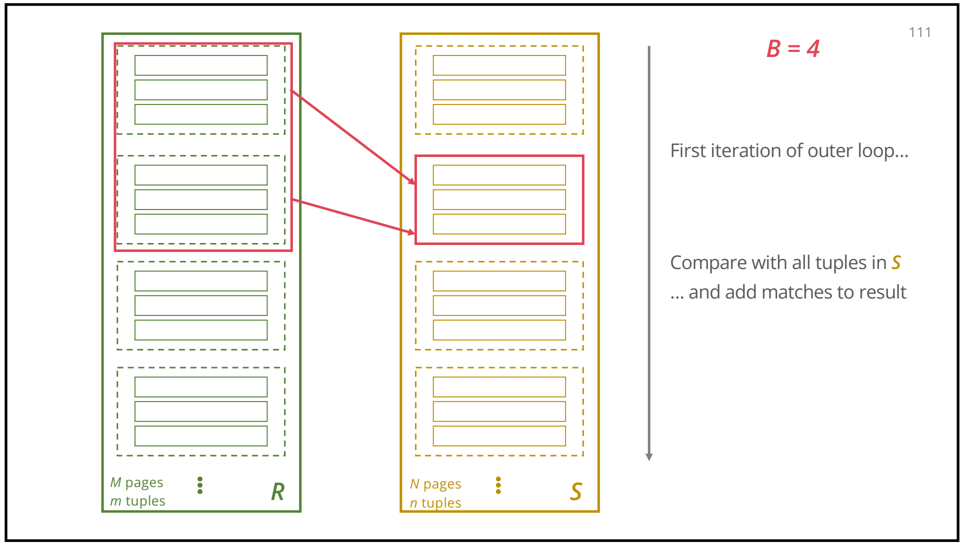


B = 4

First iteration of outer loop…

Compare with all tuples in *S*
… and add matches to result

110

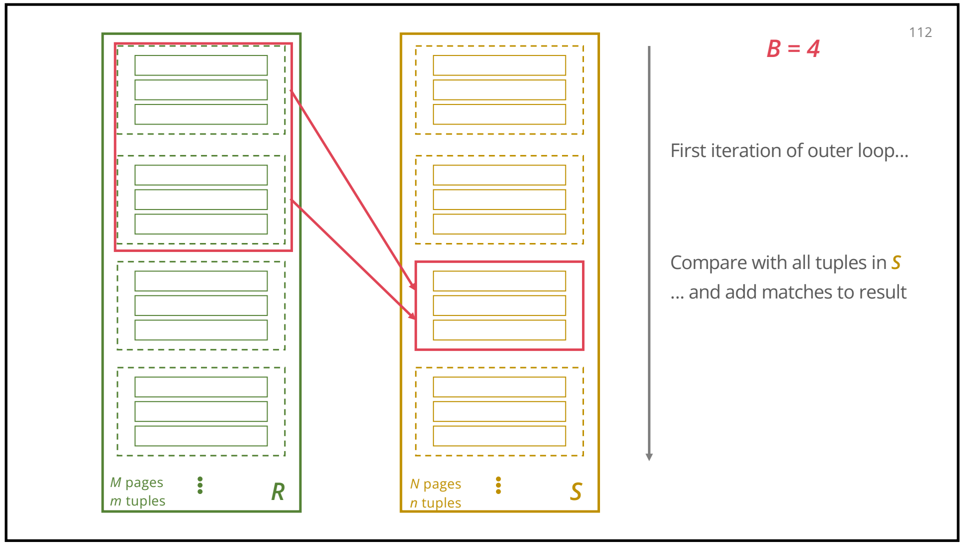


B = 4

First iteration of outer loop…

Compare with all tuples in *S*
… and add matches to result

111

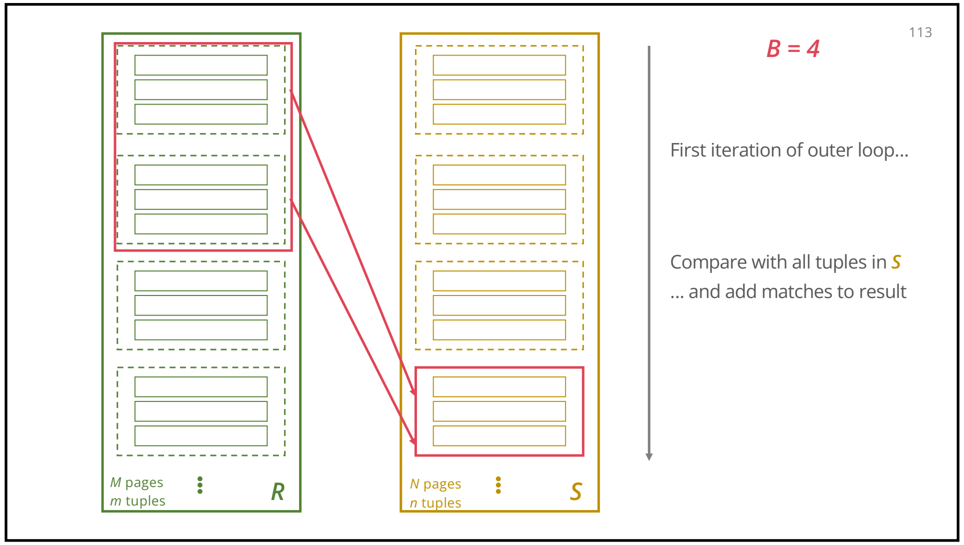


B = 4

First iteration of outer loop…

Compare with all tuples in *S*
… and add matches to result

112

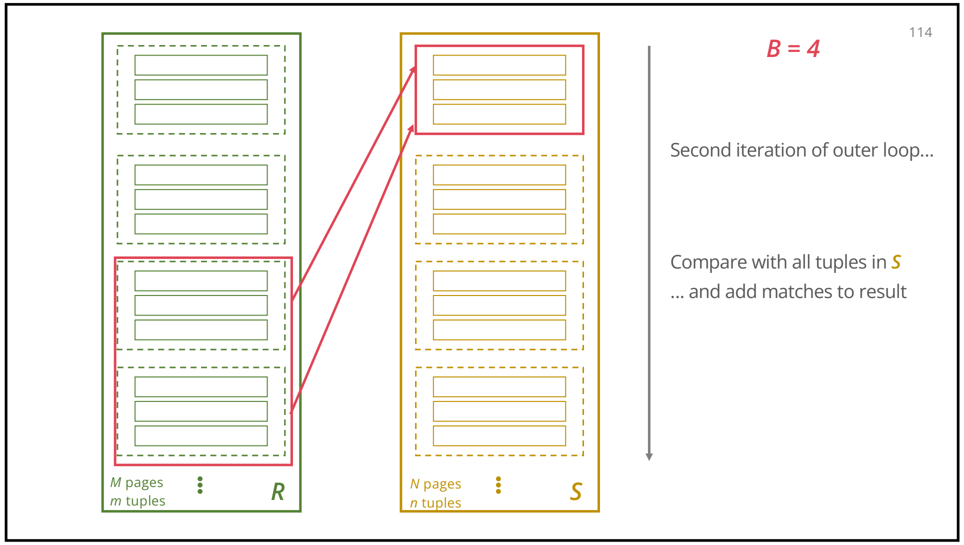


B = 4

First iteration of outer loop…

Compare with all tuples in *S*
… and add matches to result

113

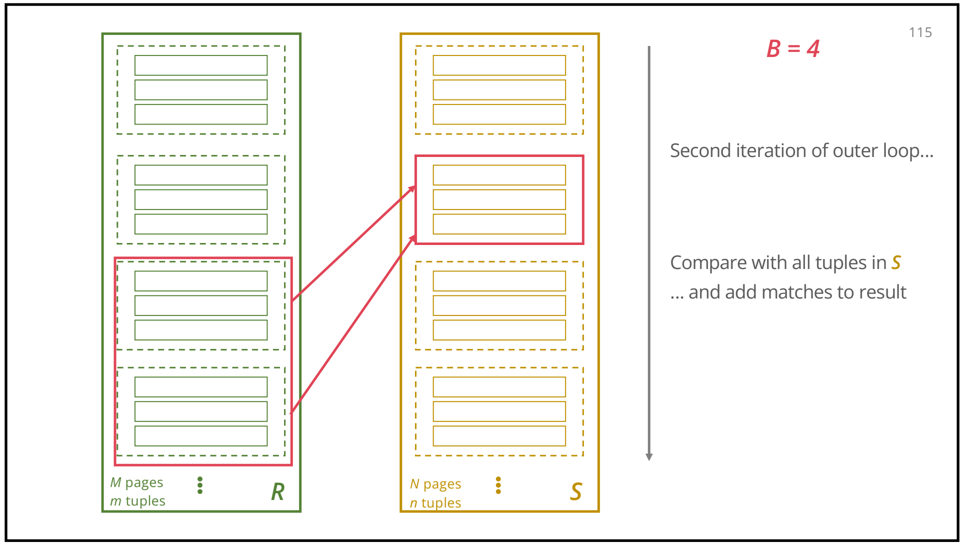


B = 4

Second iteration of outer loop…

Compare with all tuples in *S*
… and add matches to result

114

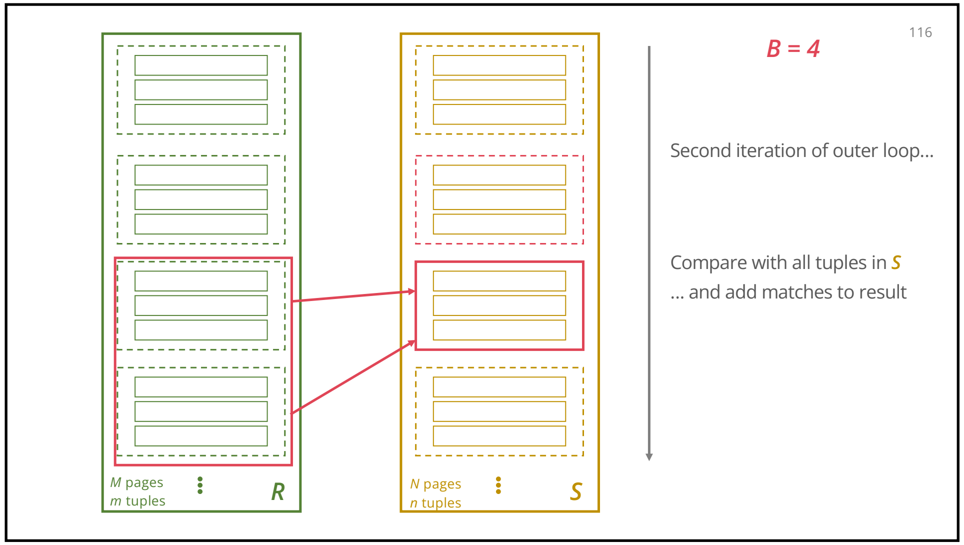


B = 4

Second iteration of outer loop…

Compare with all tuples in *S*
… and add matches to result

115

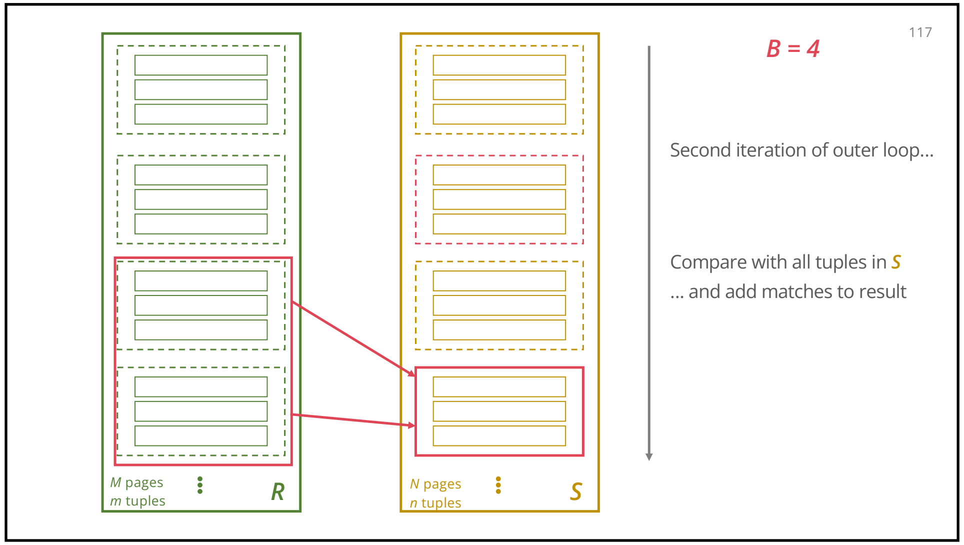


B = 4

Second iteration of outer loop…

Compare with all tuples in *S*
… and add matches to result

116

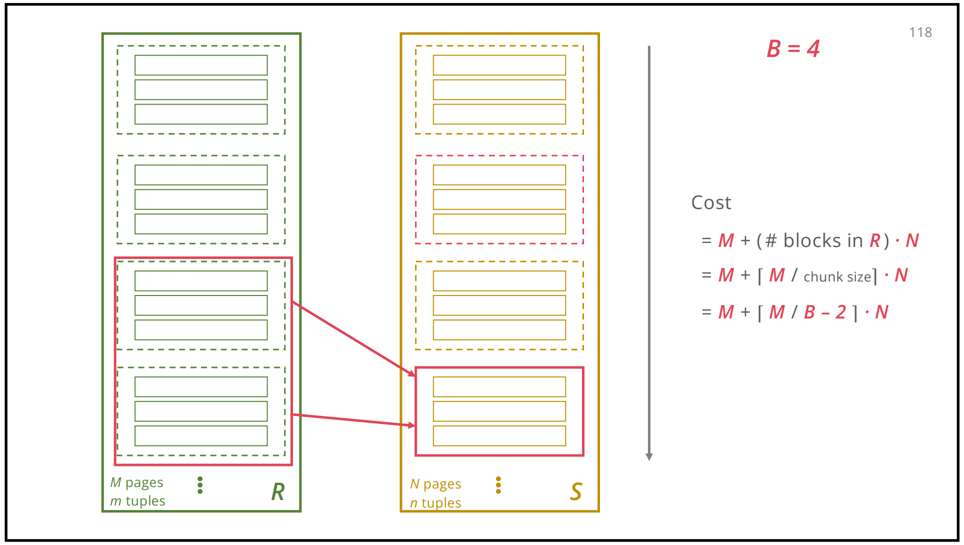


$B = 4$

Second iteration of outer loop…

Compare with all tuples in S
… and add matches to result

117



$B = 4$

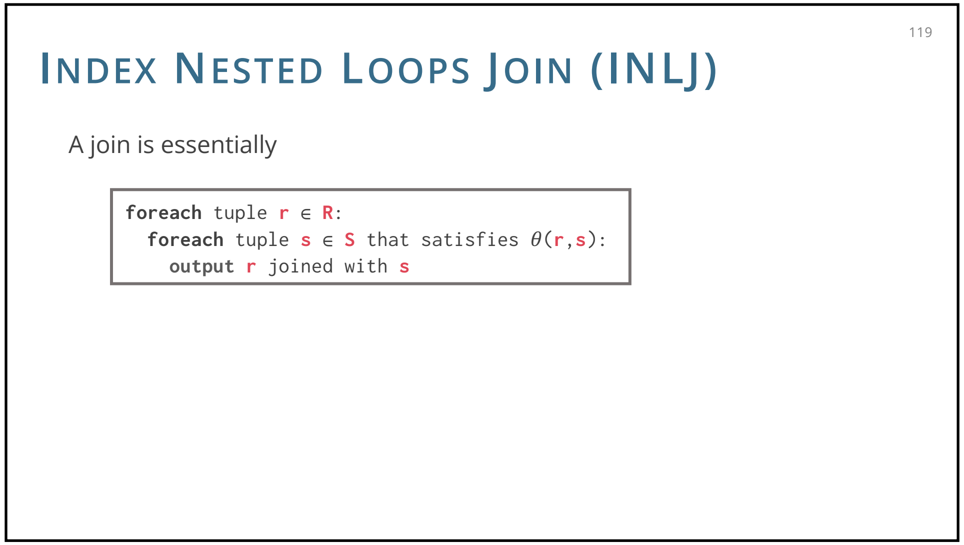
Cost

$= M + (\text{\# blocks in } R) \cdot N$

$= M + \lceil M / \text{chunk size} \rceil \cdot N$

$= M + \lceil M / B - 2 \rceil \cdot N$

118

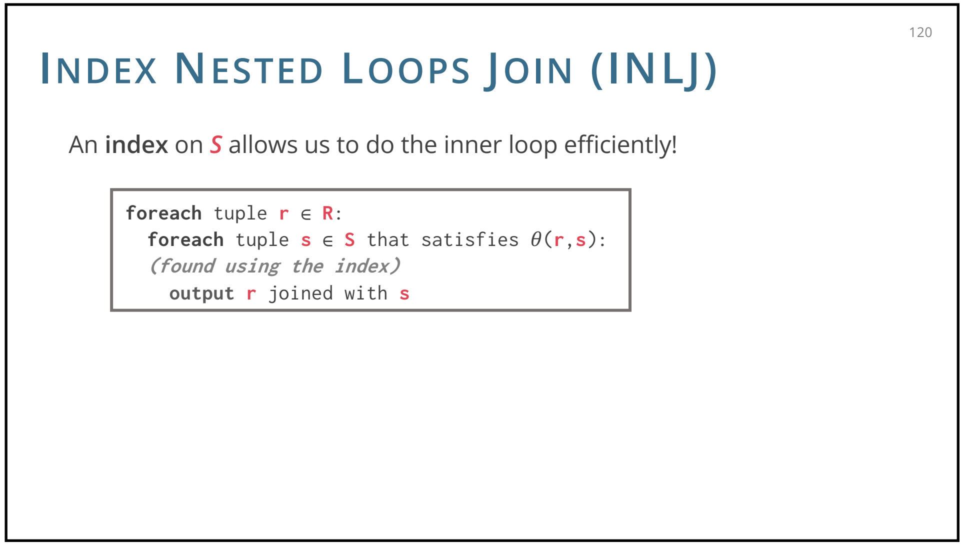


Index Nested Loops Join (INLJ)

A join is essentially

```
foreach tuple r ∈ R:
  foreach tuple s ∈ S that satisfies θ(r,s):
    output r joined with s
```

119



Index Nested Loops Join (INLJ)

An **index** on S allows us to do the inner loop efficiently!

```
foreach tuple r ∈ R:
  foreach tuple s ∈ S that satisfies θ(r,s):
  (found using the index)
    output r joined with s
```

120

Index Nested Loops Join (INLJ)

- Cost = *M* + *m* * cost to find matching ***S*** tuples
  - *M* from scanning through ***R***
- Cost to find matching ***S*** tuples via **tree index**
  - Variant A: cost to traverse root to leaf + read all the leaves with matching tuples
  - Variants B or C: cost of retrieving RIDs (similar to Variant A) + cost to fetch actual records
    - 1 I/O per **page** if clustered, 1 I/O per **matching tuple** if not

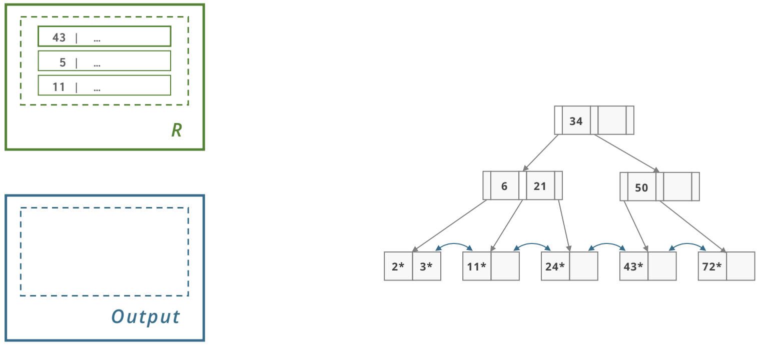
Index Nested Loops Join (INLJ)

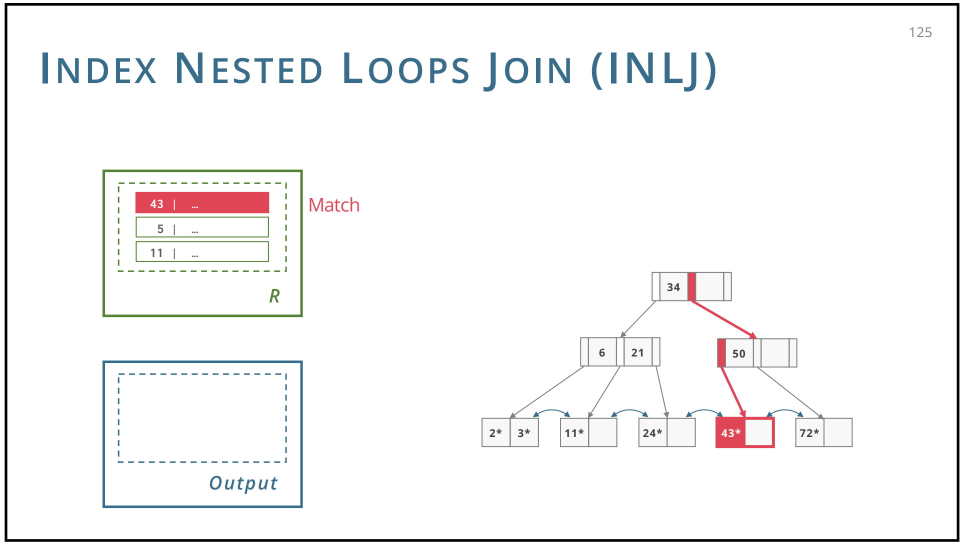
- Cost = *M* + *m* * cost to find matching ***S*** tuples
  - *M* from scanning through ***R***
- Cost to find matching ***S*** tuples via **hash index**
  - 1-2 I/Os to reach the target bucket
  - Then scan pages in that bucket
  - May stop early if search key values are unique and we found a match!

Index Nested Loops Join (INLJ)

- Cost = *M* + *m* * cost to find matching ***S*** tuples
  - *M* from scanning through ***R***
- If we have **no index**
  - Then the only way to search for matching ***S*** tuples is by scanning all of ***S*** ⇒ SNLJ
  - Cost to find matching ***S*** tuples is then *N*, giving us the formula for SNLJ cost

Index Nested Loops Join (INLJ)





125

Index Nested Loops Join (INLJ)

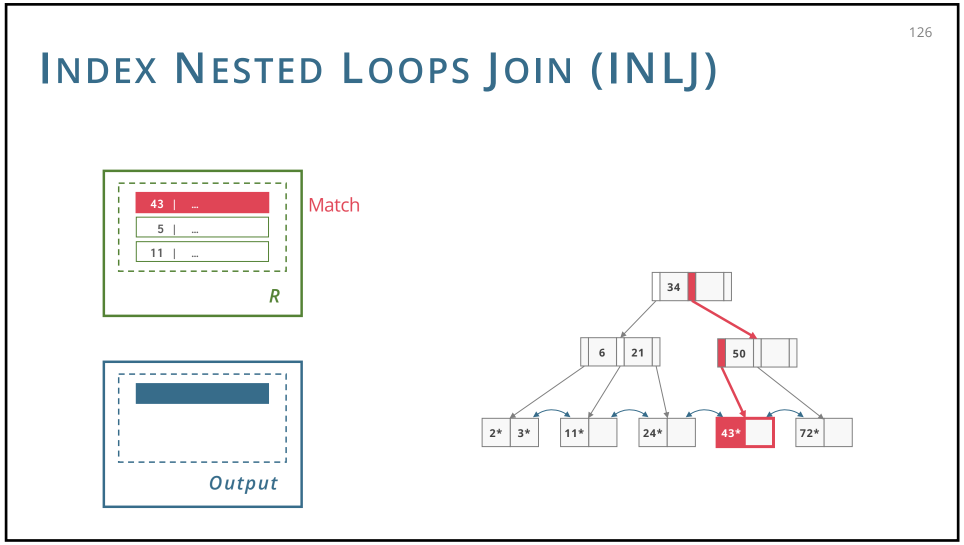
43 | ...
5 | ...
11 | ...

Match

R

Output

125



126

Index Nested Loops Join (INLJ)

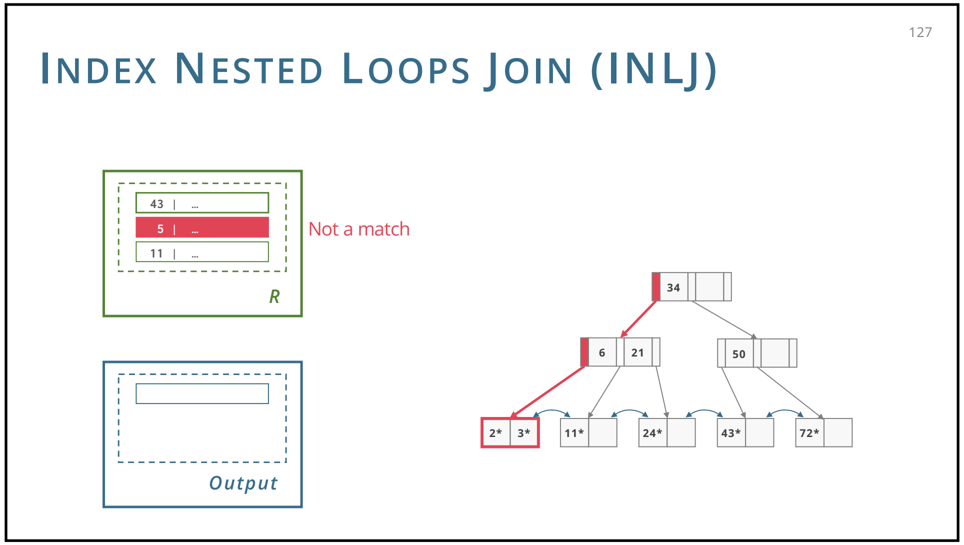
43 | ...
5 | ...
11 | ...

Match

R

Output

126



127

Index Nested Loops Join (INLJ)

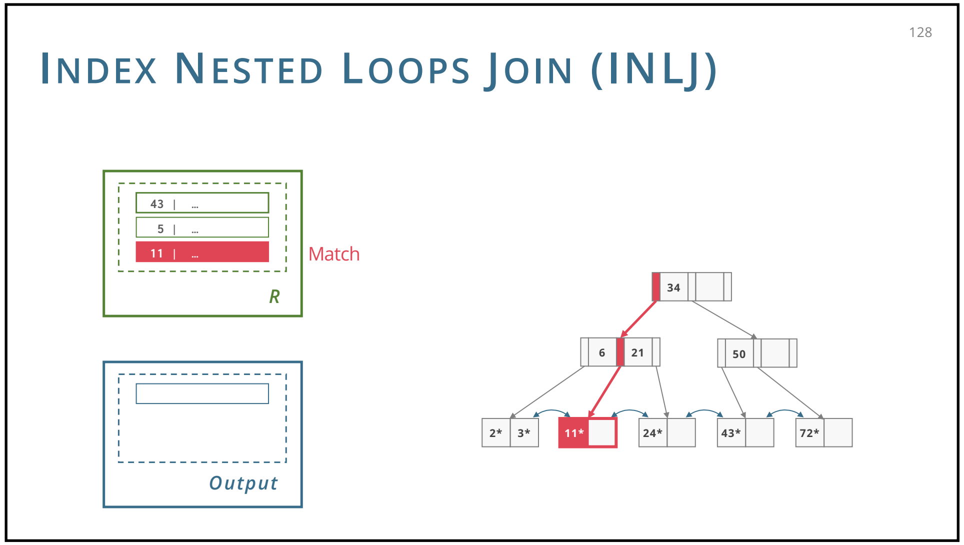
43 | ...
5 | ...
11 | ...

Not a match

R

Output

127



128

Index Nested Loops Join (INLJ)

43 | ...
5 | ...
11 | ...

Match

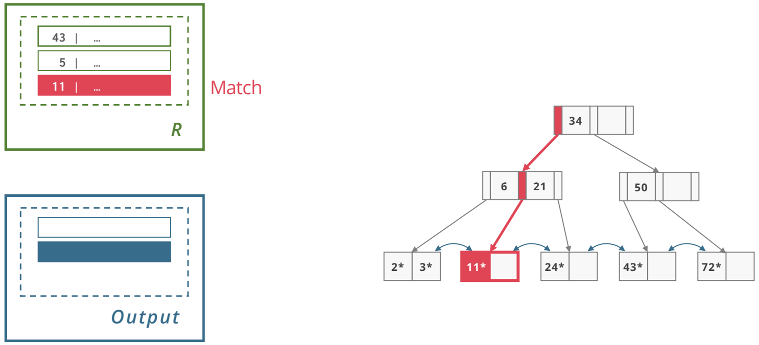
R

Output

128

129

Index Nested Loops Join (INLJ)



129

130

Sort Merge Join (SMJ)

What if we process the data a bit before we join things together?

- For example, sort both relations first! Then we can join them efficiently
- In some cases, we might even have one of the relations already sorted on the right key, and then we don't even have to spend time sorting it!

130

131

Sort Merge Join (SMJ)

First step: **sort** both *R* and *S* (with external sorting)

Second step: **merge** matching tuples from *R* and *S* together

- We do this efficiently by moving iterators over sorted *R* and sorted *S* in lockstep: move the iterator with the smaller key
  - We know that this key is smaller than all remaining key values in the other relation, so we're completely done joining that tuple!

131

132

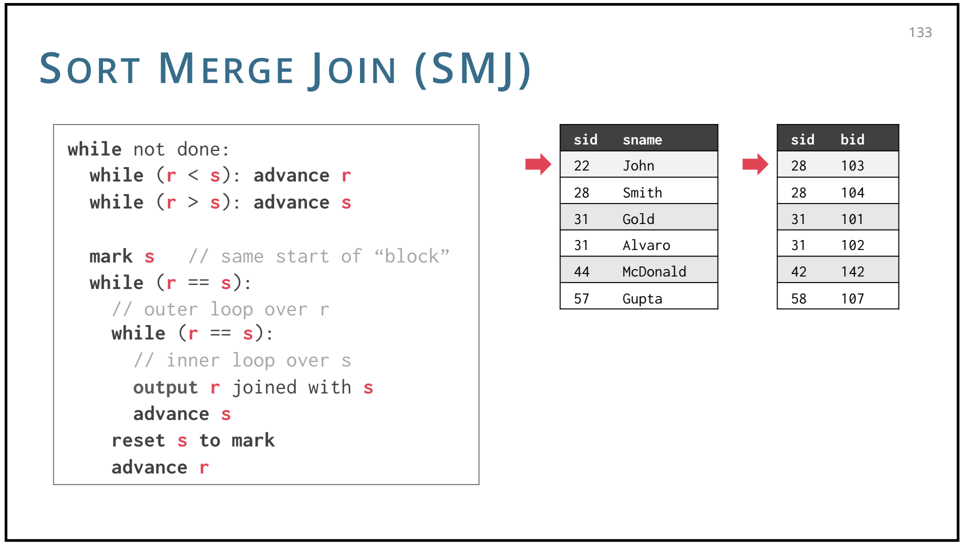
Sort Merge Join (SMJ)

First step: **sort** both *R* and *S* (with external sorting)

Second step: **merge** matching tuples from *R* and *S* together

- Need a bit more care than this:
- we might have multiple tuples in *R* matching with multiple tuples in *S*
  - **Mark** the first matching tuple in *S*
  - Match tuples with the first matching tuple in *R*,
  - Then **reset** the iterator to the mark
  - … so we can go through the tuples in *S* again for the second matching tuple in *R*

132



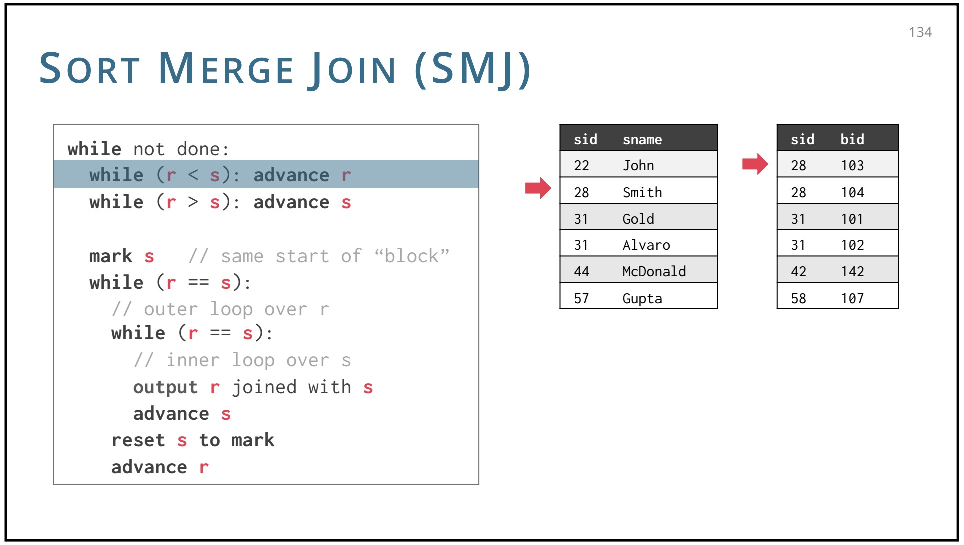
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of "block"
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

133



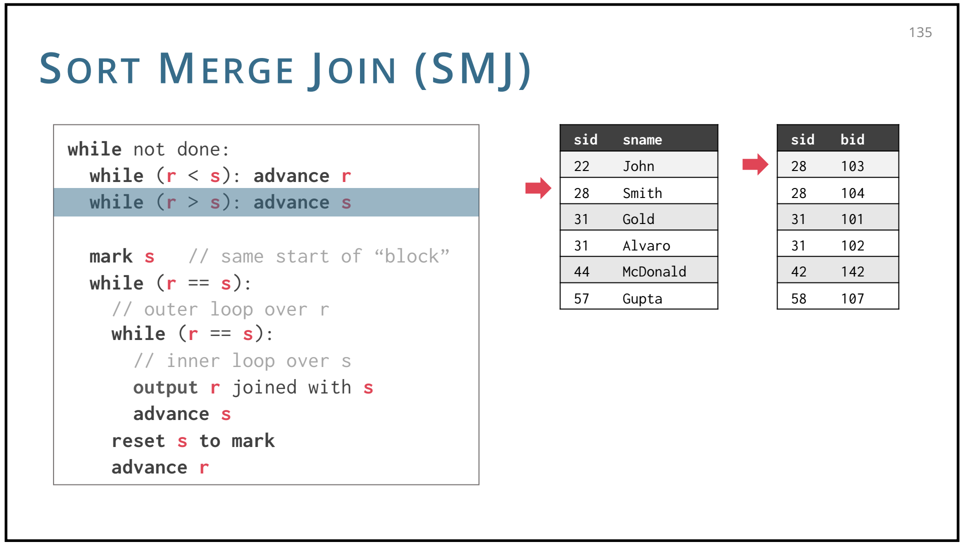
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of "block"
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

134



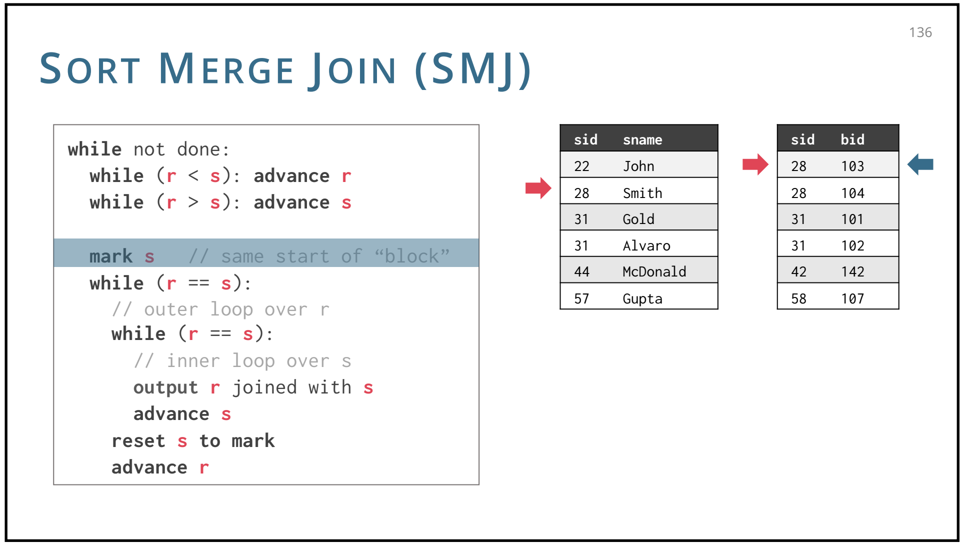
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of "block"
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

135



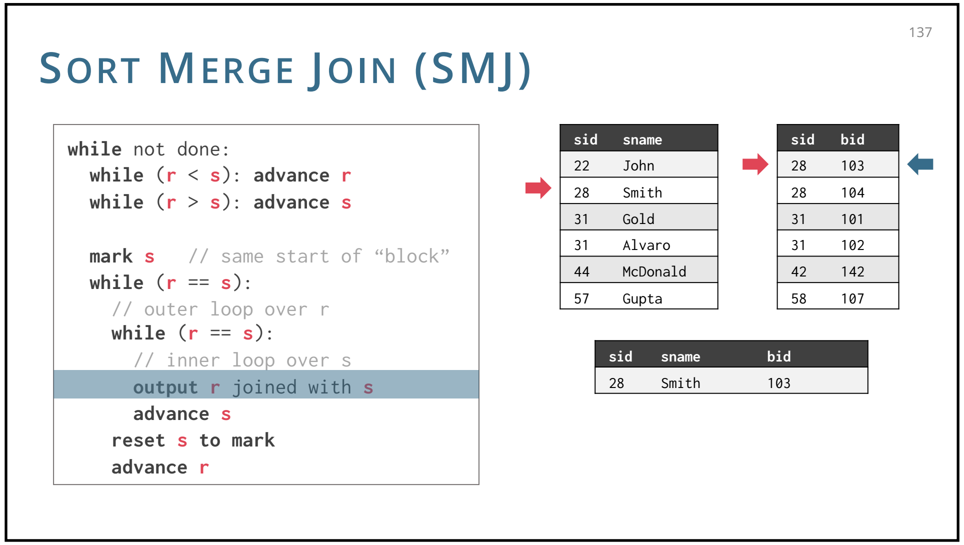
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of "block"
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

136



Sort Merge Join (SMJ)

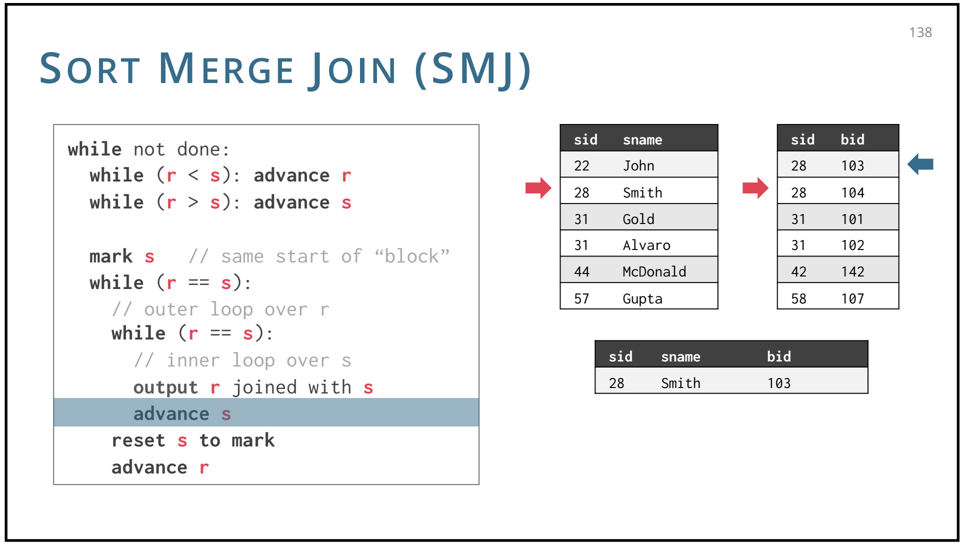
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103

137



Sort Merge Join (SMJ)

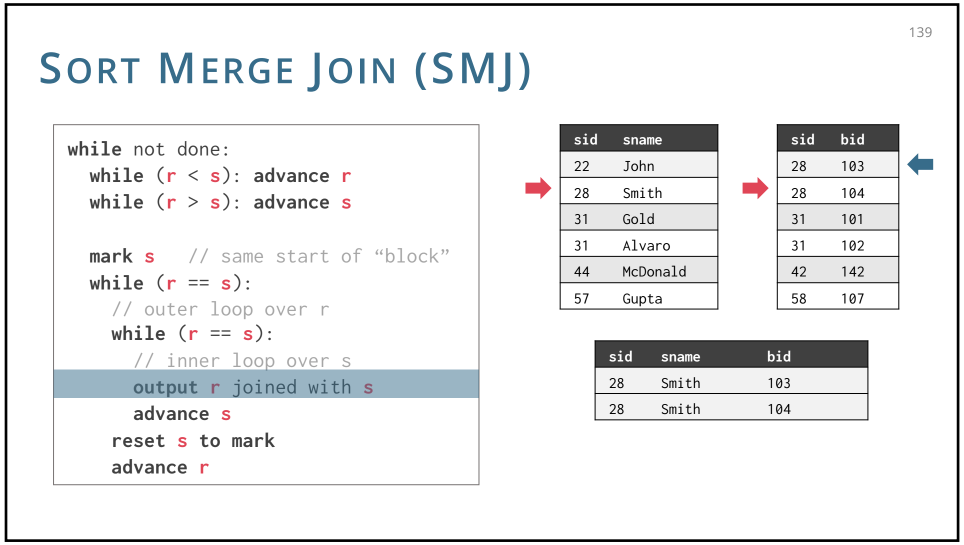
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103

138



Sort Merge Join (SMJ)

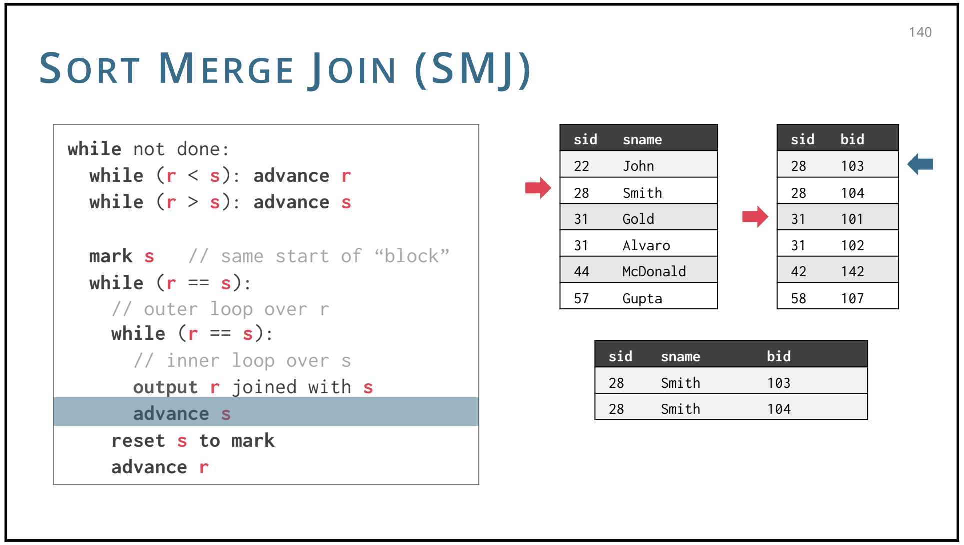
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104

139



Sort Merge Join (SMJ)

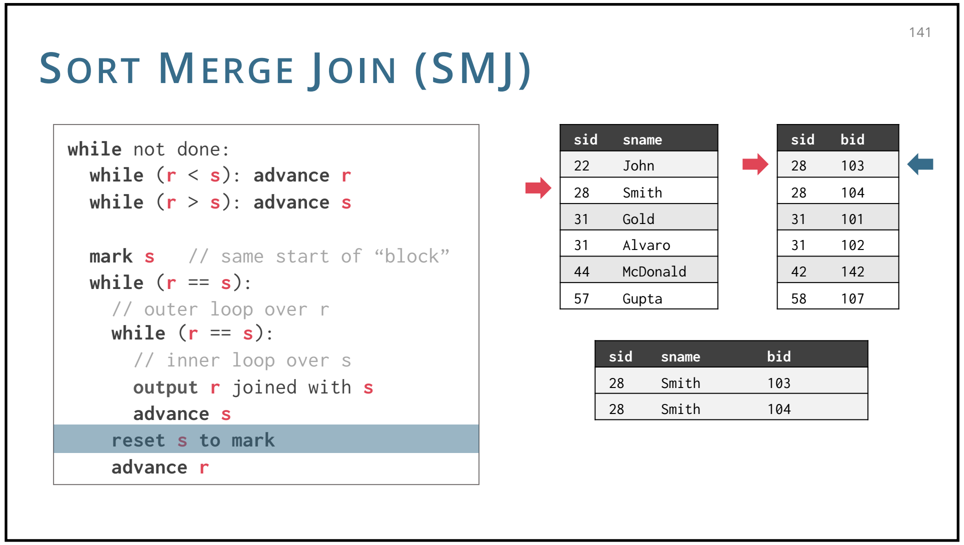
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104

140



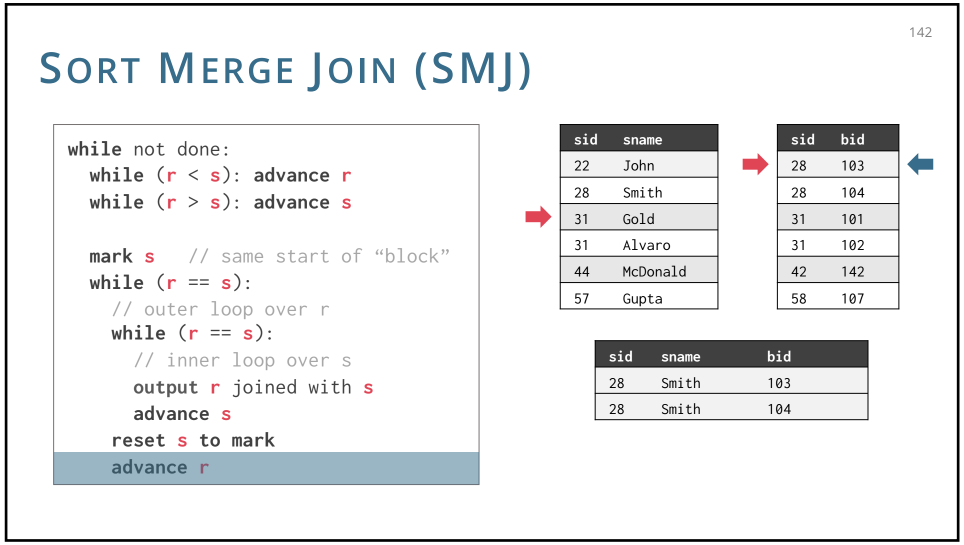
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104



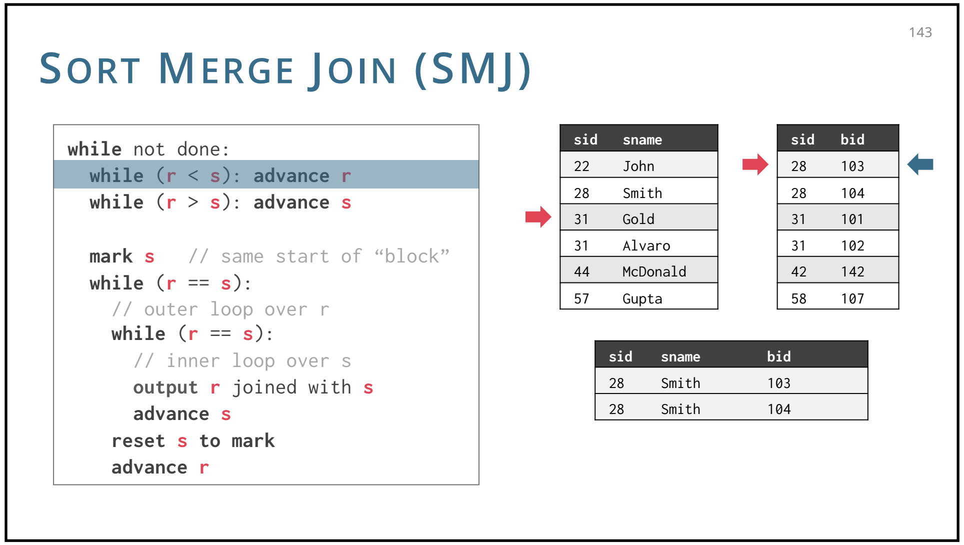
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104



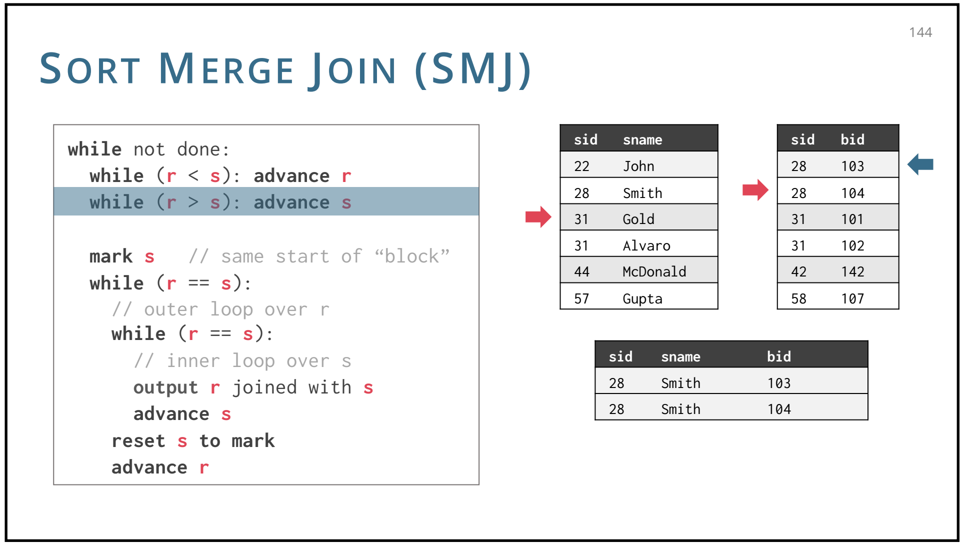
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104



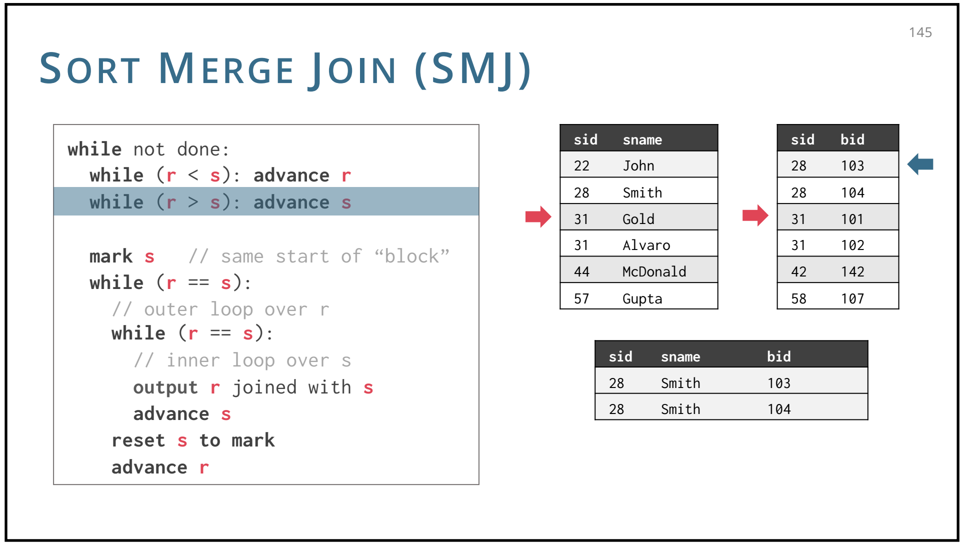
Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104



Sort Merge Join (SMJ)

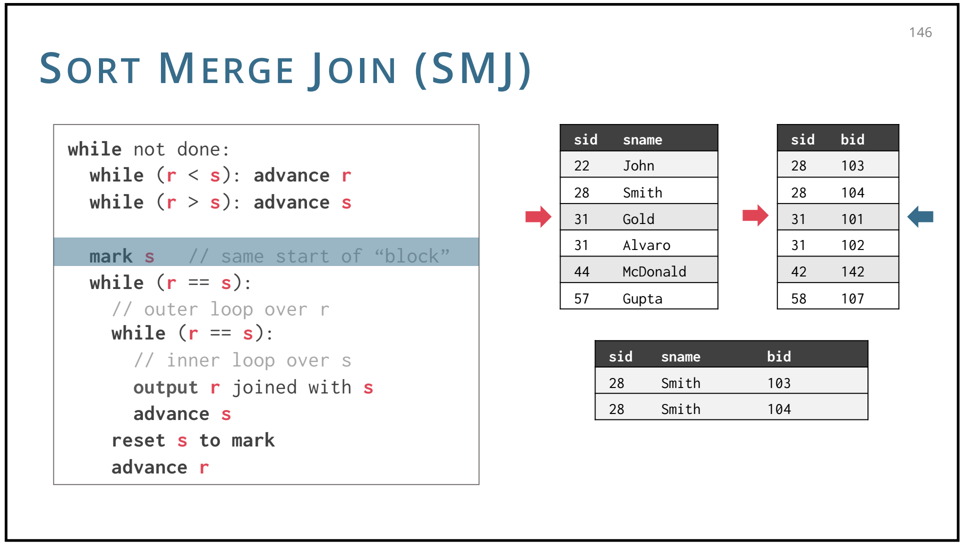
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104

145



Sort Merge Join (SMJ)

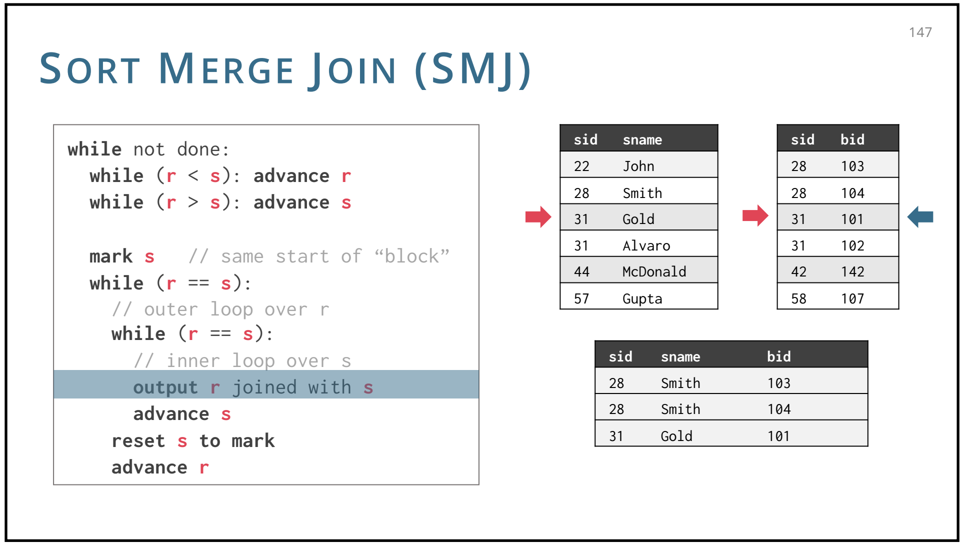
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104

146



Sort Merge Join (SMJ)

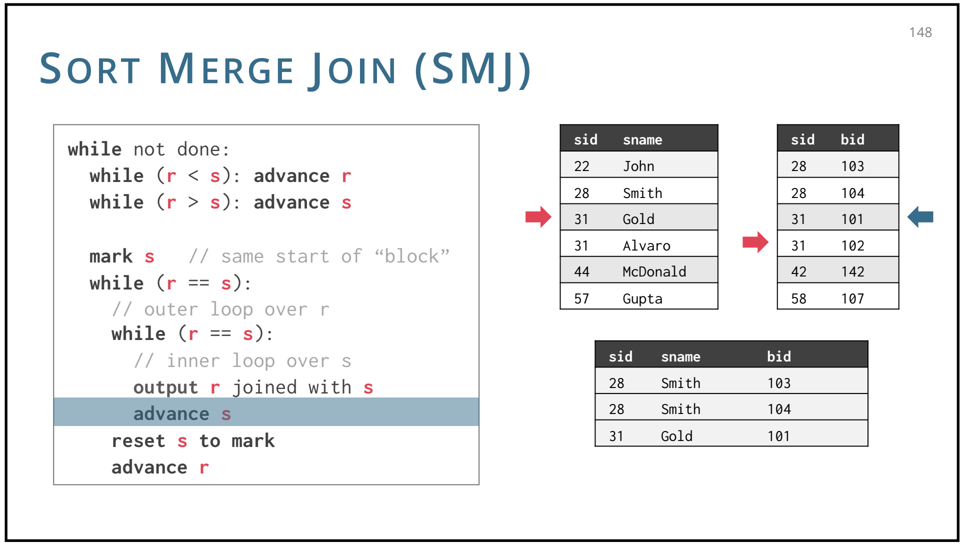
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101

147



Sort Merge Join (SMJ)

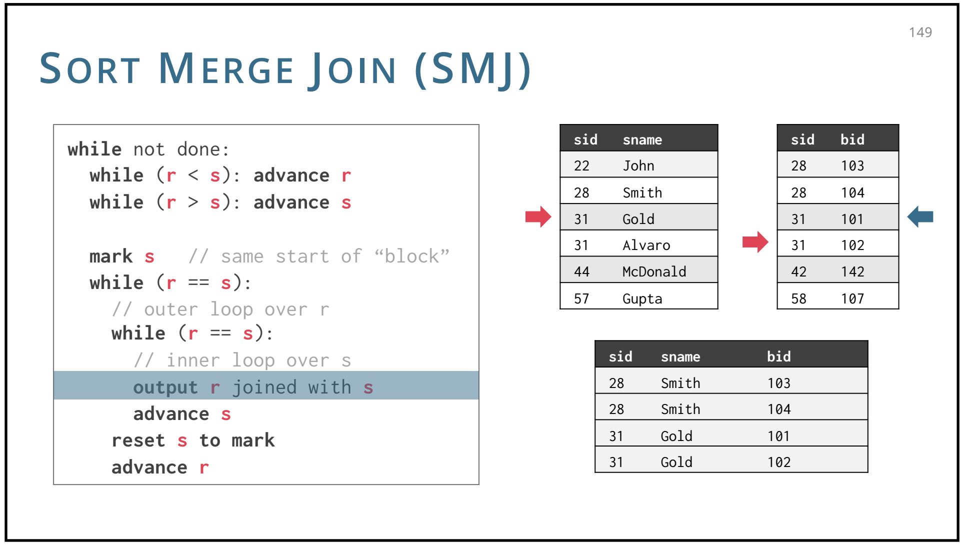
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101

148



Sort Merge Join (SMJ)

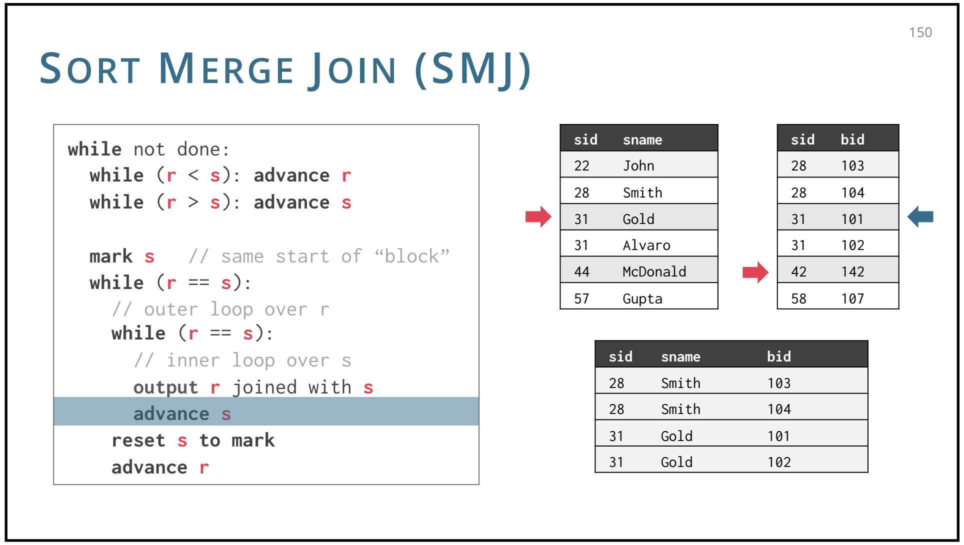
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102

149



Sort Merge Join (SMJ)

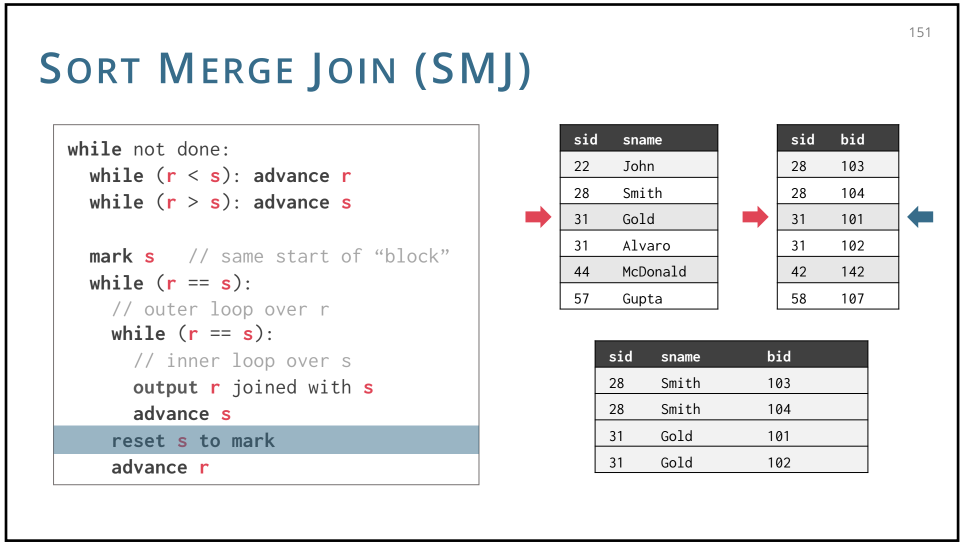
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102

150



Sort Merge Join (SMJ)

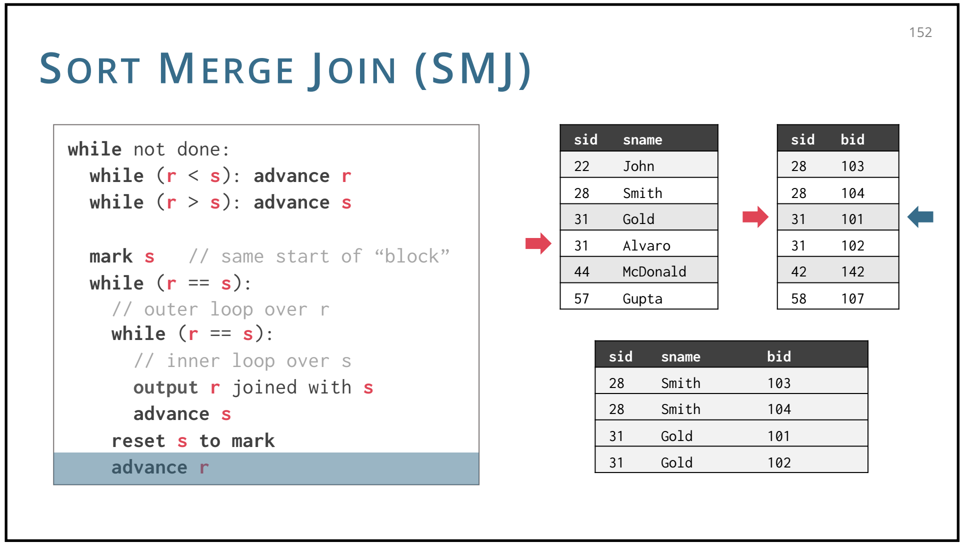
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102

151



Sort Merge Join (SMJ)

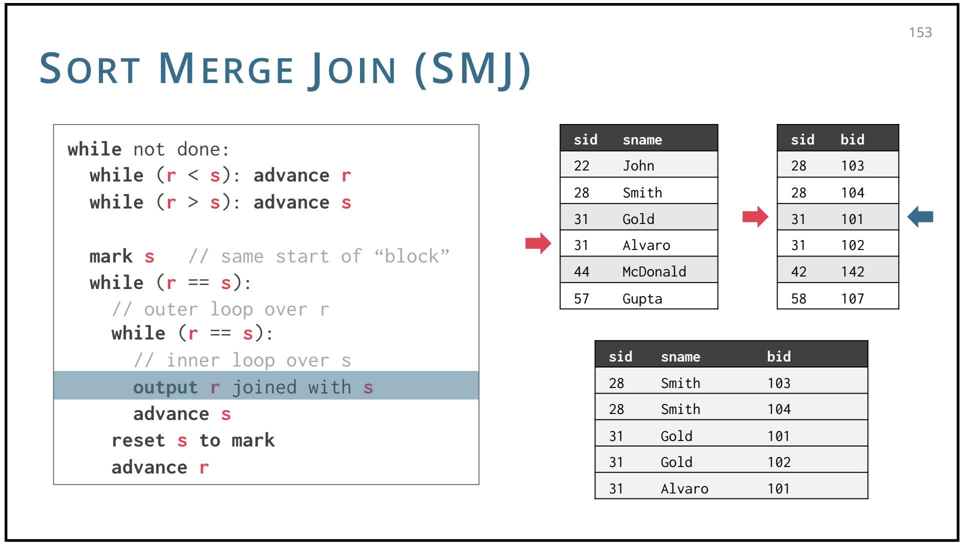
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102

152



153

Sort Merge Join (SMJ)

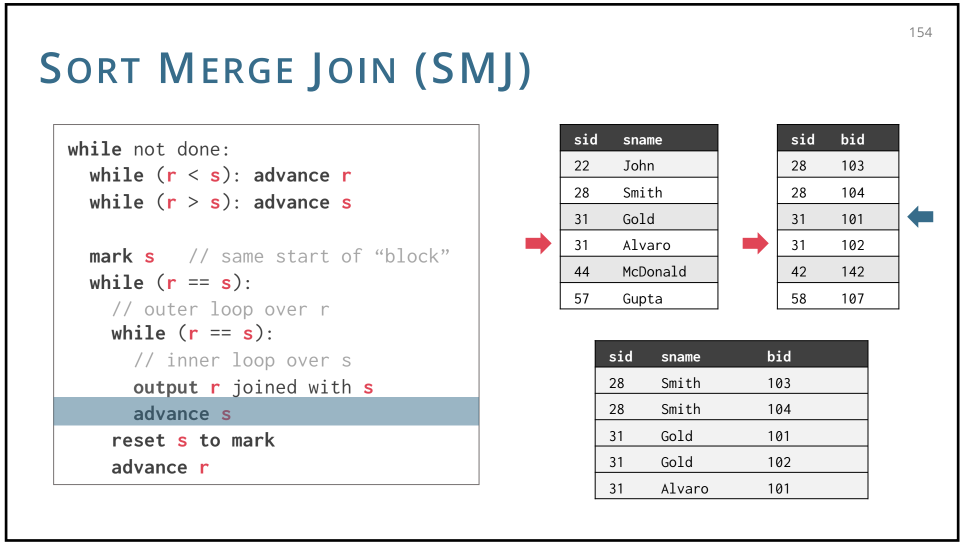
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

	sid	sname
	22	John
	28	Smith
	31	Gold
➡	31	Alvaro
	44	McDonald
	57	Gupta

	sid	bid	
	28	103	
	28	104	
➡	31	101	⬅
	31	102	
	42	142	
	58	107	

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101

153



154

Sort Merge Join (SMJ)

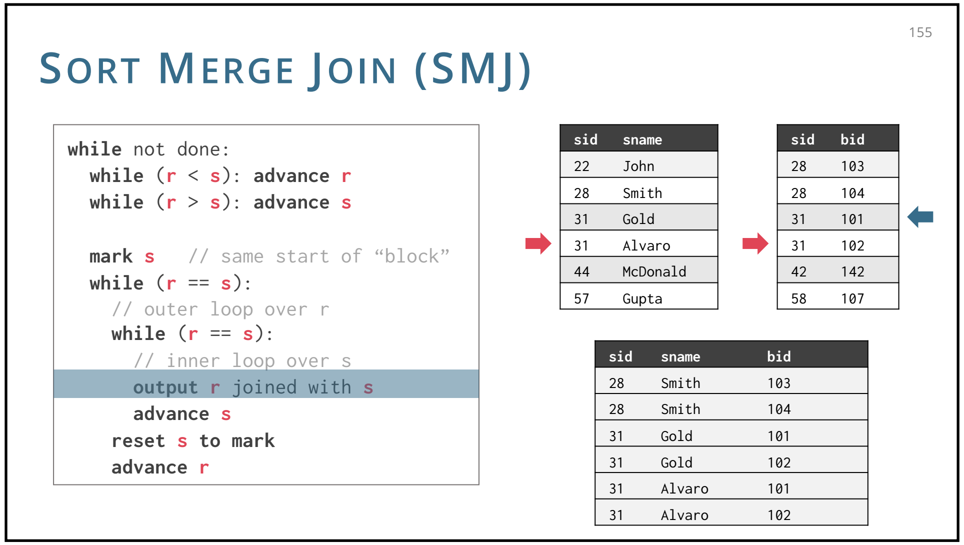
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

	sid	sname
	22	John
	28	Smith
	31	Gold
➡	31	Alvaro
	44	McDonald
	57	Gupta

	sid	bid	
	28	103	
	28	104	
	31	101	⬅
➡	31	102	
	42	142	
	58	107	

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101

154



155

Sort Merge Join (SMJ)

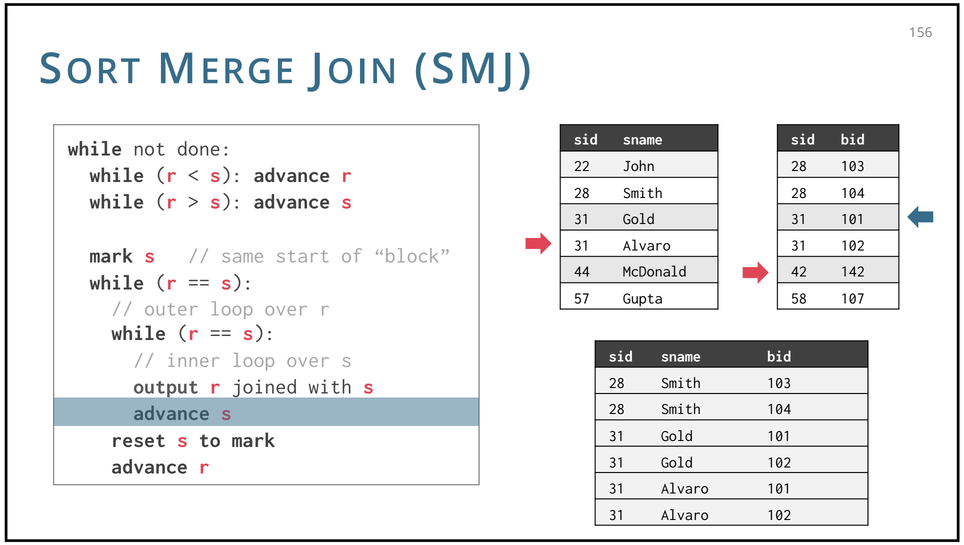
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

	sid	sname
	22	John
	28	Smith
	31	Gold
➡	31	Alvaro
	44	McDonald
	57	Gupta

	sid	bid	
	28	103	
	28	104	
	31	101	⬅
➡	31	102	
	42	142	
	58	107	

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

155



156

Sort Merge Join (SMJ)

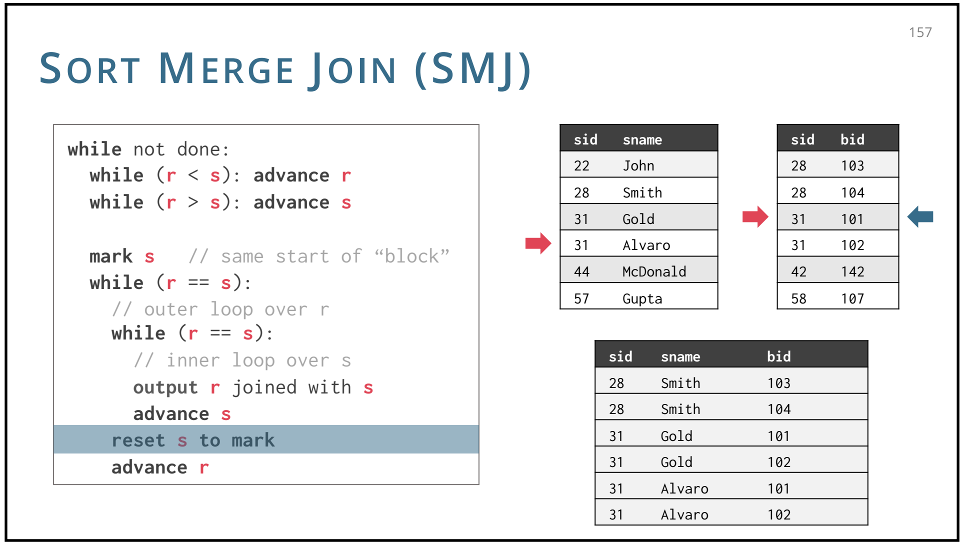
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

	sid	sname
	22	John
	28	Smith
	31	Gold
➡	31	Alvaro
	44	McDonald
	57	Gupta

	sid	bid	
	28	103	
	28	104	
	31	101	⬅
	31	102	
➡	42	142	
	58	107	

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

156



Sort Merge Join (SMJ)

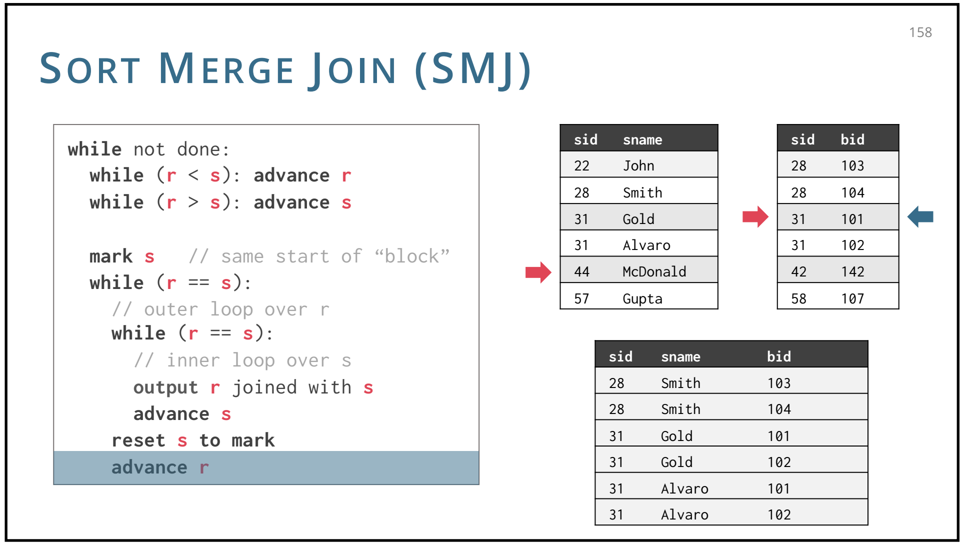
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

157



Sort Merge Join (SMJ)

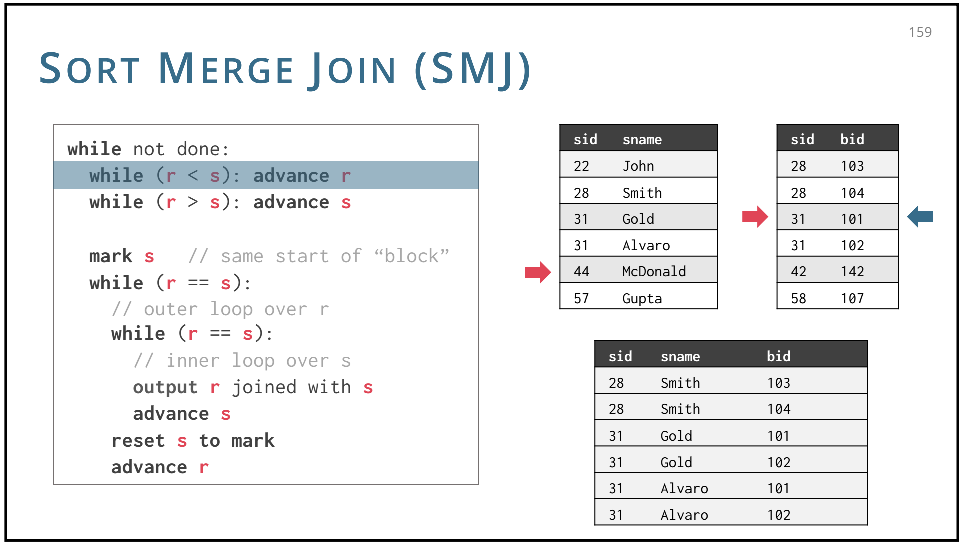
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

158



Sort Merge Join (SMJ)

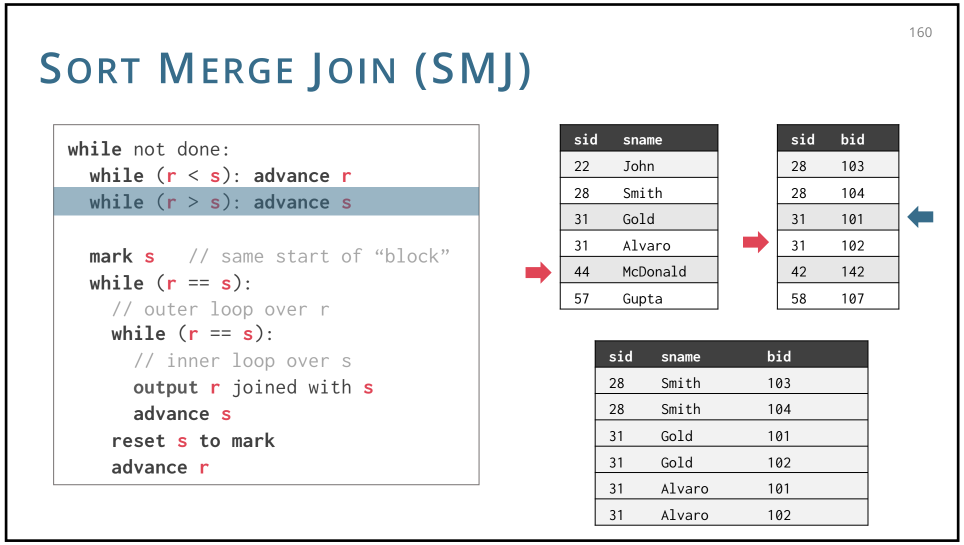
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

159



Sort Merge Join (SMJ)

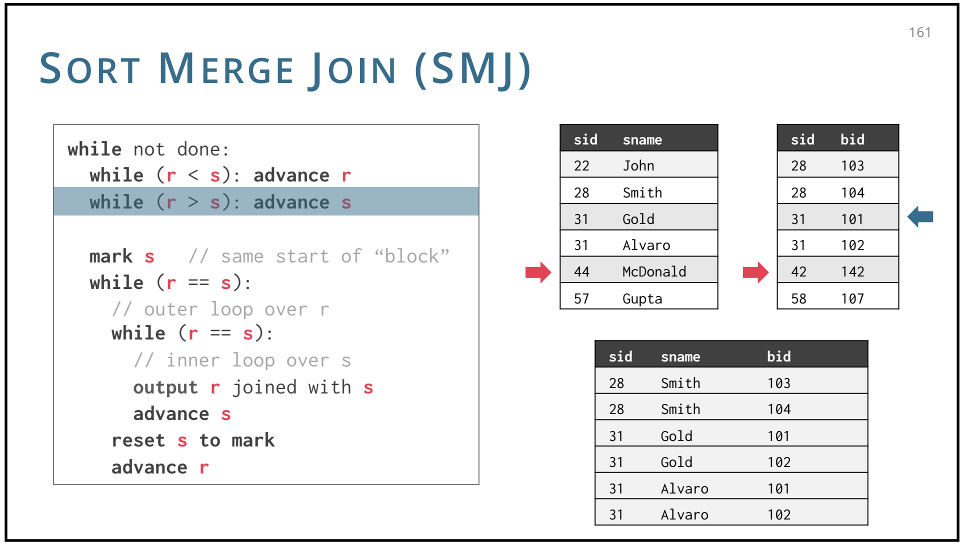
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

160



Sort Merge Join (SMJ)

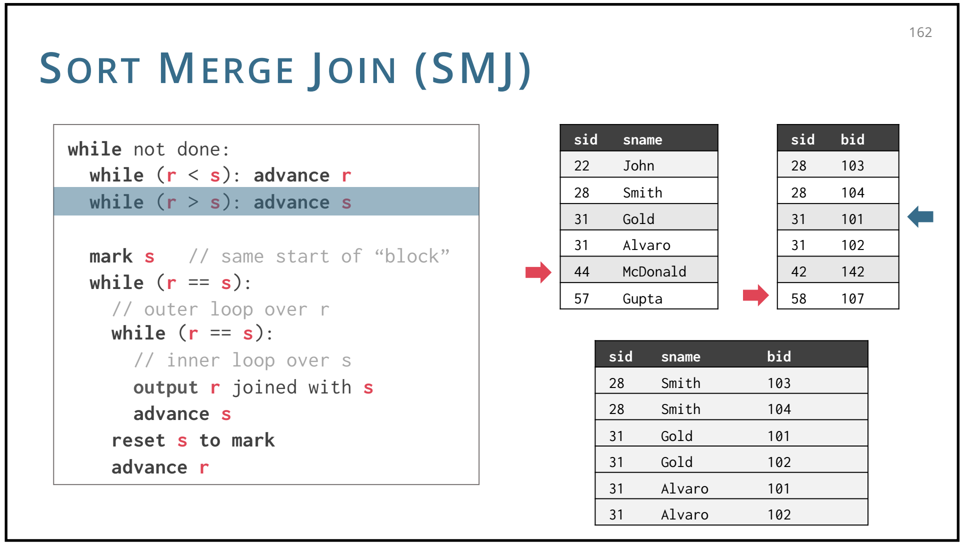
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

161



Sort Merge Join (SMJ)

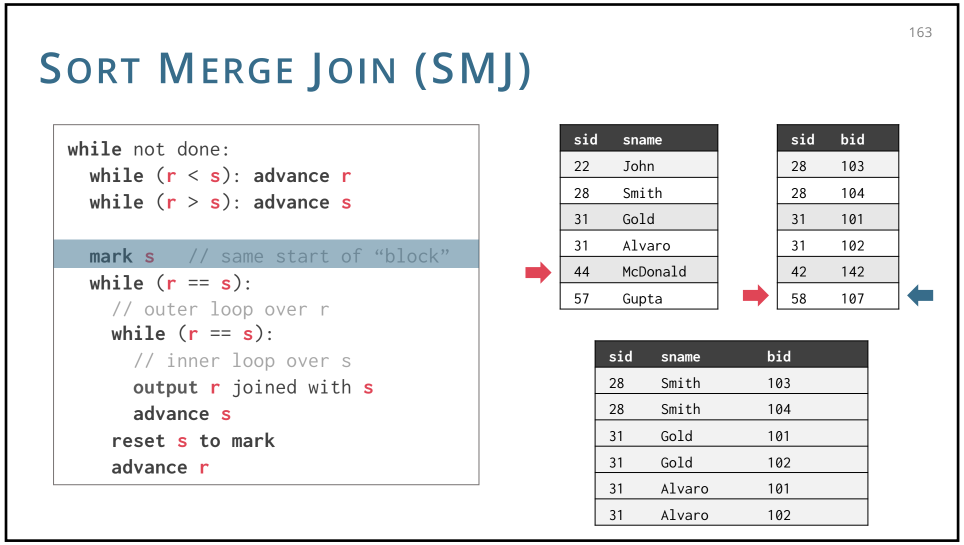
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

162



Sort Merge Join (SMJ)

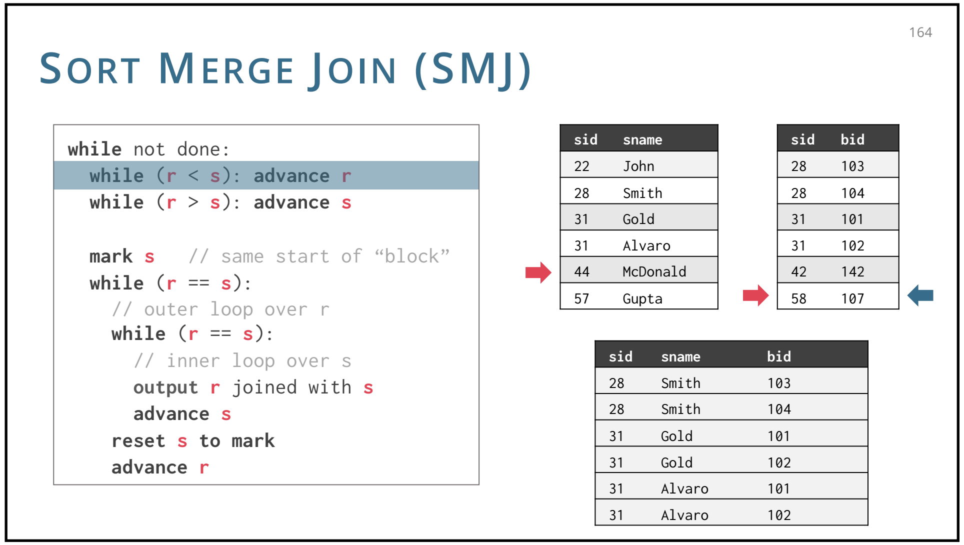
```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

163



Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s    // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

164

Sort Merge Join (SMJ)

```
while not done:
  while (r < s): advance r
  while (r > s): advance s

  mark s   // same start of “block”
  while (r == s):
    // outer loop over r
    while (r == s):
      // inner loop over s
      output r joined with s
      advance s
    reset s to mark
    advance r
```

sid	sname
22	John
28	Smith
31	Gold
31	Alvaro
44	McDonald
57	Gupta

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	Smith	103
28	Smith	104
31	Gold	101
31	Gold	102
31	Alvaro	101
31	Alvaro	102

Sort Merge Join (SMJ)

Total cost:

- Cost of sorting *R*
- Cost of sorting *S*
- Cost of merging: *M* + *N*
  - Only one pass (if we assume there aren’t a lot of duplicates)

Sort Merge Join (SMJ)

We can be sometimes smarter about SMJ

Observation:

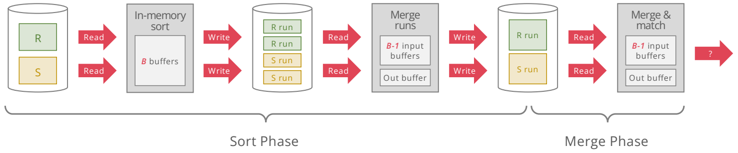
We make one pass through each sorted relation (assuming no duplicate values in *R*)

⇒ We do not need the sorted relations to be materialized!

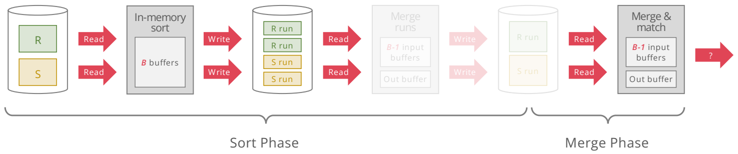
Optimisation:

In the final merge pass of sorting both relations, instead of writing the sorted relations to disk, we can stream them into the second part of SMJ!

Sort Merge Join (SMJ)



Sort Merge Join (SMJ)



We have to be able to fit the input buffers of the last merge pass of sorting ***R*** and sorting ***S*** in memory, as well as have one output buffer for joined tuples

- Need: (# runs in last merge pass for ***R***) + (# runs in last merge pass for ***S***) ≤ ***B – 1***
- Reduces I/O cost by ***2 · (M + N)***!

Grace Hash Join

Similar idea as SMJ, but let's build some hash tables instead

Two passes: **partition** the data, then **build** a hash table and **probe** it

- Partition ***R*** and ***S*** into ***B - 1*** partitions (like in external hashing) using same hash function
- All the tuples in ***R*** matching a tuple in ***S*** must be in the same partition
  - ⇒ We can consider each partition independently

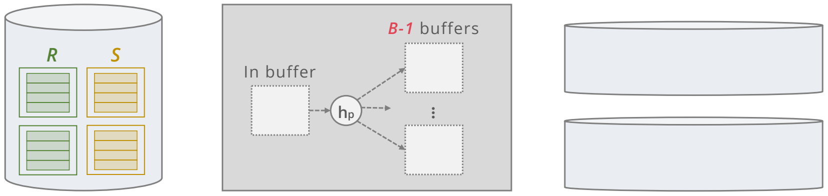
Grace Hash Join

Similar idea as SMJ, but let's build some hash tables instead

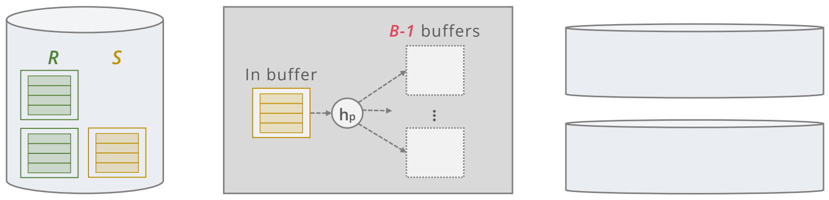
Two passes: **partition** the data, then **build** a hash table and **probe** it

- Then, build an in-memory hash table for a partition of ***R***
- We can use this in-memory hash table to find all the tuples in ***R*** that match a tuple in ***S***
  - Stream in tuples of ***S***, probe the hash table, output matching tuples

Grace Hash Join: Partition

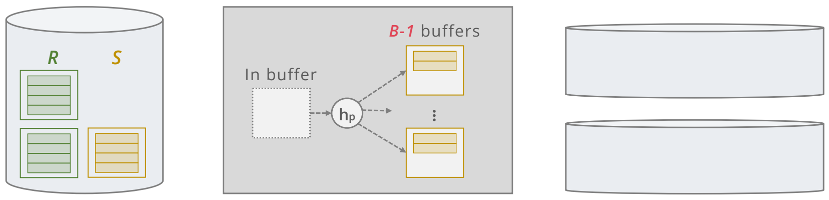


Grace Hash Join: Partition



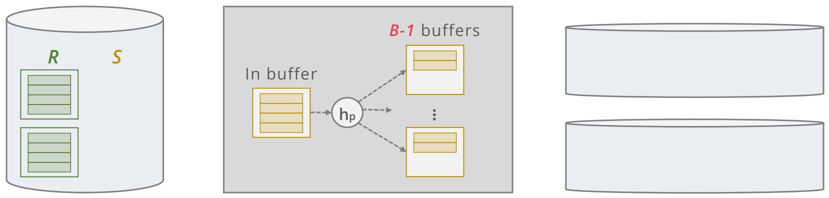
173

Grace Hash Join: Partition



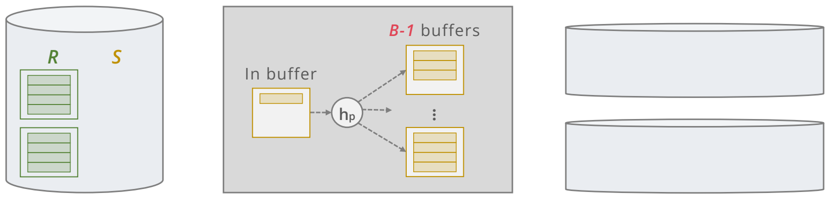
174

Grace Hash Join: Partition



175

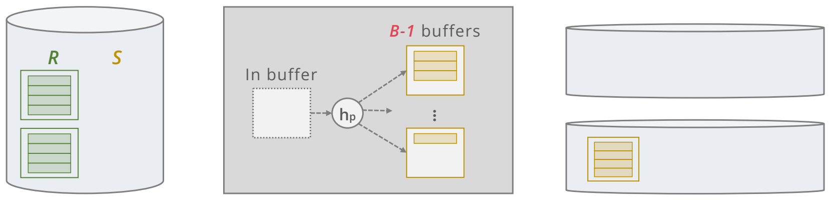
Grace Hash Join: Partition



176

177

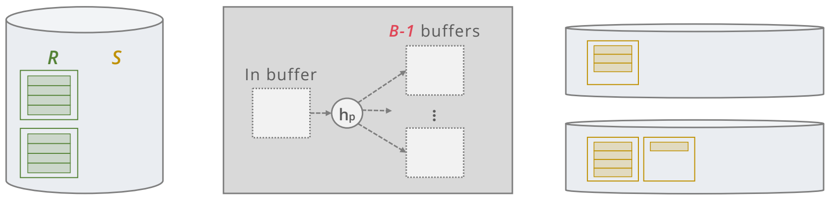
Grace Hash Join: Partition



177

178

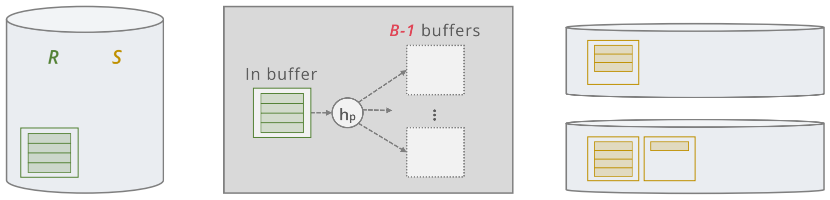
Grace Hash Join: Partition



178

179

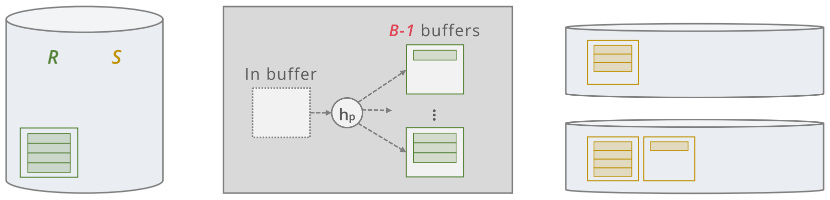
Grace Hash Join: Partition



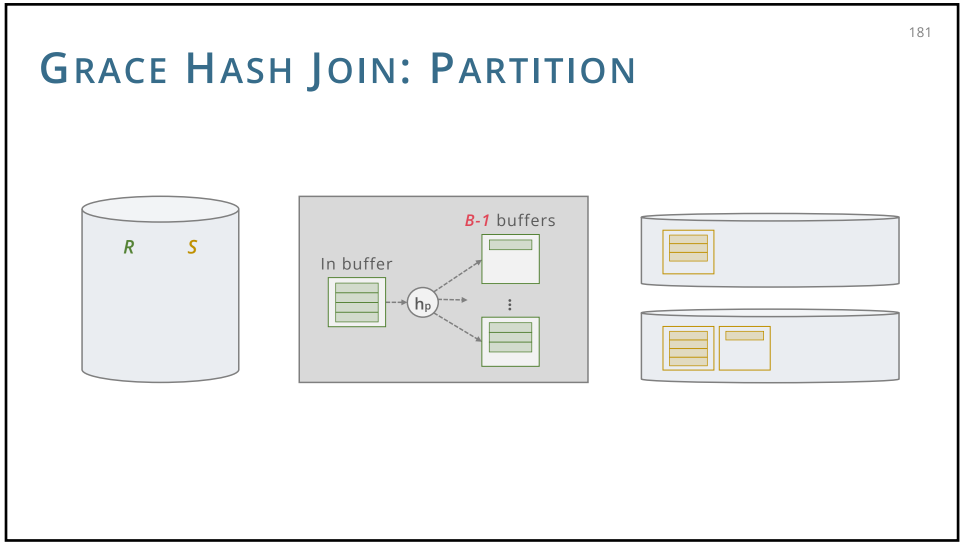
179

180

Grace Hash Join: Partition



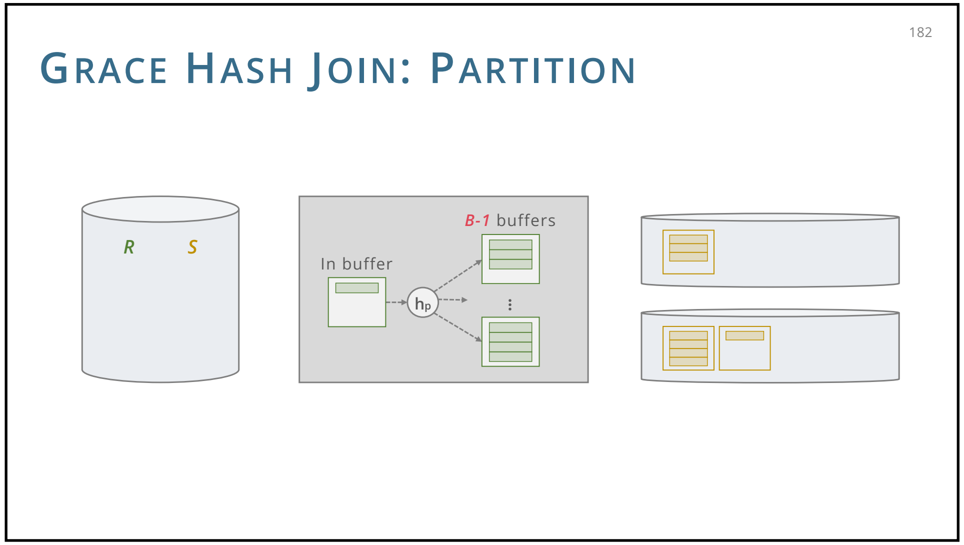
180



181

Grace Hash Join: Partition

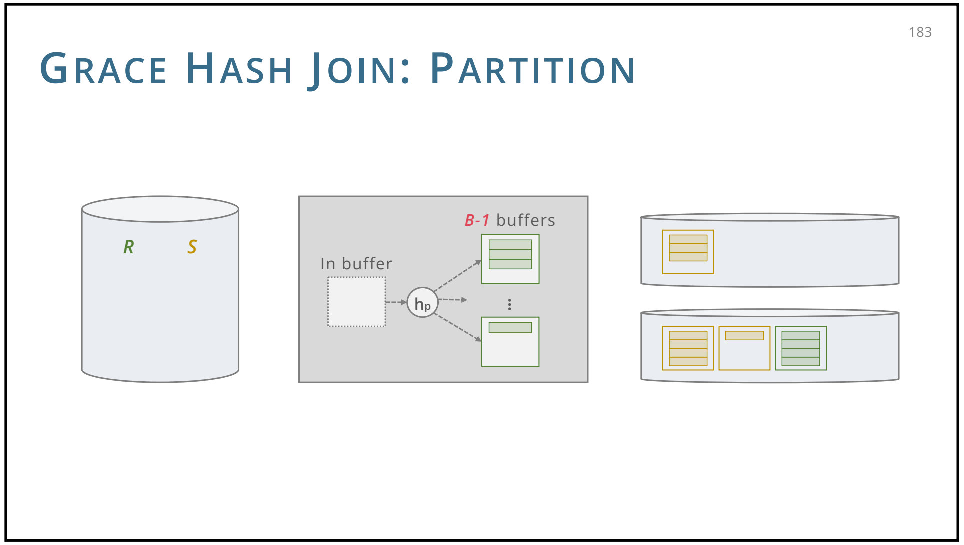
181



182

Grace Hash Join: Partition

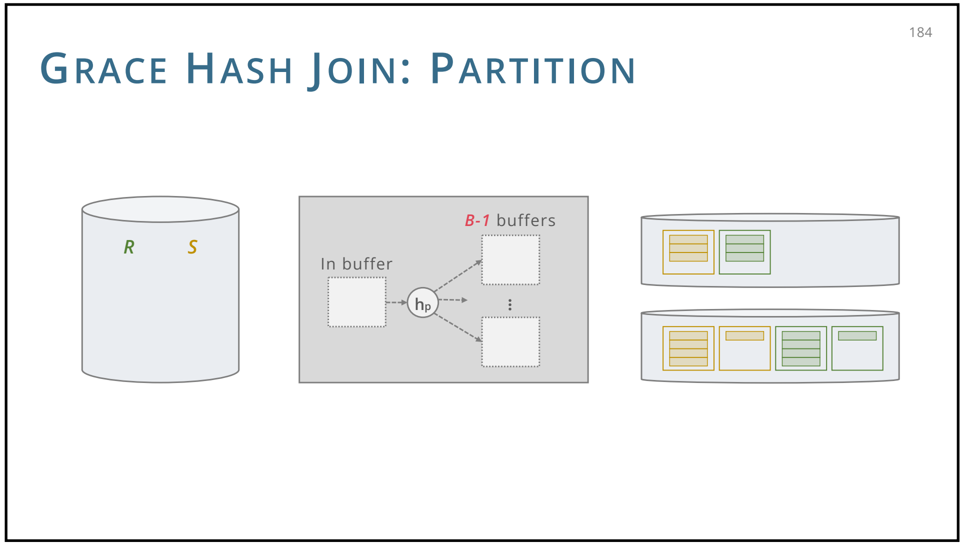
182



183

Grace Hash Join: Partition

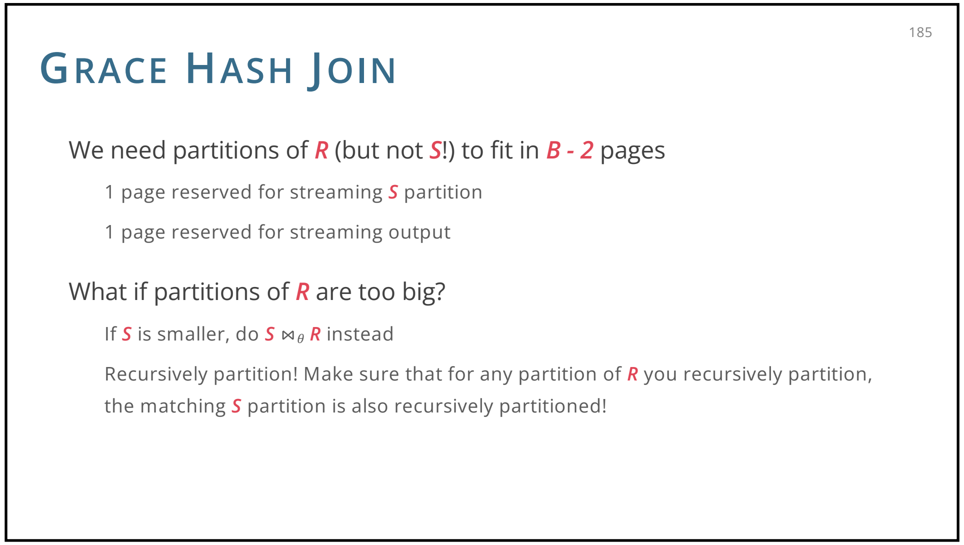
183



184

Grace Hash Join: Partition

184



185

GRACE HASH JOIN

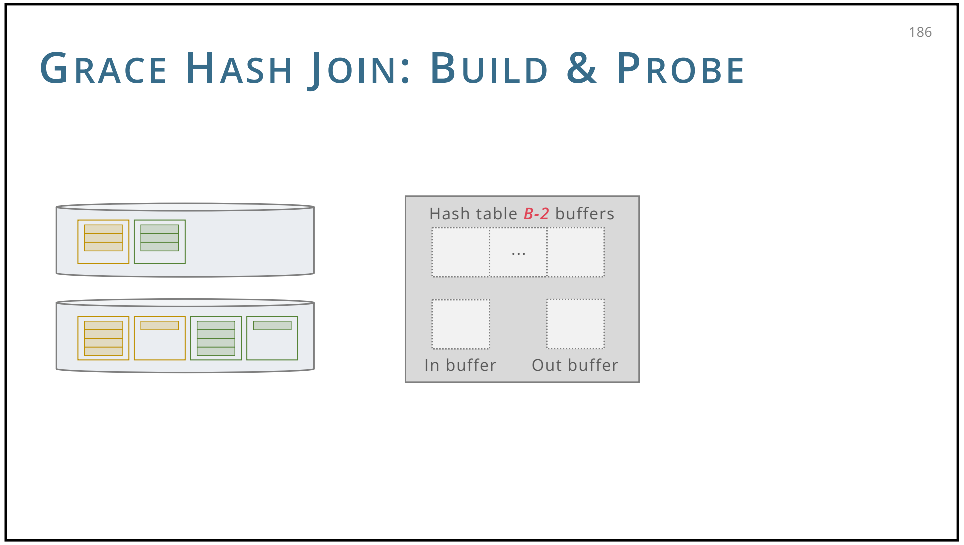
We need partitions of *R* (but not *S*!) to fit in *B - 2* pages

- 1 page reserved for streaming *S* partition
- 1 page reserved for streaming output

What if partitions of *R* are too big?

- If *S* is smaller, do $S \bowtie_\theta R$ instead
- Recursively partition! Make sure that for any partition of *R* you recursively partition, the matching *S* partition is also recursively partitioned!

185



186

GRACE HASH JOIN: BUILD & PROBE

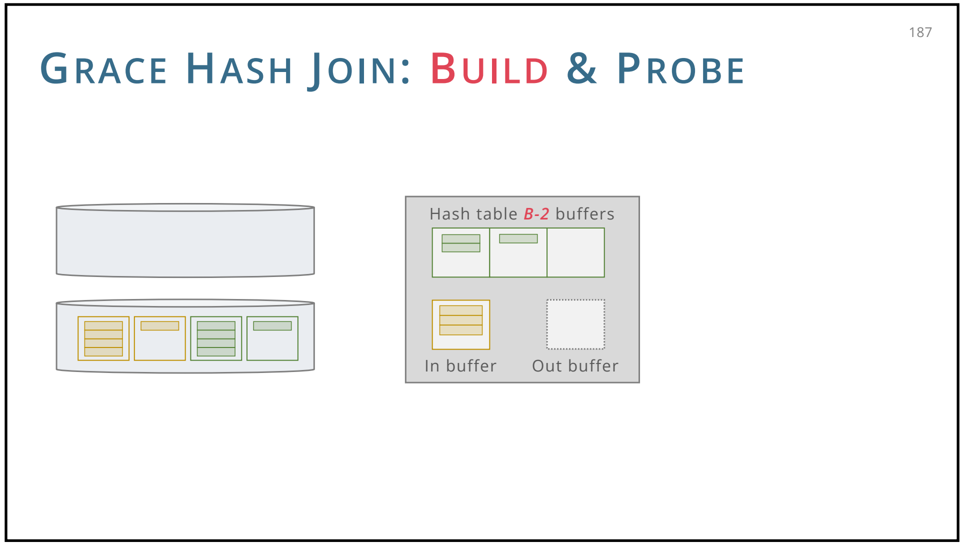
Hash table *B-2* buffers

...

In buffer

Out buffer

186



187

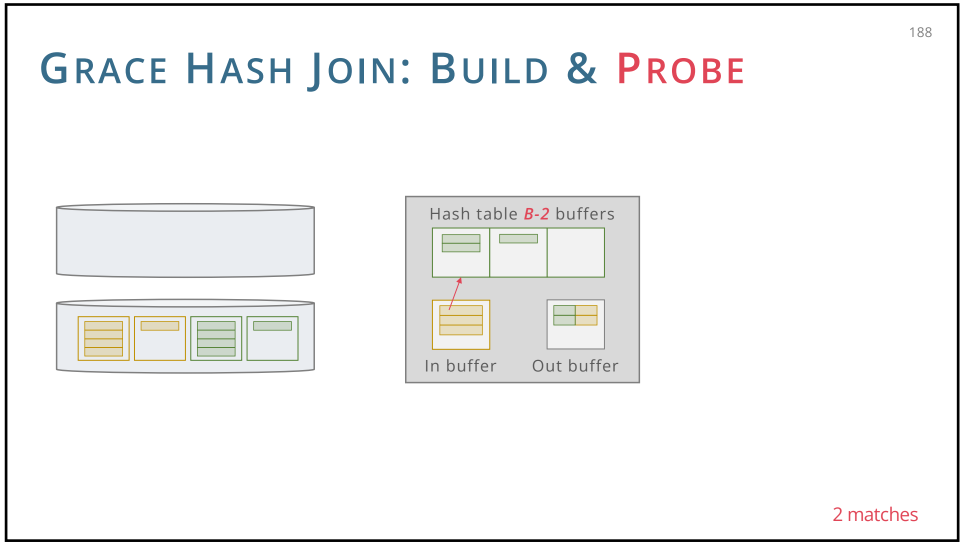
GRACE HASH JOIN: BUILD & PROBE

Hash table *B-2* buffers

In buffer

Out buffer

187



188

GRACE HASH JOIN: BUILD & PROBE

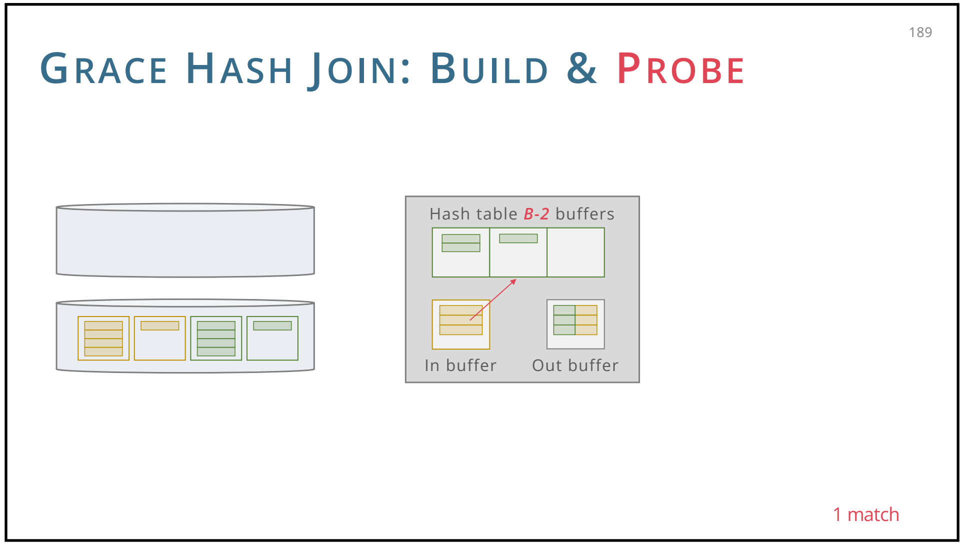
Hash table *B-2* buffers

In buffer

Out buffer

2 matches

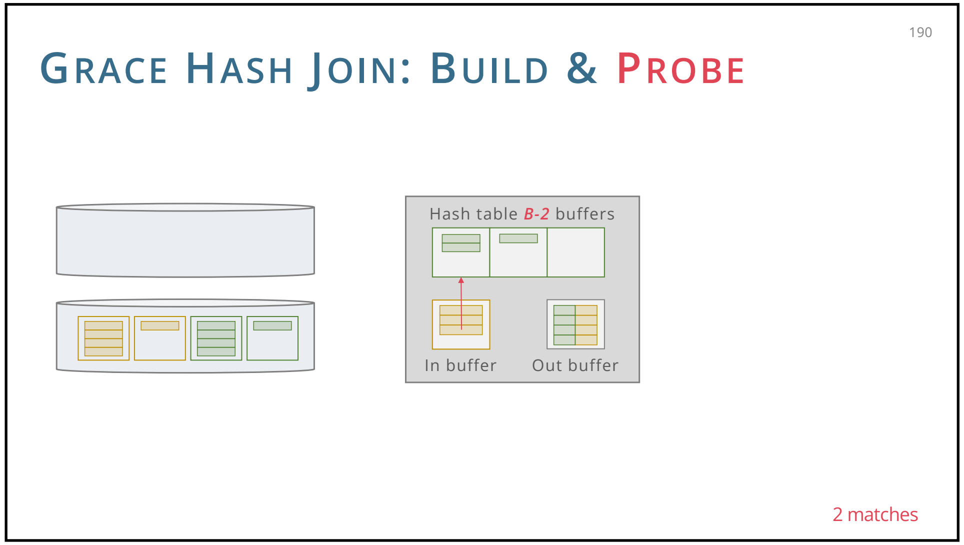
188



Grace Hash Join: Build & Probe

1 match

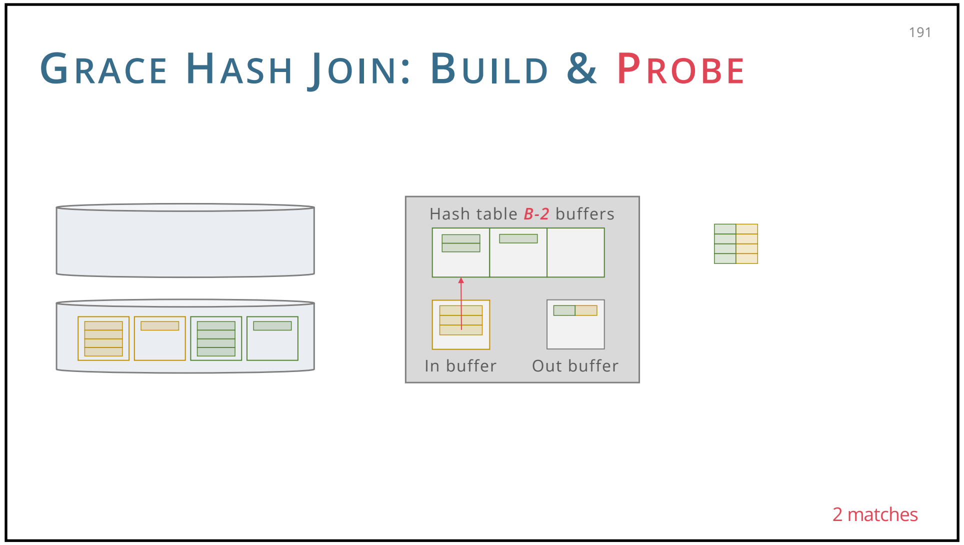
189



Grace Hash Join: Build & Probe

2 matches

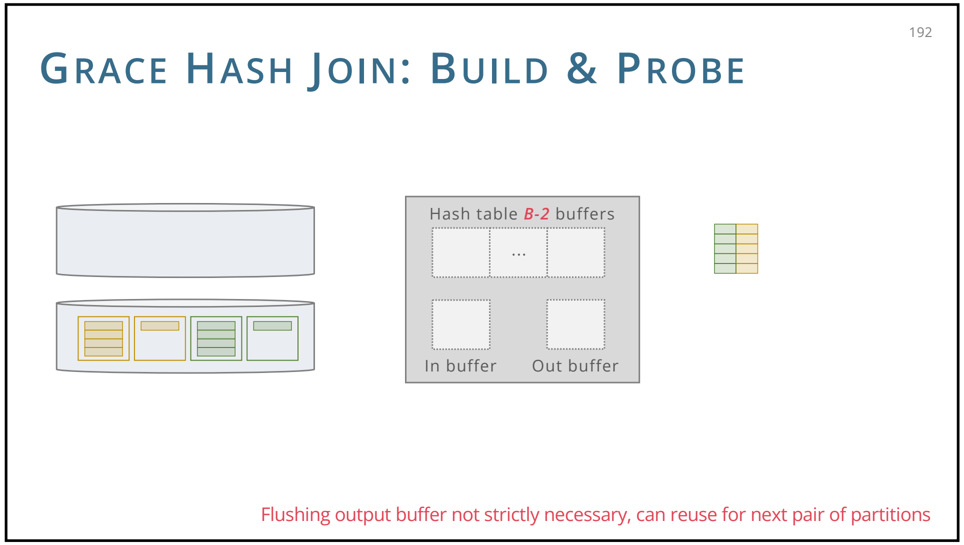
190



Grace Hash Join: Build & Probe

2 matches

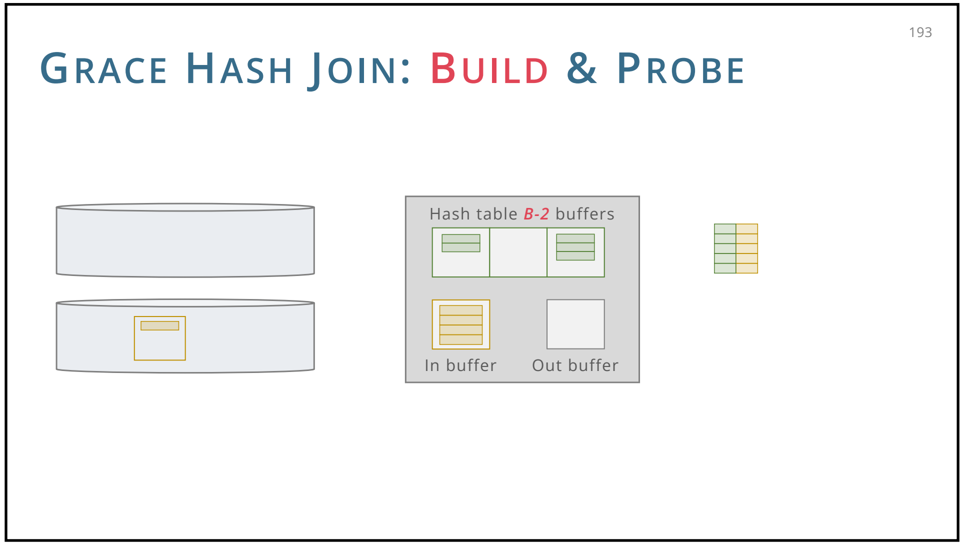
191



Grace Hash Join: Build & Probe

Flushing output buffer not strictly necessary, can reuse for next pair of partitions

192



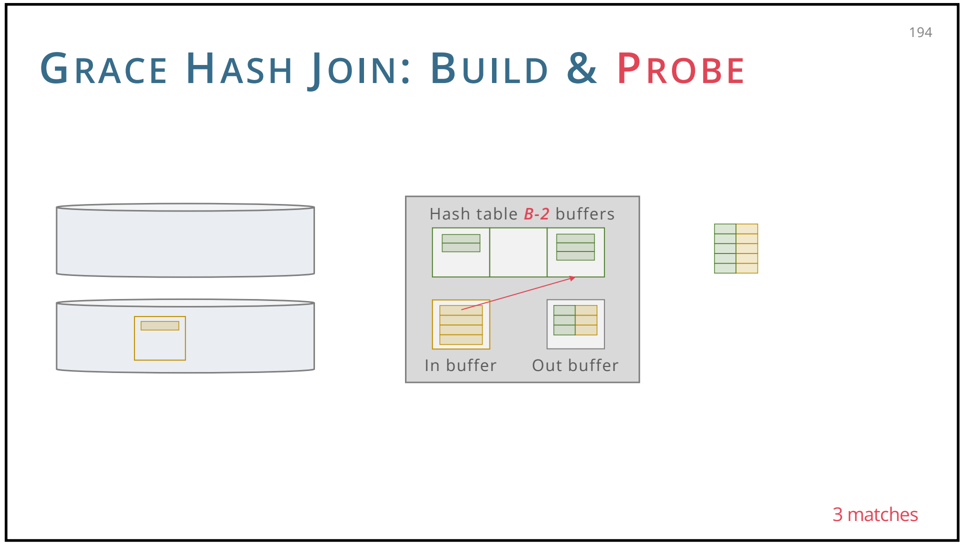
GRACE HASH JOIN: BUILD & PROBE

Hash table *B-2* buffers

In buffer

Out buffer

193



GRACE HASH JOIN: BUILD & PROBE

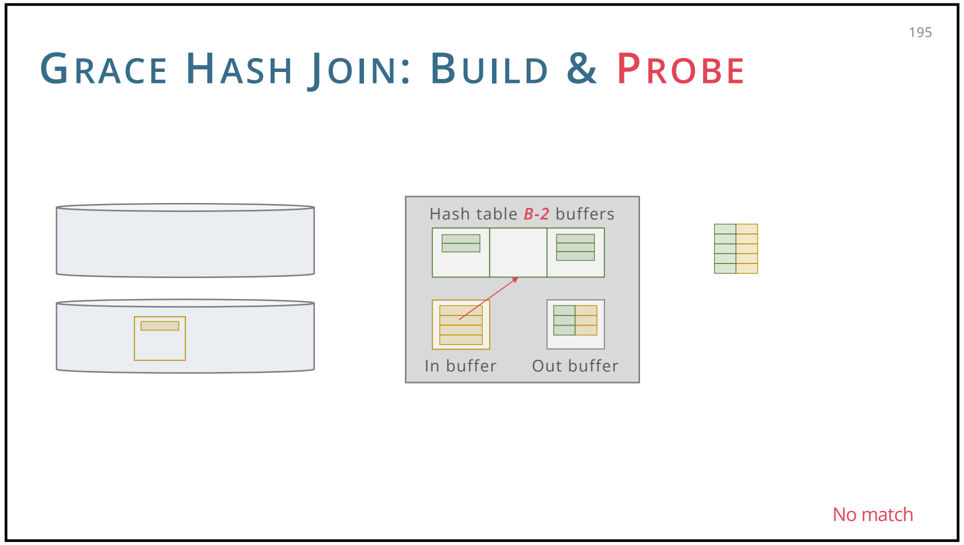
Hash table *B-2* buffers

In buffer

Out buffer

3 matches

194



GRACE HASH JOIN: BUILD & PROBE

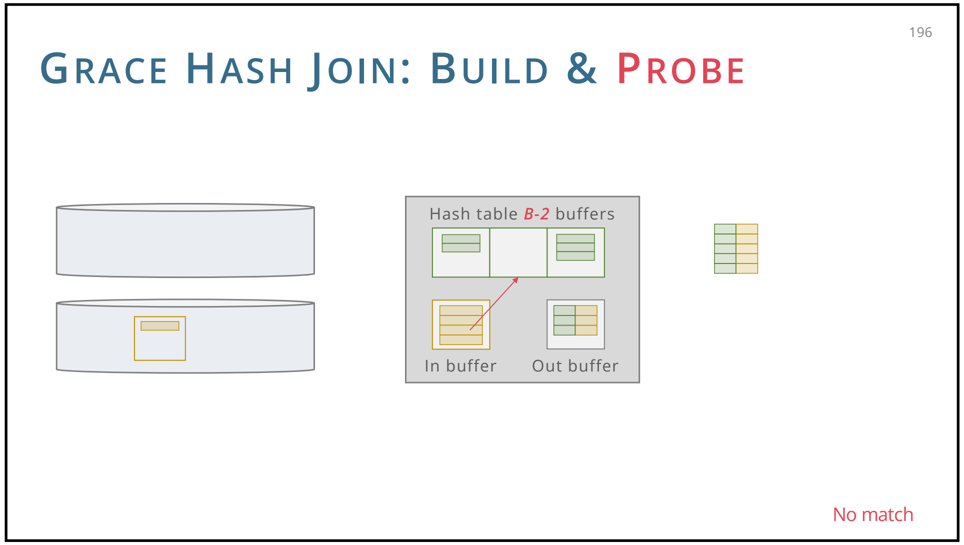
Hash table *B-2* buffers

In buffer

Out buffer

No match

195



GRACE HASH JOIN: BUILD & PROBE

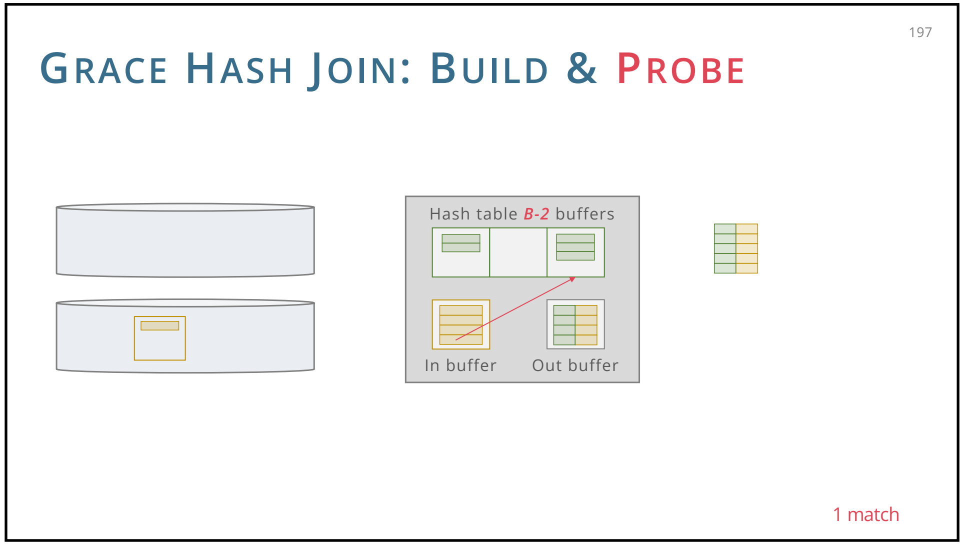
Hash table *B-2* buffers

In buffer

Out buffer

No match

196



197

Grace Hash Join: Build & Probe

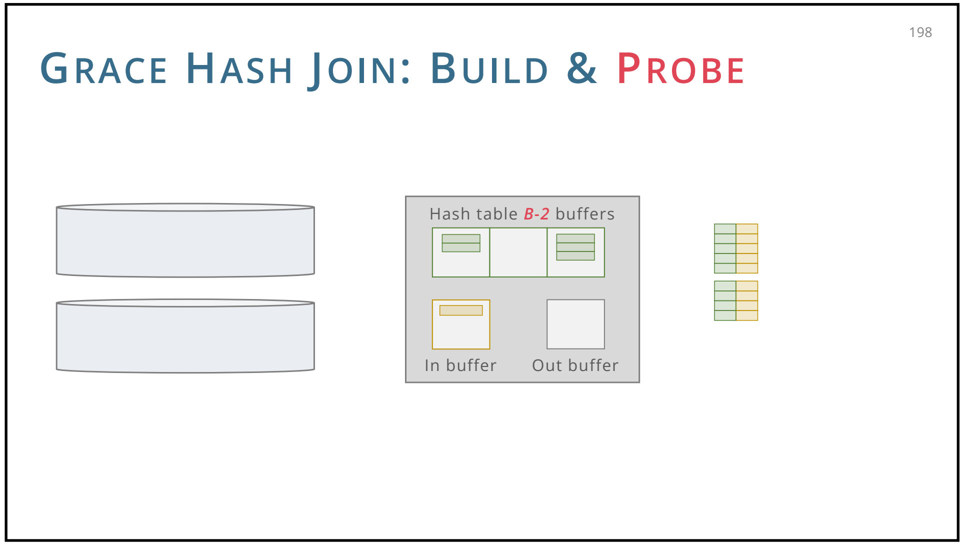
Hash table *B-2* buffers

In buffer

Out buffer

1 match

197



198

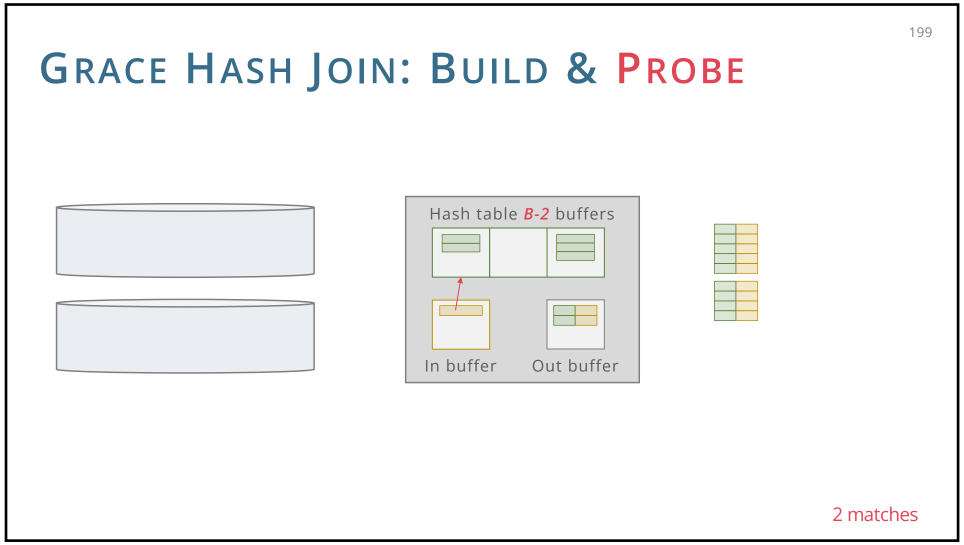
Grace Hash Join: Build & Probe

Hash table *B-2* buffers

In buffer

Out buffer

198



199

Grace Hash Join: Build & Probe

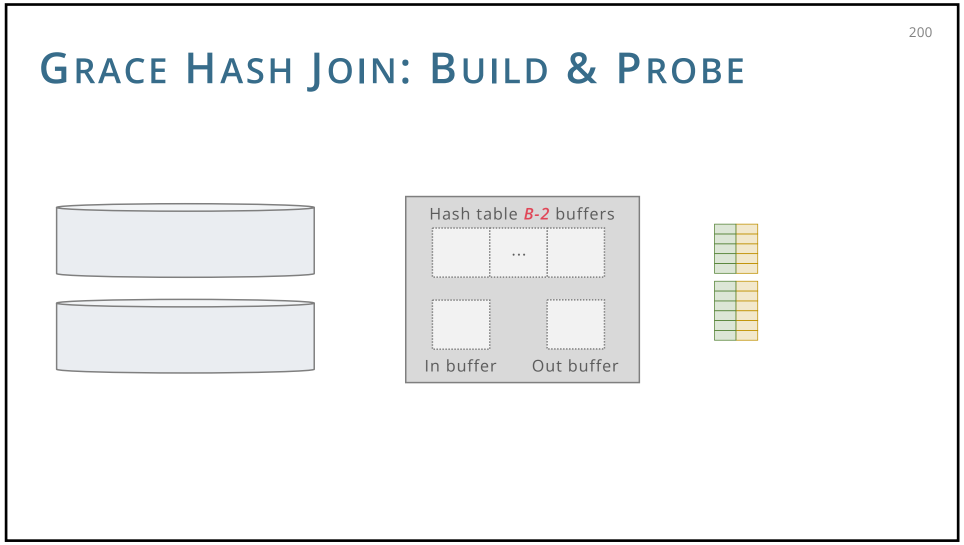
Hash table *B-2* buffers

In buffer

Out buffer

2 matches

199



200

Grace Hash Join: Build & Probe

Hash table *B-2* buffers

…

In buffer

Out buffer

200