Introduction to Algorithms and Data Structures

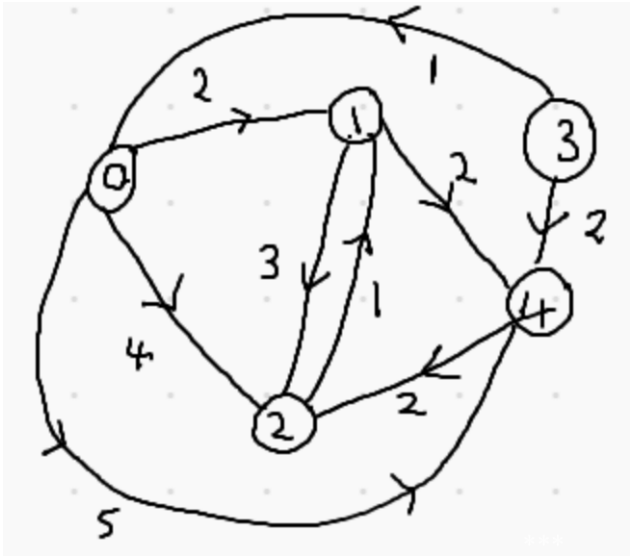
Tutorial 6

your tutor

University of Edinburgh

26th-30th January, 2026

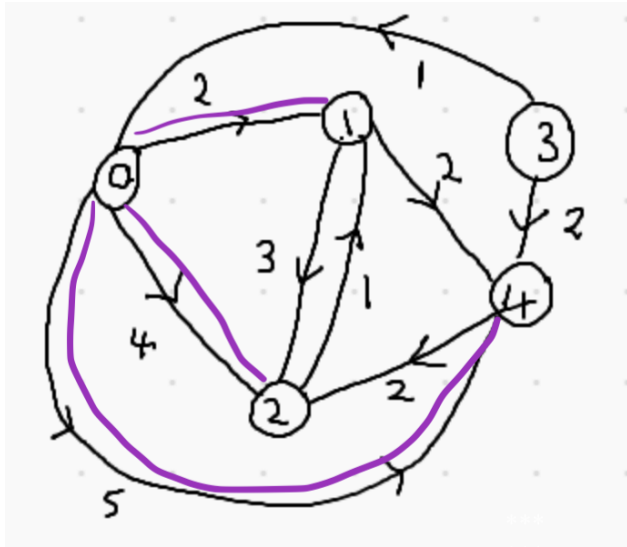
Q1: Dijkstra's Algorithm worked example



- Start with the d, π arrays initialised to ∞ and NULL.
- 0 is the initial vertex being added to the set S (with distance 0), so after that we have the following update to d (no update to π yet):

0	∞	∞	∞	∞
0	1	2	3	4

Q1: Dijkstra's Algorithm worked example



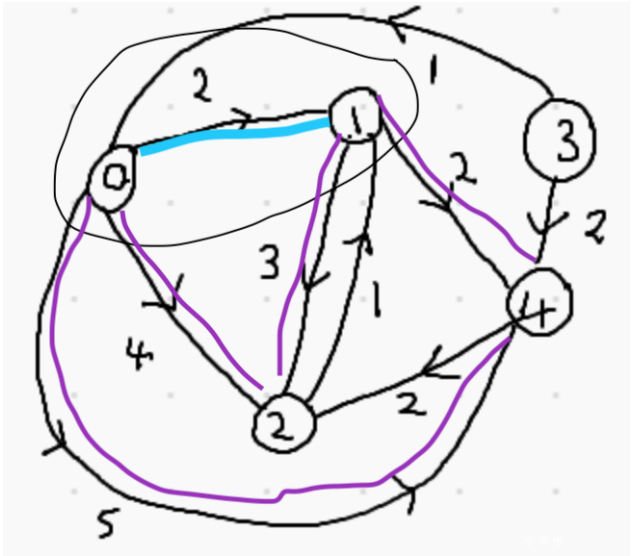
- ($S = \{0\}$) Examine the outgoing edges from $\{0\}$ to $V \setminus S = \{1, 2, 3, 4\}$:
  $(0 \to 1)$: cost $d[0] + 2 = 2$
  $(0 \to 2)$: cost $d[0] + 4 = 4$
  $(0 \to 4)$: cost $d[0] + 5 = 5$

  $\Rightarrow$ *encode all options into* d, π *and commit vertex 1 (cheapest) to* S

0	**2**	4	∞	5
0	1	2	3	4

null	**0**	0	NULL	0
0	1	2	3	4

Q1: Dijkstra's Algorithm worked example



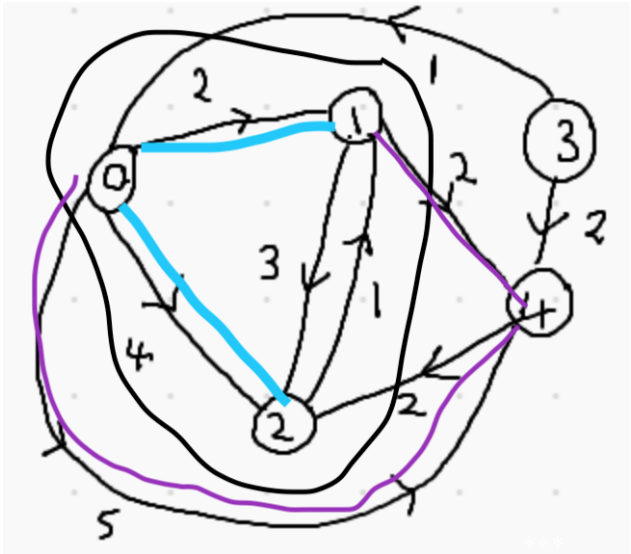
- $(0 \to 2)$: (already fringe, cost 4)
  $(0 \to 4)$: (already fringe, cost 5)
  $(1 \to 2)$: New fringe, cost is $d[1] + 3 = 5$
  $(1 \to 4)$: New fringe, cost is $d[1] + 2 = 4$. Better option for $d[4], \pi[4]$

$\Rightarrow$ **Commit** *a cost-4 option, eg* $(0 \to 2)$*:* $S \leftarrow \{0, 1, 2\}$, *update* $d[4], \pi[4]$:

0	**2**	**4**	∞	4
0	1	2	3	4

null	**0**	**0**	NULL	1
0	1	2	3	4

Q1: Dijkstra's Algorithm worked example



- No new fringe edges this time.

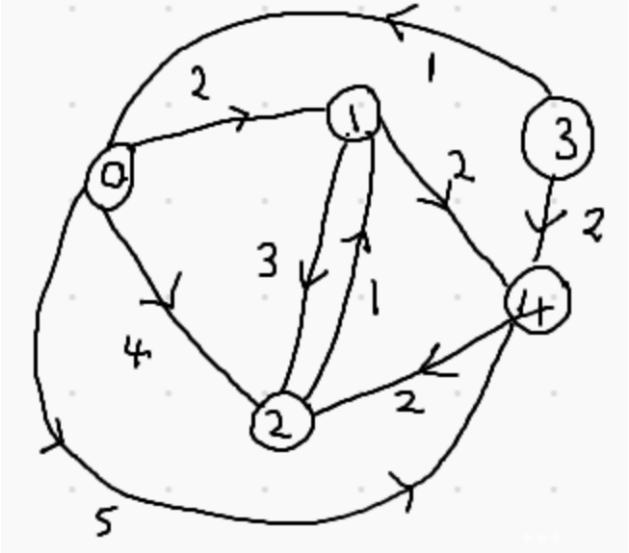
  $\Rightarrow$ **Commit** 4 to S via $(1 \rightarrow 4)$, S becomes $\{0, 1, 2, 4\}$, no updates to d, π.

0	**2**	**4**	∞	**4**
0	1	2	3	4

null	**0**	**0**	NULL	**1**
0	1	2	3	4

After S becomes $\{0, 1, 2, 4\}$, no fringe edges any more (3 has no incoming edges). Hence the algorithm terminates with this p, π.

Q2: Using $\min\{w(u, v)\}$ for the next fringe edge/vertex



Consider the exact same Graph with $s = 0$, but change the criterion we use to select the next fringe edge/vertex: set $(u^*, v^*) = \arg\min\{w(u, v)\}$ and add v^* to S with pre-decessor $\pi[v^*] = u^*$.

We will see that this constructs a too-large value for $d[2]$. Do the workings.

Q3: fractional knapsack

Input: Values $v_i \in \mathbb{N}$ and sizes $w_i \in \mathbb{N}$ for $i = 1, \ldots, n$
("item i has value v_i and size w_i").
Capacity $C \in \mathbb{N}$ ("size of the knapsack").

Fractional knapsack: We want to choose fractional weights $x_i \in [0, 1]$ for every $i \in [n]$ so that we maximize the total weighted value (while fitting in the C):

$$\max \sum_{i=1}^{n} x_i \cdot v_i \tag{1}$$

$$\text{subject to } x_i \in [0, 1], i = 1, \ldots, n \text{ and } \sum_{i=1}^{n} x_i \cdot w_i \leq C \tag{2}$$

Want highest-valued total knapsack i (as measured in (1)).

Q3: fractional knapsack

Suggest to apply the greedy heuristic by choosing some item i "greedily" and adding the largest possible fraction $0 < x_i \leq 1$ of item i that is possible to fit in the (current) leftover capacity of the knapsack, without violating C.

We can think about 2 different greedy rules to select the next item i:

(a) Add the item with the largest v_i value among all remaining items.

(b) Add the item with the largest v_i/w_i ratio among all remaining items.

All students should have tried an example to understand workings!

Does (a) give optimal result on your example?

Does (b) give optimal result on your example?

Q3 (i): fractional knapsack

(i): We need to show that "largest v_i first" fails on some instances.

Here is a specific input to demonstrate non-optimality:
values $v_1 = 3, v_2 = 3, v_3 = 4, v_4 = 5$,
weights $w_1 = 3, w_2 = 4, w_3 = 4, w_4 = 9$,
capacity $C = 12$.

Consider the items in order of value: $i = 4$, then $i = 3$, finally $i = 1, 2$.

Taking $i = 4$: take entire item (as $w_4 < 12$) $\Rightarrow$ $x_4 \leftarrow 1$ and $C' \leftarrow 12 - 9 = 3$.

Next $i = 3$: we have $w_3 = 4$, so we can't fit all of item $i = 3 \Rightarrow$ set $x_3 \leftarrow (C'/w_3)$-fraction, which is $x_3 \leftarrow 0.75$, with the new $C' \leftarrow 3 - w_3 \cdot 0.75 = 0$.

Leftover capacity is now 0, so we don't consider $i = 1, i = 2$.
We have $x_1 = 0, x_2 = 0$.

The total value we get with this version of Greedy is $5 + 0.75 \cdot 4 = 8$.

However, we can get $10.5555555\ldots$ by taking $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1/9$.

Q3 (ii): fractional knapsack

(ii) We have to prove that "largest v_i/w_i first" is guaranteed to construct an optimal assignment for $x_1, \ldots, x_n$ for the input.

We will use a clever transformation,
mapping the general case to "unit weight" equivalent

UNIT-WEIGHT FRACTIONAL KNAPSACK: Set of items $i = 1, \ldots, n$ of *unit weight* each, and with values $v_1, \ldots, v_n \in \mathbb{Q}^+$ respectively. Capacity $C \in \mathbb{N}$.

We transform general WEIGHTED FRACTIONAL KNAPSACK (with $w_i \in \mathbb{N}$ values), to an *equivalent* instance with UNIT-WEIGHTs.

item i: weight w_i, value v_i	$\Leftrightarrow$	w_i different items, each $\widehat{v_i} = v_i/w_i$
capacity C	$\Leftrightarrow$	capacity C
n original items	$\Leftrightarrow$	$\sum_{i=1}^{n} w_i$ items altogether (call this $\widehat{n}$)

Q3 (ii): fractional knapsack

Formal definition of UNIT WEIGHT subproblem:

UNIT-WEIGHT FRACTIONAL KNAPSACK: We are given a set of items $i = 1, \ldots, n$ of *unit weight* each, and with values $v_1, \ldots, v_n \in \mathbb{Q}^+$ respectively. We are also given a capacity $C \in \mathbb{N}$. Our aim is to find fractional values $x_i \in [0, 1], i \in [n]$ such that $\sum_{i=1}^{n} x_i \leq C$ and such that $\sum_{i=1}^{n} x_i \dot{v}_i$ is maximised subject to the capacity constraint.

(note: we need to allow the values to be in $\mathbb{Q}$, as we will build unit-weight instances which have values v_i/w_i)

Two steps to our proof:

- equivalence of Greedy (b) operating on original problem (weights w_i, values v_i) with Greedy (b) operating on unit weight re-formulation (weights 1, values $\widehat{v_i} = v_i/w_i$).
- correctness of Greedy (b) on unit weight instances

Q3 (ii): equivalence of Greedy (b) on unit weight re-formulation

I. The mapping to the UNIT WEIGHT version is:

$$\begin{aligned} \text{item } i\text{: weight } w_i\text{, value } v_i &\Leftrightarrow w_i \text{ different items, each value } v_i/w_i \\ \text{capacity } C &\Leftrightarrow \text{capacity } C \\ n \text{ original items} &\Leftrightarrow \textstyle\sum_{i=1}^{n} w_i \text{ items altogether (call this } \widehat{n}) \end{aligned}$$

(we define "break-ties" on the items of the unit-weight construction to group the unit-weight items from the same item i together.)

Steps of Greedy (b) on the new unit-weight instance is equivalent to the operation of Greedy (b) on the original instance (we set x_i on the original instance to be the sum of the x-values for corresponding unit-weight items).

Q3 (ii): Greedy (b) on unit weights instances is optimal

II. Greedy (b) is guaranteed to return an optimal solution for every instance of UNIT-WEIGHT FRACTIONAL KNAPSACK.

PROOF OF II. In the UNIT-WEIGHT case

- all weights are 1
- the values $\widehat{v_i}$ (for $i \in [\widehat{n}]$) are positive *rational numbers*.
- We can assume $\widehat{n} > C$

  (if not, then the capacity allows us to set $x_i \leftarrow 1$ for all items, and optimal and Greedy (b) are exactly the same).

Since all $\widehat{v_i}$ are > 0, it is the case that if $\widehat{n} > C$, we expect an *optimal* x' solution to satisfy $\sum_{i=1}^{\widehat{n}} x'_i = C$ (to use all capacity).

We observe that Greedy (b) *always constructs a solution which uses all capacity* (assuming $\widehat{n} > C$).

Q3 (ii): Greedy (b) on unit weights instances is optimal

PROOF OF II. Consider the first step of Greedy (b), assuming $C > 0$. Value/weight scores are the $\widehat{v_i}$s.

- Let i^* be the item of maximum value $\widehat{v_{i^*}}$ in the collection of items.
- Greedy (b) chooses this item to add, setting $x_{i^*} \leftarrow \min\{1, C\}$, which (as C is a whole number, and $C > 0$) is 1.

Will show there must be some optimal solution x' with $x'_{i^*} = 1$.

Work towards proof by contradiction:

- write x' for any optimal solution which has the maximum assignment to x'_{i^*} among all optima. Suppose that this x'_{i^*} is *not* equal to 1.
- Then we can find $j \in [\widehat{n}] \setminus \{i^*\}$ such that $x'_j > x_j$ (due to our observations that both optimal x' and Greedy (b) x will use all of C).
- By choice of i^*, we know that $\widehat{v_{i^*}} \geq \widehat{v_j}$.

Q3 (ii): Greedy (b) on unit weights instances is optimal

- write x' for any optimal solution which has the maximum assignment to x'_{i^*} among all optima. Suppose that this x'_{i^*} is *not* equal to 1.
- Then we can find $j \in [\widehat{n}] \setminus \{i^*\}$ such that $x'_j > x_j$ (due to our observations that both optimal x' and Greedy (b) x will use all of C).
- By choice of i^*, we know that $\widehat{v_{i^*}} \geq \widehat{v_j}$.

Hence we can transform x' by increasing x'_{i^*} by $\min\{1 - x'_{i^*}, x'_j - x_j\}$ and decreasing x'_j by the same amount ... and the *value* of x' does not decrease.

This **contradicts** the choice of x' being the one with maximum value x'_{i^*} (among all optimum assignments x').

Hence there is an optimum assignment x' with $x'_{i^*} = 1$, just as Greedy (b) has assigned.

This justifies step 1 of Greedy (b), taking the item of maximum value.

Q3 (ii): Greedy (b) on unit weights instances is optimal

Proof of II. (wrapping up)
If we had $C = 1$, then this is also the final step of Greedy (b), and we are finished.

Otherwise, continuation of Greedy (b) is equivalent to a new instance ($\widehat{v_i}, i \in [\widehat{n}] \setminus \{i^*\}$ with $C - 1$) of UNIT-WEIGHT FRACTIONAL KNAPSACK.
We can apply induction to infer that there is an optimal assignment that matches the one constructed by Greedy (b).

We relate the execution of Greedy (b) with the $\widehat{v_i} = v_i/w_i$ weights to the original problem where we have $v_i, w_i \in \mathbb{N}$. □

Q3 (ii): remarks on the correctness of Greedy (b)

REMARK: Might be confused that we have proven correctness of Greedy (b) using ranking of the $\widehat{v_i}$ values alone? Does this contradict our counter-example for Greedy (a)?
What does it tell us about Greedy (a) in the unit weights case?

REMARK: The definition of the UNIT-WEIGHT FRACTIONAL KNAPSACK states inputs $v_i, i \in [b]$, and capacity $C \in \mathbb{N}$. I wrote our proof with $\widehat{v_i}$ and $\widehat{n}$ to emphasise the fact the application to the modified weights/items of the original problem.

Q4 (a): 0/1 knapsack

Algorithm maxKnapsack($w_1, \ldots, w_n$, C)

1. initialise row 0 of kp to "all-0s"
2. initialise column 0 of kp to "all-0s"
3. **for** ($i \leftarrow 1$ **to** n) **do**
4. **for** ($C' \leftarrow 1$ **to** C) **do**
5. **if** ($w_i > C'$) **then**
6. $kp[i, C'] \leftarrow kp[i-1, C']$
7. **else**
8. $kp[i, C'] \leftarrow \max\{kp[i-1, C'], kp[i-1, C'-w_i] + v_i\}$
9. **return** $kp[n, C]$

$$kp(k+1, C') = \begin{cases} kp(k, C') & w_{k+1} > C' \\ \max\{kp(k, C'),\ v_{k+1} + kp(k, C' - w_{k+1})\} & \text{otherwise} \end{cases}$$

Q4 (b): 0/1 knapsack

The following is the main dynamic programming table, where the cell value for (i, j) is the value of the "max-knapsack which uses items 1 to i to achieve weight at most j".

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	2	2	2	2	2
2	0	0	3	3	3	5	5	5
3	0	0	3	4	4	7	7	7

Q4 (c): 0/1 knapsack

Greedy algorithm (b) will *not* deliver an optimal solution for all instances of the 0/1 knapsack problem.

One counterexample is $v_1 = 3, v_2 = 5, v_3 = 2$ and $w_1 = 3, w_2 = 4, w_3 = 2$. $C = 5$

In this case Greedy (b) will first add item 2 ($v_2/w_2 = 1.25$).
Next have residual capacity $C' = 5 - 4 = 1$, and in the 0/1 setting, this means that we cannot add any extra items (as weights are 2 and 3),
Hence we return value 4.

Optimum is taking items 1 and 3: use capacity $3 + 2 = 5 = C$, get value $3 + 2 = 5$