UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR08026 INFORMATICS 2: INTRODUCTION TO ALGORITHMS AND DATA STRUCTURES

Monday 9th May 2022

13:00 to 15:00

INSTRUCTIONS TO CANDIDATES

1. **Answer all five questions in Part A, and two out of three questions in Part B. Each question in Part A is worth 10% of the total exam mark; each question in Part B is worth 25%.**
2. **This is an Open Book examination.**

Convener: D.K.Arvind
External Examiner: J.Gibbons

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

PART A

1. (a) Consider the following sequence of commands in Python:

```
A = [500,'miles']
B = A[:]
C = [A[0] is B[0], A[1] is B[1]]
```

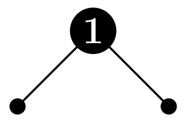
Draw a schematic representation of the contents of program memory after executing these three commands, showing the data living on the stack and the heap and how these are related. A memory location containing a reference to another memory location should be represented by an arrow, as in the examples in lectures. [*7 marks*]

(b) Suppose now that the execution of the above commands is followed by the execution of

```
B[1] = 'more'
```

Draw a diagram for the new state of program memory. [*3 marks*]

2. Consider the following trivial red-black tree representing the set $\{1\}$.

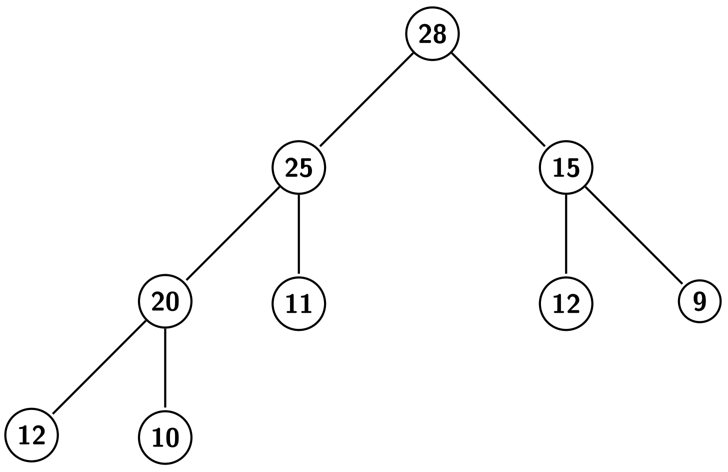


Explain with the help of diagrams what happens when the following operations are performed in succession:

insert(2), insert(3), delete(2)

You should show any intermediate steps performed in the course of these operations as well as the final results. Your diagrams should include the trivial nodes. You may indicate the colours of nodes in any way you wish. [*10 marks*]

3. (a) Consider the (Max) Heap below, showing the keys of the items stored within it. Suppose that we carry out the following sequence of operations: Heap-Extract-Max(), Max-Heap-Insert(17), Max-Heap-Insert(16) and Heap-Extract-Max(). What is the order of the (remaining) elements after this sequence? [*6 marks*]



28	**25**	**15**	**20**	**11**	**12**	**9**	**12**	**10**	**-**

(b) The key operations of Heap-Extract-Max() and Max-Heap-Insert() are known to have worst-case running-time $\Theta(\lg(n))$ for a Heap containing n items, in the general case.

Suppose we consider the setting of a priority queue, where the items only have a few different options for key values - say $\{1, 2, 3\}$ are the only possible key values. Does this change the worst-case running-time for Heap-Extract-Max() and Max-Heap-Insert(), and if so, how? [*4 marks*]

Write a sentence or two to justify your answer (for each operation).

4. (a) Compute the *edit distance* of the following two words, completing both the distance table d and also the table a. In building a, list each of the three options 1/2/3 that are relevant for cell $[i, j]$ (that would achieve the edit distance for those prefix strings). [*6 marks*]

carnet around

Explain your workings for the two final rows of the tables.

(b) Use the details in a to compute the *number of different alignments* that will achieve the edit distance. [*4 marks*]

5. (a) Draw a diagram for a register machine that tests whether a given number n is divisible by 3. Your machine should have two registers A and B, with the input n supplied in A, and B initialized to 0. Your machine should have just a single exit point, and the value of B on exit should be 1 if n is divisible by 3, 0 otherwise. Each junction between basic components should be marked with an arrowhead, as in the examples in lectures.

   It does not matter if the value of n is lost in the course of the computation.

   [*7 marks*]

   (b) Would you expect it to be possible to build a register machine that tests whether a given number n is *prime*? Very briefly justify your answer, drawing on any relevant statements from lectures. (Do not attempt to construct such a machine explicitly.) [*3 marks*]

PART B

1. (a) For each of the following pairs of functions f and g, state which of the five statements $f = o(g)$, $f = O(g)$, $f = \Theta(g)$, $f = \Omega(g)$, $f = \omega(g)$ are true. (You may simply list the relevant set of symbols $o, O, \Theta, \Omega, \omega$). Informally justify your answers.

   i. $f(n) = \lg(n^n)$, $g(n) = n\sqrt{n}$.

   ii. $f(n) = \lg(n!)$, $g(n) = n \lg n$. (Hint: rewrite $\lg(n!)$ as a summation.) *[10 marks]*

   (b) A certain algorithm published in 1971 (here called 'A') is able to multiply two n-digit decimal numbers in time $\Theta(n \lg n \lg \lg n)$, spectacularly beating the obvious quadratic-time method. In 2019 it was discovered that a more elaborate algorithm ('B') can do the same in time $O(n \lg n)$.

   For the purpose of this question, we shall suppose that multiplying two n-digit numbers using algorithm A always takes between $6n \lg n \lg \lg n$ and $7n \lg n \lg \lg n$ computation steps, and that doing so with algorithm B always takes between $105n \lg n$ and $120n \lg n$ steps.

   On the basis of this information, identify:

   i. A constant N_1 such that A always performs faster than B on inputs of size $< N_1$. Your N_1 should be as large as possible.

   ii. A constant N_2 such that B always performs faster than A on inputs of size $> N_2$. Your N_2 should be as small as possible.

   (You may give your answers as unevaluated expressions.) Explain your reasoning in each case. *[6 marks]*

   (c) By developing the idea used in (b), prove that $n \lg n = o(n \lg n \lg \lg n)$, arguing rigorously from the formal definition of o. *[9 marks]*

2. Below is the LL(1) parse table for a simple grammar for shell commands, consisting of a command name followed by (perhaps) some options and then some filenames. The start symbol is Shell. The terminals are com, opt, file.

	com	opt	file	$
Shell	com Args			
Args		Opts Files	Opts Files	Opts Files
Opts		opt Opts	ϵ	ϵ
Files			file Files	ϵ

(a) Show how the LL(1) parsing algorithm executes on the input string

com opt

showing at each stage the operation performed, remaining input and stack state. Use the table format adopted in Lecture 23. [*10 marks*]

(b) Explain what happens if we try to run the algorithm on the input

com com

How exactly does the algorithm detect the error? [*5 marks*]

(c) One reason that LL(1) parsing may sometimes fail is that a predicted terminal does not match the symbol appearing next in the input. Could this ever happen for the grammar considered above? Justify your answer. [*4 marks*]

(d) Consider now the following grammar (with start symbol S) for even-length palindromic strings over $\{a, b\}$:

$$S \rightarrow \epsilon \mid aSa \mid bSb$$

Is this grammar ambiguous? Is it LL(1)? Briefly justify your answers. [*6 marks*]

3. The SAT problem is the central problem for the complexity class NP, concerning the satisfiability (or not) of a given formula Φ in *conjunctive normal form (CNF)*

$$\Phi = \bigwedge_{j=1}^{m} C_j,$$

where each clause C_j is the *disjunction* $\vee$ of a number of literals over the set of logical variables $\{x_1, \ldots, x_n\}$.

Over the years, a huge number of approximation algorithms and heuristics have been developed to find high-quality assignments for CNF formulae, including our (derandomized) (8/7)-approximation algorithm for the special case of 3-CNF formulae (each clause has 3 literals, over 3 distinct variables).

(a) Apply derandomization of conditional expectations to obtain a specific assignment to $\{x_1, x_2, x_3, x_4\}$ which satisfies $\geq 9 \times \frac{7}{8}$ of the clauses of formula Φ shown below. Consider the variables in the order of increasing index, and show your workings. [*15 marks*]

$$\begin{aligned}\Phi \;=\; & (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee x_4) \wedge \\ & (x_1 \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_3} \vee x_4) \wedge \\ & (x_2 \vee x_3 \vee x_4) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee x_3 \vee \bar{x_4}).\end{aligned}$$

(b) The algorithm for $k = 3$ above can be generalised to the 4-CNF case where each clause contains exactly 4 literals (over 4 distinct variables).

i. What is the *expected number of satisfied clauses* of an initial 4-CNF formula Φ with m clauses? [*2 marks*]

ii. Consider an intermediate stage during conditional derandomization, where we have set a value for some variables and have some already satisfied clauses, some already refuted clauses, and some unresolved clauses. Let Φ' be the collection of not-yet-resolved clauses of sizes $1, 2, 3$ and 4. Suppose x_i is the next variable to be assigned a bit value, and suppose that the set of clauses of Φ' *that involve* x_i consist of:

- k_1^+ clauses of size 1 which contain x_i as the only remaining literal
- k_1^- clauses of size 1 which contain $\bar{x}_i$ as the only remaining literal
- k_2^+ clauses of size 2 which contain x_i
- k_2^- clauses of size 2 which contain $\bar{x}_i$
- k_3^+ clauses of size 3 which contain x_i
- k_3^- clauses of size 3 which contain $\bar{x}_i$
- k_4^+ clauses of size 4 which contain x_i
- k_4^- clauses of size 4 which contain $\bar{x}_i$

Write a short linear inequality[1] of the form

$$f(k_1^+, k_2^+, k_3^+, k_4^+) \geq g(k_1^-, k_2^-, k_3^-, k_4^-)$$

determining the conditions under which setting $x_i \leftarrow 1$ gives a conditional expectation greater than or equal to setting $x_i \leftarrow 0$. Justify your details. [*8 marks*]

[1]A linear inequality over variables x, y, z is something like $x/3 + y \geq z$ or $2x \geq z - y$. The meaning of f and g in the phrasing is only to indicate that we want an inequality where all the k_j^+ terms are on one side (the f), and all the k_j^- terms are on the other side (the g). For example, $x/3 + y \geq z$ would be a linear inequality of the form $f(x, y) \geq g(z)$.