UNIVERSITY OF BRISTOL

Not a Real Examination Period

Department of Computer Science

2nd Year Practice Paper for the Degrees of
Bachelor in Computer Science
Master of Engineering in Computer Science
Master of Science in Computer Science

COMS20010
Algorithms II

TIME ALLOWED:
2 Hours

This paper contains **eighteen** questions.
All questions will be marked except where otherwise indicated.
If you attempt a question and do not wish it to be marked, delete it clearly.
The maximum for this paper is **150 marks**.

Other Instructions

1. **You may bring up to four A4 sheets of pre-prepared notes with you into the exam, but no other written materials.**
2. **You may use a calculator with the Faculty seal of approval if you wish.**

TURN OVER ONLY WHEN TOLD TO START WRITING

Section 1 — Short-answer questions (75 marks total)

You do not need to justify your answers for any of the questions in this section, and you will not receive partial credit for showing your reasoning. Just write your answers down in the shortest form possible, e.g. "A" for multiple-choice questions, "True" for true/false questions, or "23" for numerical questions. If you do display working, circle or otherwise indicate your final answer, as if it cannot be identified then the question will not be marked.

Question 1 (5 marks)

For **each** of the following statements, identify whether it is true or false.

(a) $n \in O(n^2)$. (1 mark)

(b) $n \in \omega(\log^2 n)$. (1 mark)

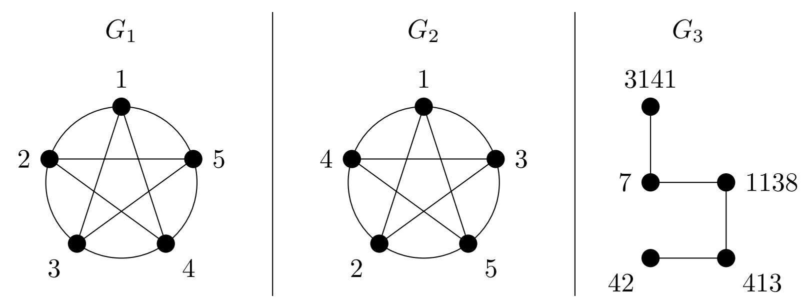
(c) $n^2 + 50n - 12 \in O(\frac{1}{2}n^2 - n + 100)$. (1 mark)

(d) $5^n \in \Theta(6^n)$. (1 mark)

(e) $mn^2 + m^2n \in O(m^{100}n)$. (1 mark)

Question 2 (5 marks)

Consider the three graphs G_1, G_2 and G_3 shown below.



For **each** of the following statements, identify whether it is true or false.

(a) G_1 is equal to G_2. (1 mark)

(b) G_2 is isomorphic to G_2. (1 mark)

(c) G_1 contains a subgraph isomorphic to G_3. (1 mark)

(d) G_1 contains an induced subgraph isomorphic to G_3. (1 mark)

(e) G_1 contains an Euler walk from some vertex back to itself. (1 mark)

Question 3 (5 marks)

You are working on a recently-established competitor to Google Maps, and you have been asked to estimate storage requirements for adding the small landlocked nation of Tropico to the service. Tropico's road network will be stored internally as a graph: Each junction and dead end is a vertex, and the roads joining them are edges. After doing some research, you discover that Tropico's road network has 100 dead ends, 350 3-way junctions, 50 4-way crossroads, and one terrifying central roundabout with 20 separate exits (which you should consider as a 20-way junction). How many edges will the resulting graph have? Choose **one** of the following options.

A. 645.

B. 685.

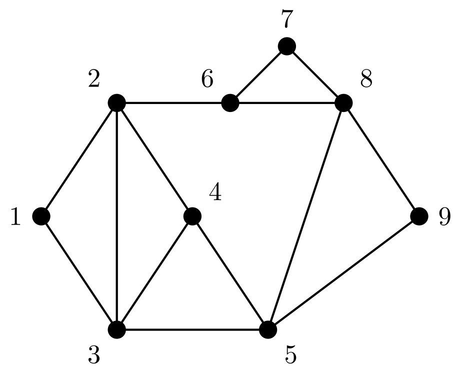
C. 1290.

D. 1370.

E. None of the above.

Question 4 (5 marks)

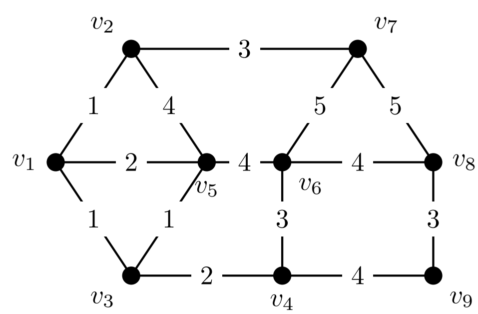
Consider a depth-first search in the following graph starting from vertex 1.



In the implementation of depth-first search given in lectures, we say vertex i is *explored* when `explored`$[i]$ is set to 1. Which vertex will be explored sixth? Assume that whenever the search has a choice of two or more vertices to visit next, it picks the vertex with the lowest number first.

Question 5 (5 marks)

Consider the following edge-weighted graph.



Which of the following edges will **always** be selected as part of a minimum spanning tree by Prim's algorithm starting from v_1, no matter how it breaks ties? Choose **one** of the following options.

A. $\{v_3, v_5\}$.

B. $\{v_6, v_8\}$.

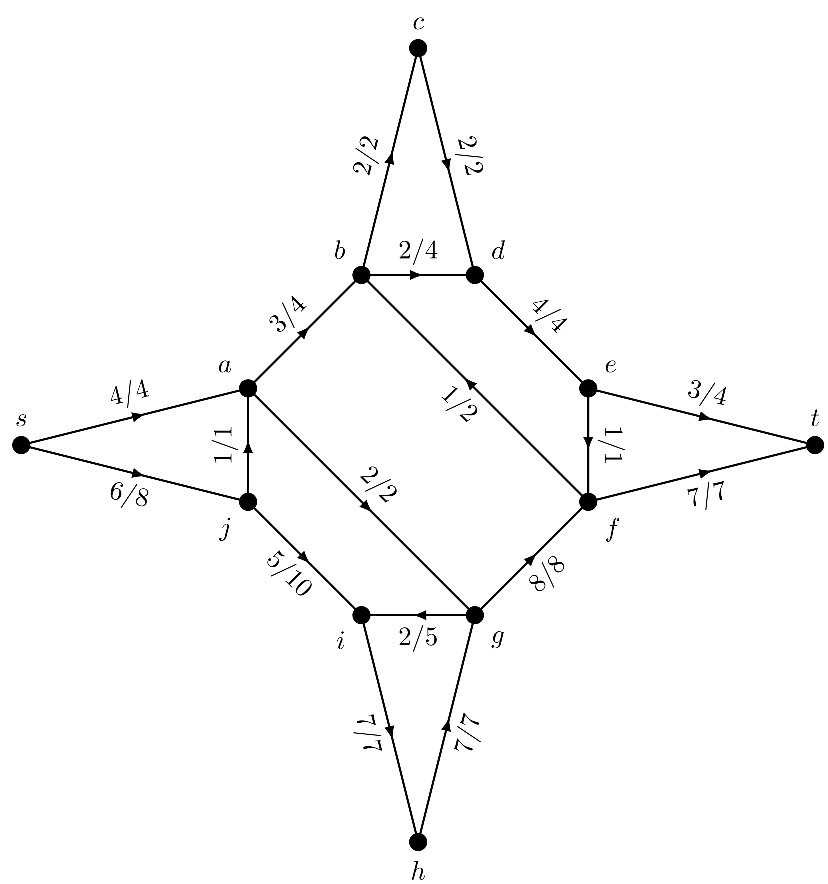
C. $\{v_8, v_9\}$.

D. More than one of these edges will always be selected.

E. None of these edges will always be selected.

Question 6 (5 marks)

Consider the following vertex flow network with source s, sink t and flow f.



(a) What is the value of f? (1 mark)

(b) What is $f^-(a)$? (1 mark)

(c) Is $(\{s, b, e\}, \{a, c, d, f, g, h, i, j, t\})$ a valid cut? (1 mark)

(d) Give an augmenting path for f. (2 marks)

Question 7 (5 marks)

Which of the following CNF formulae are satisfiable?

(a) $x \vee \neg x$. (1 mark)

(b) $\neg a \wedge (a \vee \neg c) \wedge (a \vee c)$. (1 mark)

(c) $(x \vee y) \wedge (x \vee \neg y)$. (1 mark)

(d) $(\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (a \vee b \vee c)$.
(1 mark)

(e) $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge \cdots \wedge (x_{99} \vee \neg x_{100}) \wedge (\neg x_{99} \vee x_{100})$.
(1 mark)

Question 8 (5 marks)

Context: You are attempting to steal a valuable work of art from the Tate Modern. By using the ventilation ducts, you believe you can steal any artwork not directly in view of a camera. With great difficulty, you have determined the number of cameras in the building and a limited set of points where they could be mounted. You also know, for each mounting point, which artworks a camera mounted there would protect, and you have recorded this information in the form of a bipartite graph. Unfortunately, you haven't been able to narrow down the cameras' locations exactly. You don't want to go to the trouble of breaking in and then leave empty-handed, so you would like to know: Is it true that *however* the cameras are mounted, you will be able to steal at least one artwork?

Problem: The ARTTHIEF problem is defined as follows. An instance of the problem consists of an integer k, a set A of artworks, a set P of mounting points, and a bipartite graph G with vertex classes A and P. The answer is `Yes` if for all sets of camera locations $C \subseteq P$ with $|C| = k$, there exists $a \in A$ which is not adjacent to any $p \in P$. Otherwise, the answer is `No`. Which of the following statements has a short and simple proof?

(a) ARTTHIEF is in NP.

(b) ARTTHIEF is not in NP.

(c) ARTTHIEF is in Co-NP.

(d) ARTTHIEF is not in Co-NP.

(e) More than one of the above, or none of the above.

Question 9 (5 marks)

Consider the "unoptimised" union-find data structure presented in lectures, in which a sequence of n operations has a worst-case running time of $\Theta(n \log n)$ rather than $\Theta(n\alpha(n))$. Let G be the graph of such a data structure initialised with the following commands:

`MakeUnionFind`([10]);
`Union`(3, 4);
`Union`(3, 5);
`Union`(3, 6);
`Union`(1, 2);
`Union`(1, 4);
`Union`(7, 1).

(a) How many components does G have? (2 marks)

(cont.)

(b) What is the maximum depth of any component of G? (Remember that depth is the greatest number of **edges** from the root to any leaf.) (3 marks)

Question 10 (5 marks)
You are implementing rudimentary pathfinding in a video game, trying to find the shortest paths from a large number of enemies to the player in a maze. Which of the following algorithms would be the most efficient choice for this situation?

(a) Breadth-first search.

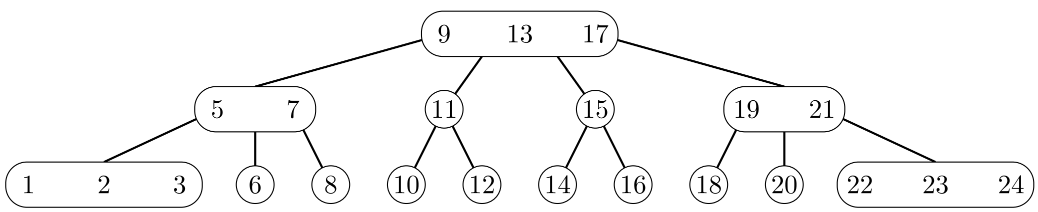
(b) Depth-first search.

(c) Dijkstra's algorithm.

(d) The Bellman-Ford algorithm.

(e) None of the above algorithms would work.

Question 11 (5 marks)
Let T be the 2-3-4 tree below.

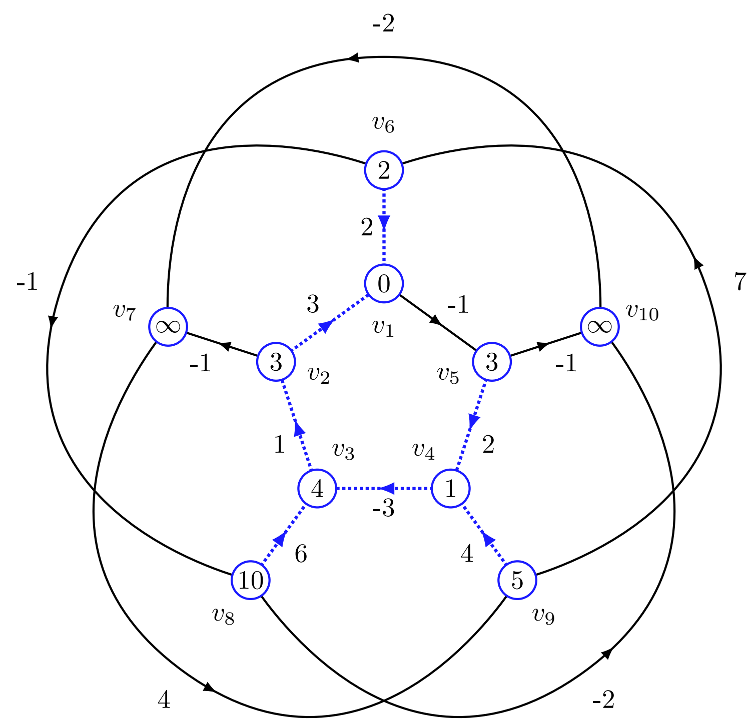


Let U be the 2-3-4 tree formed by first using the `Insert` operation to add 4 into T, then using the `Delete` operation to remove 15 from T.

(a) What is the path traversed on applying the `Find` operation to search for 3 in U? (For example, in T, this would be $(9\ 13\ 17), (5\ 7), (1\ 2\ 3)$.) (2 marks)

(b) What is the path traversed on applying the `Find` operation to search for 16 in U? (3 marks)

Question 12 (10 marks)
Consider the edge-weighted directed graph below, pictured part of the way through executing the Bellman-Ford algorithm to find the distances $d(v, v_1)$ for all vertices v. The current bounds on distance recorded by the algorithm are written inside each vertex. The edges currently selected by the algorithm are drawn thicker, dotted, and in blue.



Carry out **one** further iteration of the Bellman-Ford algorithm — that is, updating each vertex exactly once — processing the vertices in the order $v_1, v_2, \ldots, v_{10}$. After carrying out this iteration:

(a) What is the weight of v_5? (2 marks)

(b) What is the weight of v_7? (2 marks)

(c) What is the weight of v_{10}? (2 marks)

(d) What is the currently-stored path from v_7 to v_1? (2 marks)

(e) Is another iteration of the algorithm required to achieve accurate distances to v_1? (2 marks)

Question 13 (10 marks)

Context: You are a lowly apprentice to the master dwarven blacksmith Cheery Littlebottom, and you would like to use the facilities in your spare time to reforge the dread blade of A'Neem'Ay before the sacred festival of Jankon; this will allow you take over the world and avoid ever having to do Cheery's laundry again. Unfortunately, while Cheery's forge contains many anvils, most of them are in use most of the time, and as an apprentice you are unable to book any anvil time of your own — you must work in the gaps left by more important people. Worse, before you can make use of an anvil, you must spend a full day secretly dedicating it to A'Neem'Ay, and only one anvil can be dedicated this way at once. The one piece of good news is A'Neem'Ay has given you

(cont.)

the ability to see the future, and you know in advance how much time you'll be able to get at each anvil on each day.

Problem: You are given a set of k anvils $A_1, \dots, A_k$, and a time limit of D days. For each anvil A_i, you are also given a length-D list h_i (indexed from 1) such that $h_i[j] \in \{1, \dots, 16\}$. For all $j \in [D]$, on day j, if you are currently at anvil i, then you can **either** spend $h_i[j]$ hours working at anvil A_i, **or** spend the full day switching to a different anvil. Your goal is to maximise your total time spent working, across all anvils. **Fill in the blanks** of the following dynamic programming algorithm, which solves the problem in $O(kD)$ time. **Hint:** This algorithm makes use of a space-saving technique that was also used in the Bellman-Ford algorithm.

(The first four blanks are worth two marks each, the last two are worth one mark each. **Don't copy the whole algorithm out**, just write what should go in each blank!)

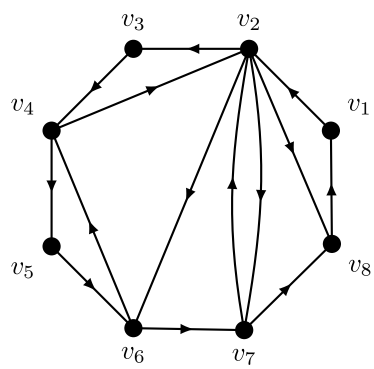
Algorithm: WORLDCONQUEST

```
Let actions[i] ← "" for all i ∈ [k].
Let hours[i], temp[i] ← 0 for all i ∈ [k].
for j = 1 to d do
    for i = 1 to k do
        temp[i] ← hours[i].
    for i = 1 to k do
        Let foo ← ______ + ______.
        Let bar ← ______{temp[x]: x ∈ [k]}.
        Let ℓ be such that temp[ℓ] = ______.
        if foo > bar then
            hours[i] ← foo.
            Add "Work on the sword. " to the ______ of actions[i].
        else
            hours[i] ← bar.
            Add "Switch to anvil ℓ. " to the ______ of actions[i].
for i = 1 to k do
    Add "Start at anvil i. " to the start of actions[i].
Let ℓ be such that hours[ℓ] = max{hours[i]: i ∈ [k]}.
Return actions[ℓ].
```

Section 2 — Long-answer questions (75 marks total)

In this section, you should give the reasoning behind your answers unless otherwise specified — you **will** receive credit for partial answers or for incorrect answers with sensible reasoning.

Question 14 (10 marks)
Consider the directed graph G below.



(a) Express G in adjacency matrix form. (5 marks)

(b) Express G in adjacency list form. (5 marks)

Question 15 (5 marks)
The *Bacon number* of an actor is defined as follows. Kevin Bacon has a Bacon number of zero. The Bacon number of another actor is the shortest chain of co-stars linking them to Kevin Bacon. For example, Elvis Presley was in Change of Habit with Edward Asner, and Edward Asner was in JFK with Kevin Bacon, so Elvis Presley has a Bacon number of at most 2. Elvis was never in any movies with Kevin directly, so he has a Bacon number of exactly 2. Given API access to IMDB, **briefly summarise** an efficient algorithm to output the Bacon number of a given actor by reducing to a problem solved in the course. You do not need to prove your algorithm works.

Question 16 (15 marks)
You have been placed in charge of overseeing plywood distribution in the USSR. You are given a list of production facilities $F_1, \ldots, F_n$ and destinations $D_1, \ldots, D_m$. Each facility F_i can produce at most p_i units of plywood per day, and each destination D_j requires at least r_j units of plywood per day. The cost of moving one unit of plywood from facility F_i to destination D_j is $c_{i,j}$, and you may assume this scales linearly (so that e.g. the cost of moving half a unit is $c_{i,j}/2$). You wish to ensure that the requirements of each destination are met while spending as little on transportation as possible. Formulate this as a linear programming problem. (Your answer does not have to be in standard form, and you do not have to justify it.)

Question 17 (30 marks)

Solve <u>**two**</u> from the following four questions (15 marks each). **If you attempt more than two, you will not gain any extra credit, and only the first two questions attempted will be marked. If you do not wish one of your attempts to be marked, cross it out clearly along with all working.**

(a) The Travelling Salesman Problem (TSP) is defined as follows. We are given a list of n cities and, for each unordered pair $\{i, j\}$ of distinct cities, the cost $c(i, j)$ of travelling between i and j. (These costs can be arbitrary positive integers.) We are also given an integer k. We must output `Yes` if there is a way to travel to each city **exactly once**, then return back to the starting point, with total cost at most k. We call this a *round trip*. Otherwise, we must output `No`.

As an example, the input may be the set of cities $\{\text{Amsterdam}, \text{Baghdad}, \text{Cairo}, \text{Dublin}\}$, the cost function

$$\begin{aligned} c(\text{Amsterdam}, \text{Baghdad}) &= 175, & c(\text{Amsterdam}, \text{Cairo}) &= 95, \\ c(\text{Amsterdam}, \text{Dublin}) &= 24, & c(\text{Baghdad}, \text{Cairo}) &= 140, \\ c(\text{Baghdad}, \text{Dublin}) &= 250, & c(\text{Cairo}, \text{Dublin}) &= 122, \end{aligned}$$

and the integer $k = 500$. The round trip from Amsterdam to Baghdad to Cairo to Dublin and back to Amsterdam costs $175 + 140 + 122 + 24 = 461$. So there is a round trip with cost at most 500, and the desired output is `Yes`.

Prove that the Travelling Salesman Problem is NP-complete (under Karp reductions). You may use the fact that the problem HC of deciding whether or not a graph contains a Hamilton cycle is NP-complete.

(b) Consider the following greedy algorithm to find a large independent set in a graph.

Algorithm: GREEDYIS(G, k)

Input : A graph G.
Output: An independent set of G.

begin
  Let `output` $\leftarrow \emptyset$.
  foreach $v \in V(G)$ **do**
    if `output` $\cup \{v\}$ *is an independent set* **then**
      `output` $\leftarrow$ `output` $\cup \{v\}$.
  Return `output`.

Fix an integer $\Delta > 0$. If G has n vertices and maximum degree Δ, then how large an independent set is GREEDYIS **guaranteed** to output? Your answer should include both an upper bound and a lower bound, and a brief justification of each.

(c) Consider the following alternative greedy algorithm to

Algorithm: BETTERGREEDYIS(G, k)

Input : A graph G.
Output: An independent set of G.

begin
  Sort $V(G)$ in increasing order of degree. Write $V(G) = \{v_1, \ldots, v_n\}$, with $d(v_1) \leq \cdots \leq d(v_n)$. **Let** `output` $\leftarrow \emptyset$.
  for $i = 1$ *to* n **do**
    if `output` $\cup \{v_i\}$ *is an independent set* **then**
      `output` $\leftarrow$ `output` $\cup \{v_i\}$.
  Return `output`.

Prove that BETTERGREEDYIS may still output an independent set of size $O(1)$ even if G contains an independent set of size $\Omega(n)$.

Question 18 (15 marks)

Choose <u>**one**</u> of the three following problems to solve. **If you attempt more than one, you will not gain any extra credit, and only the first question attempted will be marked. If you do not wish one of your attempts to be marked, cross it out clearly along with all working.**

(a) Consider the following problem, which we call Approx-SAT. The input is a logical formula F in conjunctive normal form. The desired output is `Yes` if there is an assignment of truth values to variables which satisfies at least two thirds of F's OR clauses, and `No` otherwise. Prove that Approx-SAT is NP-complete under Karp reductions.

(b) Let $G = (V, E)$ be a bipartite graph with bipartition (A, B). Let a and b be positive integers, and suppose that every vertex in A has degree a and every vertex in B has degree b. What is the average degree $\frac{1}{|V|}\sum_{v \in V} d(v)$ of G's vertices, in terms of a and b? (Note that your answer should **not** depend on $|A|$ or $|B|$.)

(c) Let $G = (V, E)$ be a graph. A *dominating set* in G is a subset $X \subseteq V$ such that for all $v \in V \setminus X$, we have $\{x, v\} \in E$ for some $x \in X$. In other words, every vertex of G is either contained in X or joined to X by an edge (or both). *DS* is the problem which asks: Given a graph G and an integer k, does G contain a dominating set of size at most k? Give a Karp reduction from VC to DS and briefly explain why it works. (**Hint:** Among other things, you will need to insert a vertex into the middle of each of G's edges.)

END OF PAPER