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 nineteen 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)

(Short question.) Which of the following options correctly finishes the following sentence? "We say $f(n) \in O(g(n))$ if there exist $C, n_0 > 0$ such that..."

A. $f(n_0) \leq Cg(n_0)$.

B. $f(n) \geq Cg(n)$ for all $n \geq n_0$.

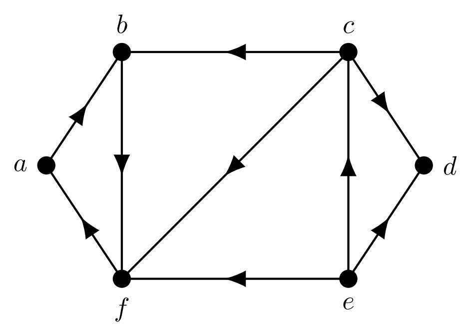
C. $f(n) \leq Cg(n)$ for all $n \geq n_0$.

D. $f(n) \leq Cg(n)$ for some $n \geq n_0$.

E. None of the above, or more than one of the above.

Question 2 (5 marks)

(Short question.) Consider the directed graph G given below.



(a) What is the in-degree of f? (1 mark)

(b) What is the out-neighbourhood of d? (1 mark)

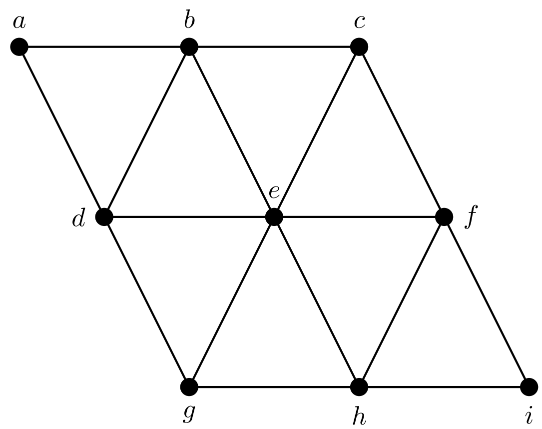
(c) Is G weakly connected? (1 mark)

(d) Is G strongly connected? (1 mark)

(e) Does G contain an Euler walk? (1 mark)

Question 3 (5 marks)

(Short question.) Consider the graph G below.



Is each of the following subsets of $V(G)$ an independent set, a vertex cover, both, or neither?

(a) $\{e\}$. (1 mark)

(b) $\{a, b, c, d, e, f, g, h, i\}$. (1 mark)

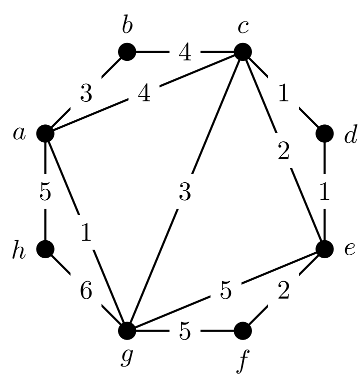
(c) $\{a, e, i\}$. (1 mark)

(d) $\{a, c, d, h\}$. (1 mark)

(e) $\{a, c, d, e, h, i\}$. (1 mark)

Question 4 (5 marks)

(Short question.) Suppose we run Prim's algorithm on the weighted graph below with initial vertex a. List the edges of the output minimum spanning tree in the order they are added by the algorithm. You may assume the algorithm breaks ties between edges any way you like.



Question 5 (5 marks)

(Short question.) Which of the following options correctly fills the blank in the following two sentences? "The greedy interval scheduling algorithm covered in lectures works by gradually building up a compatible set X. At each stage, the algorithm adds an interval to X which _____ subject to the new set X still being compatible."

A. "has a finish time as early as possible".

B. "has a start time as early as possible".

C. "intersects as few intervals in X as possible".

D. "is as short as possible".

E. None of the above.

Question 6 (5 marks)

(Short question.) Suppose that $G = ((V, E), w)$ is directed weighted graph, and let $s \in V$. Suppose that we are running Dijkstra's algorithm to find the distance $d(s, v)$ from s to v for all $v \in V$. Suppose that we have already found a set $X \subset V$ such that for all $x \in X$, the distance $d(s, x)$ from s to x is known. In order to increase the size of X, Dijkstra's algorithm finds an edge from a vertex u inside X to a vertex v outside X with a certain property that makes the algorithm work. What is that property?

A. $d(s, u) \cdot w(u, v)$ is as small as possible.

B. $w(u, v)$ is as small as possible.

C. v's degree is as small as possible.

D. $d(s, u) + w(u, v)$ is as small as possible.

E. None of the above.

Question 7 (5 marks)

(Short question.) In lectures, we discussed two main versions of the union-find data structure, both of which could efficiently handle long sequences of operations. The optimised version of the data structure dynamically flattened tree components in `FindSet` and `Union` operations, while the unoptimised version did not do this.

(a) What is the worst-case running time of the **unoptimised** version on a sequence of n operations? (3 marks)

(b) What is the worst-case running time of the **optimised** version on a sequence of n operations? (2 marks)

Question 8 (5 marks)

(Short question.) Which of the following algorithms would be most appropriate to find a shortest path between two given vertices in a **directed** graph with **both** negative-weight edges **and** negative-weight cycles?

A. Depth-first search.

B. Breadth-first search.

C. Dijkstra's algorithm.

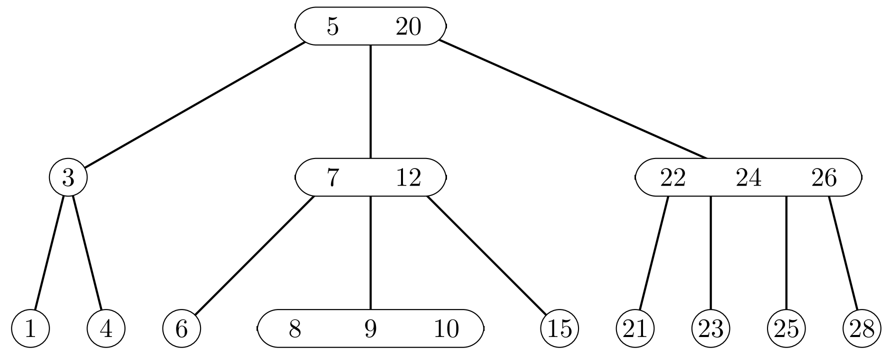
D. The Bellman-Ford algorithm.

E. None of the above, or more than one of the above.

Question 9 (5 marks)

(Short question.) If the below 2-3-4 tree were expressed as a red-black tree under the equivalence discussed in lectures, how many **red** nodes would it have in total?

(**Hint:** You don't need to fully convert the tree to answer this question.)



Question 10 (5 marks)

(Short question.) Mark each of the following statements true or false.

(a) A linear programming problem with variables x and y could not have $x^2 + y^2$ as an objective function. (1 mark)

(b) The linear programming problem $x + y \to \max$ subject to $y \leq 2$, $x \geq 1$ and $x, y \geq 0$ is unbounded. (1 mark)

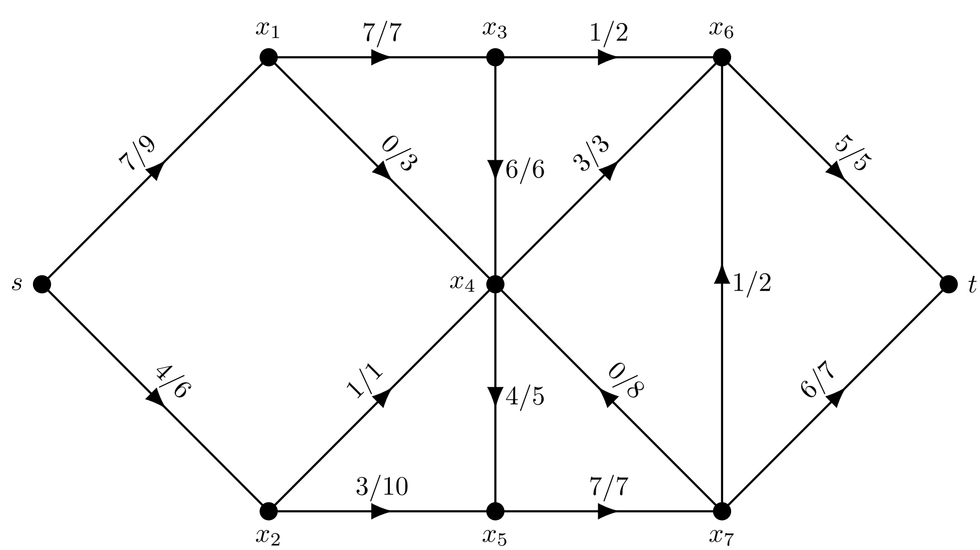
(c) The simplex algorithm always terminates in polynomial time. (1 mark)

(d) Every linear programming problem has at least one feasible solution. (1 mark)

(e) Every linear programming problem can be expressed in standard form. (1 mark)

Question 11 (10 marks)

(Parts (a)–(e) are short questions, parts (f) and (g) are medium questions.) Consider the flow network (G, c, s, t) and the flow f shown below.



(cont.)

(a) What is $c(x_7, x_4)$? (1 mark)

(b) What is $f^-(x_6)$? (1 mark)

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

(d) Is $(\{x_1, x_3, x_6\}, \{s, x_2, x_4, x_5, x_6, x_7, t\})$ a valid cut? (1 mark)

(e) Is $(\{s, x_1, x_6\}, \{x_2, x_3, x_4, x_5, x_7, t\})$ a valid cut? (1 mark)

(f) Find an augmenting path P for f which the Edmonds-Karp algorithm might push flow down. (**Hint:** What's the difference between Edmonds-Karp and Ford-Fulkerson?) (3 marks)

(g) Show the result of pushing flow down P as in the Edmonds-Karp or Ford-Fulkerson algorithms. In other words, writing F for the new flow that results from pushing flow down P, give the value of $F(e)$ for all edges e such that $F(e) \neq f(e)$. (2 marks)

Question 12 (5 marks)

(Short question.) What does it mean for a rooted tree T to be a DFS tree of a **connected** graph G?

A. $V(T) = V(G)$ and every edge in G is either between two adjacent layers of T or inside a layer of T.

B. $V(T) = V(G)$ and for all vertices $u, v \in V(G)$, the path uTv has length $d(u, v)$.

C. $V(T) = V(G)$ and T has $|V(G)| - 1$ edges.

D. $V(T) = V(G)$ and every edge in G that's not an edge in T is between a vertex and its ancestor in T.

E. None of the above, or more than one of the above.

Question 13 (5 marks)

(Medium question.) For each of the following statements, say whether it is certain to be true, conjectured to be true, conjectured to be false, or certain to be false. Here "Statement S is conjectured to be true/false" means that there is some conjecture discussed in the unit, such as P $\neq$ NP, which if correct would imply that S is true/false.

(a) 3-SAT is in Co-NP. (1 mark)

(b) If X reduces to Y under Karp reductions, then X reduces to Y under Cook reductions. (1 mark)

(c) Consider the decision problem with input (G, k), where G is a bipartite graph and k is a positive integer, where the desired output is `Yes` if G contains a matching of size at least k and `No` otherwise. This decision problem is in Co-NP. (1 mark)

(d) Any algorithm for an NP-complete problem (under Karp reductions) runs in exponential time or worse. (1 mark)

(e) If a problem is NP-hard under Cook reductions, then it is in NP. (1 mark)

Question 14 (5 marks)

(Medium question.) Draw a weighted undirected graph with **no** minimum spanning tree. You do not need to justify your answer.

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 15 (5 marks)
Let (G, c, s, t) be a flow network. List the conditions that must be fulfilled for a function $f\colon E(G) \to \mathbb{R}$ to be a valid flow according to the definition given in lectures. (No further information is required.)

Question 16 (10 marks)
A *facet* of a polyhedron is the line along which two faces meet. For example, a cube has twelve facets, four along each square face. The *elongated square gyrobicupola* is a deeply horrible polyhedron with 26 faces, 8 of which are triangles and 18 of which are squares. How many facets does it have? (**Hint:** Apply the handshaking lemma to a suitable graph. Partial credit will be given if you show your working.)

Question 17 (15 marks)
Recall that the Fibonacci sequence is defined by $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$. Prove that the number of independent sets in a length-k path is given by F_{k+3}. (**Hint:** Recall that the empty set is an independent set.)

Question 18 (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) Let t be a natural number. Let G be an n-vertex graph with $2t$ vertices of odd degree and $n - 2t$ vertices of even degree. Prove by induction or t or otherwise that we can write $E(G)$ as a **disjoint** union of edge sets

$$E(G) = E(P_1) \cup \cdots \cup E(P_t) \cup E(H),$$

where $P_1, \ldots, P_t$ are paths in G and H is a graph whose vertices all have even degree.

(b) You are a university librarian trying to maintain your service through budget cuts. You have a wide range of subject areas, and there are many good journals in each one. You want to maintain access to at least k good journals in each subject area while spending as little as possible. Unfortunately, journal companies are predatory and do not allow you to buy only the journals you want. Instead, they sell "bundles" of journals and require universities to buy the entire bundle, even if they only want a single journal in that bundle. Also, some journals are good in multiple subject areas.

Formally, the JournalBuying problem is defined as follows. You are given a set S of subject areas, a set J of journals, and a positive integer k. For each subject $s \in S$, you are given a set $J_s \subseteq J$ of interesting journals; these sets need not be disjoint, and there may be sets in J not included in any J_s. You are also given a collection of bundles $B_1, \dots, B_t \subseteq J$, and a non-negative integer price p_i for each bundle B_i. These bundles also need not be disjoint. To solve this problem, you must output a collection X of bundles satisfying the following two properties:

(i) For all $s \in S$, $|J_s \cap \bigcup_{B \in X} B| \geq k$.

(ii) Subject to (i), $\sum_{B \in X} p_B$ is as small as possible.

Show that the JournalBuying problem is NP-hard.

(c) A *Hamilton path* in a graph G is a path containing every vertex in the graph exactly once — for example, a Hamilton cycle minus an edge is a Hamilton path. It is NP-hard to decide whether or not a graph has a Hamilton path. However, the naive algorithm of checking every possible path takes $\Omega(n!)$ time, and we can do better than this. Using dynamic programming, give an algorithm $\textsc{HamPath}(G, v)$ with running time $O(n2^n)$ to decide whether a graph G has a Hamilton path starting at a given vertex $v \in V(G)$, and give a brief explanation of why it works. (You don't have to write out any code or even pseudocode — writing the recurrence relation that $\textsc{HamPath}$ is based on is good enough.)

(d) Suppose we are given a weighted graph G. We say a spanning tree is an *EST* (Even-weight Spanning Tree) if the total weight of its edges is even, and an *MEST* (Minimum Even-weight Spanning Tree) if it is an EST whose total weight is as small as possible (among ESTs). Suppose that the subgraph H of G composed of even-weight edges is connected, so that we can find an EST by running Kruskal's algorithm on H — in this case, we say G is *evenly connected*. However, this EST might not be an MEST — perhaps all MESTs contain pairs of odd edges. Motivated by this, we decide to try the following variant $\textsc{EvenKruskal}$ of Kruskal's algorithm.

As short-hand, given a tree T, we say edge $e \in E(G)$ is *viable for* T if $e \notin E(T)$ and $T + e$ contains no cycles. Similarly, we say a pair of distinct edges $e, f \in E(G)$ is *viable for* T if $e, f \notin E(T)$ and $T + e + f$ contains no cycles.

```
Algorithm: EvenKruskal
Let T be the graph with V(T) = V(G) and E(T) = ∅.
while T is not a spanning tree of G do
    Let T_next = T.
    if there is a viable even-weight edge for T then
        Let e be a viable even-weight edge whose weight is as small as possible.
        Update T_next ← T + e.
        Let w_even ← w(e).
    else
        Let w_even → ∞.
    if there is a viable pair of odd-weight edges for T then
        Let f, g be a viable pair of odd-weight edges whose total weight is as small as possible.
        Let w_odd ← (w(f) + w(g))/2.
        if w_odd < w_even then
            Update T_next ← T + f + g.
    Update T ← T_next.
Return T.
```

Give a counterexample to show that EvenKruskal need not output an MEST when given an evenly-connected graph as input, and give a brief explanation of why your counterexample works.

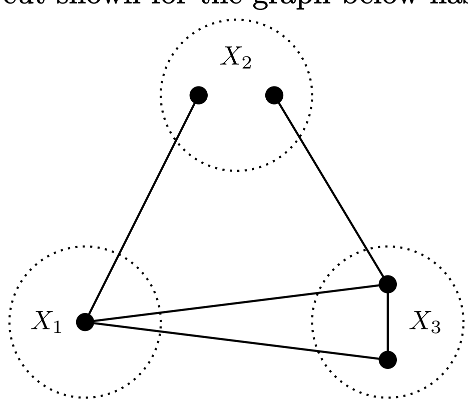
Question 19 (15 marks)

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) The decision problem of k-way min-cut is defined as follows. The input is an undirected graph $G = (V, E)$ and two positive integers k and a with $k \leq |V|$. A *k-way cut* is a list of sets $X_1, \ldots, X_k \subseteq V(G)$, and the *value* of a k-way cut is the total edge weight between sets, so that

$$\text{value}(X_1, \ldots, X_k) = \sum_{i \neq j} |\{\{u, v\} \in E \colon u \in X_i, v \in X_j\}|$$

For example, the 3-way cut shown for the graph below has value 4.



The desired output is `Yes` if G contains a k-way cut of value at most a, and `No` otherwise.

Meanwhile, the decision problem of integer linear programming takes as input an integer b and a linear programming problem which need not be in standard form. The desired output is `Yes` if the problem has a feasible solution which yields objective function value at most b **and** such that every variable in the solution has an integer value. For example, a valid instance might take $b = 2$ and take the linear programming problem to be

$$\begin{aligned} x + y - z &\to \text{min, subject to} \\ x &\geq 1, \\ 3x - y + z &= 4, \\ 2x + y - z &\leq -3, \\ x, y, z &\in \mathbb{N}. \end{aligned}$$

Give a Karp reduction from k-way min-cut to integer linear programming and **briefly** explain why it works. (**Hint:** One approach has k variables per vertex and one variable per edge.) (15 marks)

(b) The *diameter* of a graph is the largest distance between any pair of vertices. Recall that if G is a graph then the *complement* of G, written G^c, is the graph with vertex set $V(G)$ and edge set $\{\{x,y\} \colon x, y \in V(G), x \neq y\} \setminus E(G)$ — that is, the graph formed by replacing all edges of G with non-edges and all non-edges with edges. Show that if G is a graph with diameter at least 4, then G^c has diameter at most 2.

(c) Suppose that a connected n-vertex, m-edge graph G (given in adjacency-list form) contains two vertices x and y which are distance $d(x,y) \geq n - c$ apart, for some integer $c \geq 1$. Give an algorithm which, given G, x, and an integer k as input, decides whether or not G contains a k-vertex clique in time $O(m + n + c^k)$. Briefly explain why it works and why it runs in the claimed time. (You do not need to give full pseudocode — an informal description is fine.)

Hint: You may find it helpful to use a BFS tree.

END OF PAPER