

Question 1. [20 MARKS]

Short answer questions; no explanation required; all parts are worth 2 marks apiece; feel free to circle the correct answer.

- a. *True or false: A maximum matching in a bipartite graph is unique.*

False. Counterexample: $K_{2,2}$ (the graph with vertices $1, 2, a, b$ and edges $(1, a), (1, b), (2, a), (2, b)$). Demonstrate two distinct max matchings.

- b. *True or false: No decision problem in \mathbf{NP} is known to be solvable in polynomial time.*

False. $P \subseteq NP$.

- c. *True or false: If $L \in \mathbf{P}$ and $M \xrightarrow{\text{poly}} L$, then $M \in \mathbf{P}$.*

True. Polytime algorithm for M : To decide if $x \in M$ or not, consider the problem of deciding if $f(x) \in L$. It is equivalent to the original one. Computing of $f(x)$ with subsequent solution of the decision problem for $f(x)$ is polytime in size of x .

- d. *True or false: If $L \xrightarrow{\text{poly}} \text{VERTEX-COVER}$, then L is NP-hard.*

False. VERTEX-COVER is NP-hard, so every problem in \mathbf{NP} is polytime reducible to it, including those which are *not* NP-hard themselves.

- e. *True or false: Residual capacity of a path is never negative.*

True (by definition).

- f. *True or false: If $\text{HAM-CYCLE} \in \mathbf{P}$, then $\mathbf{P} = \mathbf{NP}$.*

True. HAM-CYCLE is NP-complete.

- g. *True or false: If $\mathbf{P} = \mathbf{NP}$ then $\mathbf{NP}^c \subseteq \mathbf{P}$.*

True. $\mathbf{NP}^c \subseteq \mathbf{NP}$.

- h. *True or false: If $\mathbf{NP} \subseteq \mathbf{coNP}$, then $\mathbf{NP} = \mathbf{coNP}$.*

True. Assume that $\mathbf{NP} \subseteq \mathbf{coNP}$. Consider any language L in \mathbf{coNP} . Then $L^c \in \mathbf{NP}$, hence by the assumption $L^c \in \mathbf{coNP}$, meaning $L \in \mathbf{NP}$. So $\mathbf{coNP} \subseteq \mathbf{NP}$, and therefore $\mathbf{NP} = \mathbf{coNP}$.

- i. *True or false: If the capacities of all edges in a network are rational numbers, then the Ford-Fulkerson algorithm run on this network necessarily terminates.*

True. Multiplying all capacities by their common denominator we obtain an equivalent integer-capacities problem.

- j. *True or false: Consider a flow f in a network N . It is given that f has an augmenting path p_1 with residual capacity 4 and an augmenting path p_2 with residual capacity 6. Then $|f^*| \geq |f| + 10$, where f^* is a maximum flow in N .*

False. It is easy to provide a counterexample (p_1 and p_2 could have a common edge).

Question 2. [15 MARKS]

- a. Given an oriented graph G , let u and v be two vertices in G . Consider the following question: is there a path in G from u to v ? (The path should always go in the direction prescribed by the orientation of G .) Restate the question in terms of a problem in network flows. Explain how the network is built and how the solution to the path problem is derived from a solution to the flow problem.

Build a network N from G by setting $s = u$, $t = v$, and assigning capacity 1 to every edge.

The question: "Is there a non-zero flow in N "?

Variation: "Is there a flow of value 1 in N "?

The equivalence: If there is a path from u to v in G , then there is a simple path. Let $f(e) = 1$ for any edge e in this path, and $f(e) = 0$ for any edge e outside it. f is then a flow of value 1 in N . Conversely, if there is a flow, then ...

- b. Do the same for the following question: are there k paths from u to v with no edge common to two or more of them?

Construct the flow network in the same way as part (a).

Answer: "Is there a flow of value k in N "?

Variation: "Is there a flow of value at least k in N "?

Equivalence: As above, k edge-disjoint paths can each be given a flow of 1 to obtain a total flow of k . In the other direction, if we have a k -flow with capacity 1 on each edge, the Ford-Fulkerson algorithm run on this network will augment by exactly 1 unit of flow each iteration, so finds at least k augmenting paths. These need not be edge disjoint in themselves, but if path p_2 reuses edge e from path p_1 they must be going along e in different directions, so we can obtain edge disjoint paths from p_1 and p_2 by removing e from both and swapping the subpaths each take after e . This idea can be extended to deal with any number of common edges between any number of augmenting paths.

- c. Give the running time for the algorithm solving the problem in part (b). (Make sure to introduce the necessary notation).

The running time is $O(mk)$ where m is the number of edges in G . It takes time $O(m)$ to increase the value of a given flow by finding a path of positive residual capacity (discussed in class - BFS running time). By virtue of the definition of N , if a residual capacity of a path is non-zero, then it is 1. Therefore we'll have to perform k steps with the running time $O(m)$ each.

Question 3. [15 MARKS]

Recall that a Boolean formula is called a *tautology* if it is satisfied for every assignment of its variables. Consider the following decision problem D :

$$\begin{array}{l} \text{Input :} \quad \text{A Boolean formula } S. \\ \text{Output :} \quad \begin{cases} 0, & \text{if } S \text{ is tautology} \\ 1, & \text{otherwise} \end{cases} \end{array}$$

- a. Does D belong to the class \mathbf{NP} ? Provide justification.

Yes, a certificate y for a formula S is an unsatisfying assignment of variables for S . Plugging the variables in S and evaluation of S for these values is done in time linear in size of S .

- b. Prove that D is an NP-hard problem.

We know that $\text{SAT} \in \mathbf{NPc}$. Therefore it would be enough to demonstrate a reduction of SAT to D . We need to construct a function f that maps an instance S of the SAT problem to an instance $f(S)$ of the problem D , such that S is a satisfiable formula if and only if $f(S)$ is not a tautology. We define f by $f(S) = \bar{S}$. Then “ S is satisfiable” is equivalent to “ \bar{S} is not satisfied for at least one assignment of variables” which is equivalent to “ $f(S)$ is not a tautology”.

- c. If $D \in \mathbf{P}$ would that mean $\mathbf{P} = \mathbf{NP}$? Provide a full explanation.

Answer: yes. Parts a) and b) together mean by definition that D is NP-complete, in other words any problem in \mathbf{NP} is polytime reducible to D . Therefore, if D is in \mathbf{P} , then so is any problem from \mathbf{NP} . Hence $\mathbf{NP} \subseteq \mathbf{P}$. Since also $\mathbf{P} \subseteq \mathbf{NP}$, we get $\mathbf{P} = \mathbf{NP}$.

Total Marks = 50