CS60003 Algorithm Design and Analysis, Autumn 2009–10

End-Semester Test

Maximum marks: 50 | November 24, 2009 | Total time: 3 hours

Roll no: _____ **Name:** _____

[*Write your answers in the question paper itself. Be brief and precise. Answer <u>all</u> questions.*]

Guided Thinking Section

In this section, I guide you to arrive at solutions to some computational exercises.
Proceed exactly as I tell you to. Just fill out the missing details.

1. Let S be a finite set, and $S_1, S_2, \ldots, S_k$ a collection of subsets of S. A subcollection $S_{i_1}, S_{i_2}, \ldots, S_{i_l}$ with $1 \leqslant i_1 < i_2 < \cdots < i_l \leqslant k$ is called a *cover* of S if $S = \bigcup_{j=1}^{l} S_{i_j}$. In this case, l (the number of subsets in the cover) is called the *size* of the cover. The decision problem SET-COVER takes as input a set S, a collection $S_1, S_2, \ldots, S_k$ of subsets of S, and a positive integer l, and decides whether S has a cover (in the given subsets) of size exactly l. In this exercise, we prove that SET-COVER is an NP-Complete problem. You may assume any standard representation of sets (such as sorted/unsorted arrays, linked lists, or trees).

(a) What is the output of SET-COVER for the following input? $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ with five subsets $S_1 = \{2, 3, 5, 7\}$, $S_2 = \{1, 2, 3, 5, 8\}$, $S_3 = \{1, 2, 4, 8\}$, $S_4 = \{3, 6, 9\}$, and $S_5 = \{4, 6, 8, 9\}$, and $l = 2$. No **(1)**

(b) Show that SET-COVER $\in$ NP. For an instance $\langle S, (S_1, S_2, \ldots, S_k), l\rangle$ in Accept(SET-COVER), a certificate is: **(2)**

l distinct indices $i_1, i_2, \ldots, i_l$

This certificate can be verified in polynomial time as: **(2)**

check whether $S = \bigcup_{j=1}^{l} S_{i_j}$

(c) In order to prove the NP-hardness of SET-COVER, reduce VERTEX-COVER to SET-COVER. Let $\langle G, t\rangle$ be an input for VERTEX-COVER, where $G = (V, E)$ is an undirected graph with $n = |V|$ vertices and $e = |E|$ edges. The reduction algorithm produces an instance $\langle S, (S_1, S_2, \ldots, S_k), l\rangle$ for SET-COVER, where: **(4)**

S = E

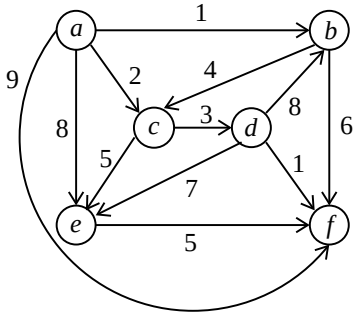
k = n

S_i = The set of edges incident upon the vertex v_i

l = t

Remark: Your reduction must fulfill the following requirement: S has a set cover of size l if and only if G has a vertex cover of size t.

2. Dijkstra's single-source-shortest-path algorithm is applied to the following graph with source a.



(a) Let us use the notations given in the notes. Fill out the following table to demonstrate how the partition P, Q and the arrays **D** and **prev** change in different iterations of Dijkstra's algorithm. Assume that these arrays are indexed by a, b, c, d, e, f from left to right. **(6)**

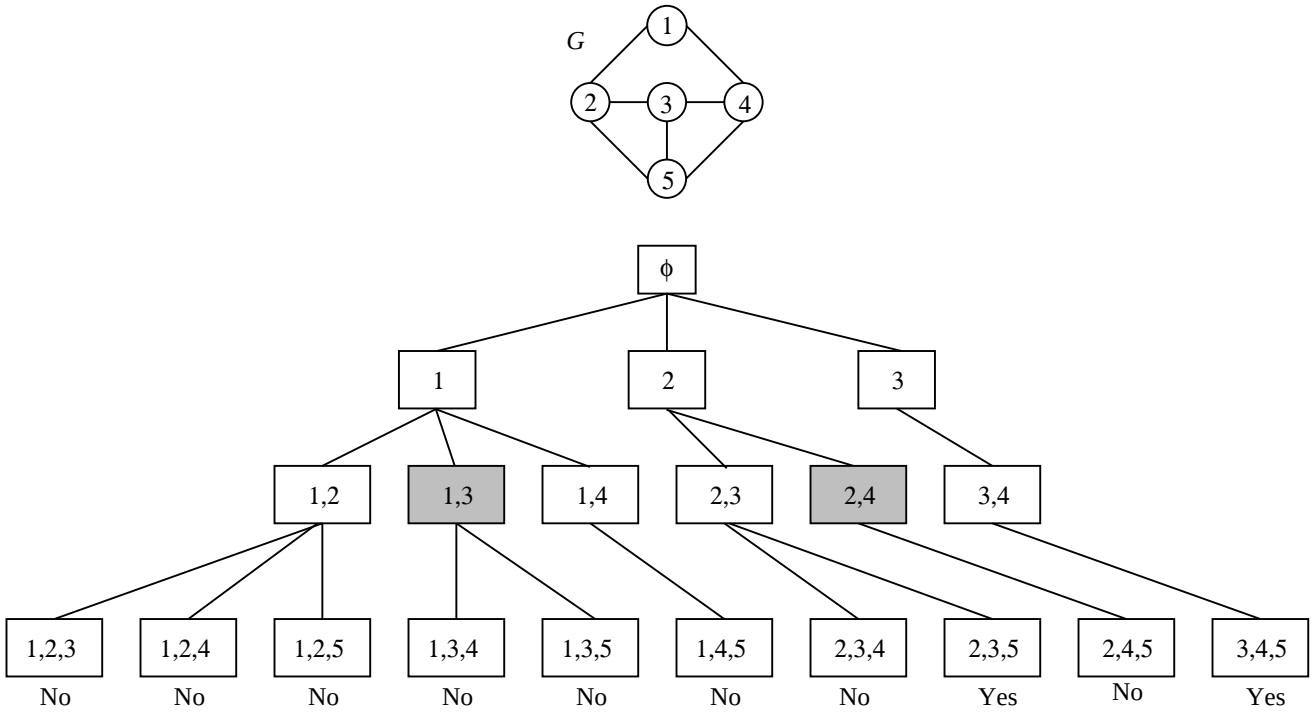
Iteration	P	Q	**D**	**prev**
Init	$\{a\}$	$\{b, c, d, e, f\}$	0 | 1 | 2 | ∞ | 8 | 9	a | a | a | a | a | a
1	$\{a, b\}$	$\{c, d, e, f\}$	0 | 1 | 2 | ∞ | 8 | 7	a | a | a | a | a | b
2	$\{a, b, c\}$	$\{d, e, f\}$	0 | 1 | 2 | 5 | 7 | 7	a | a | a | c | c | b
3	$\{a, b, c, d\}$	$\{e, f\}$	0 | 1 | 2 | 5 | 7 | 6	a | a | a | c | c | d
4	$\{a, b, c, d, f\}$	$\{e\}$	0 | 1 | 2 | 5 | 7 | 6	a | a | a | c | c | d
5	$\{a, b, c, d, e, f\}$	$\emptyset$	0 | 1 | 2 | 5 | 7 | 6	a | a | a | c | c | d

(b) Using the **prev** array, trace the shortest path from a to f. **(2)**

prev$(f) = d$, **prev**$(d) = c$, and **prev**$(c) = a$. So the shortest a, f path is a-c-d-f.

3. Consider the following non-deterministic algorithm for the CLIQUE problem. Let $G = (V, E)$ be an undirected graph with n vertices numbered as $1, 2, 3, \ldots, n$. We are required to find out whether G contains a clique of size t. We non-deterministically choose t vertices $v_1, v_2, \ldots, v_t$. In order to avoid repetitions, we choose the vertices in increasing order, that is, satisfying $1 \leqslant v_1 < v_2 < \cdots < v_t \leqslant n$.

(a) The following figure shows an incomplete non-deterministic computation tree for a graph G on 5 vertices and for $t = 3$. Complete the drawing. That is, draw the complete tree with each node labeled by appropriate non-deterministic choices and with leaf nodes marked additionally by Yes/No decisions. **(4)**



(b) Suppose that a backtracking algorithm is carried out on a computation tree for the above algorithm for CLIQUE. Describe a pruning strategy to identify appropriate intermediate nodes as dead ends. **(2)**

A node $v_1, v_2, \ldots, v_i$ of vertices not forming an i-clique. These vertices cannot belong to a bigger clique.

(c) Mark/state which non-leaf nodes in the tree of Part (a) are dead ends (for your pruning strategy). **(2)**

The nodes $1, 3$ and $2, 4$

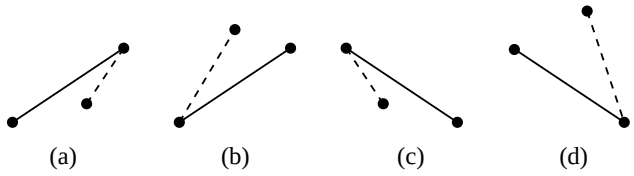
Independent Thinking Section

In this section, I supply you no guidelines.
You yourself are required to arrive at solutions to some computational exercises.

---

4. Let $P_1, P_2, \ldots, P_n$ be n points in the plane in general position. Denote by m_{ij} the slope of the segment P_iP_j. Supply an $\mathrm{O}(n \log n)$-time algorithm for identifying the pair of points P_i, P_j for which $|m_{i,j}|$ is maximum. (In this case, P_iP_j is the steepest among all the line segments connecting the given points.) **(5)**

Solution Sort the points in increasing order of x-coordinates. The steepest segment must belong to two consecutive points in the sorted list, as the following figure demonstrates.



The dashed segments are steeper than the solid segments

Sorting the points requires $\mathrm{O}(n \log n)$ time. Computing $n - 1$ slopes between consecutive points and finding the maximum of the absolute values of these slopes can be done in $\mathrm{O}(n)$ time.

5. Let IS-HAM-CYCLE denote the computational problem that, given an undirected graph G, decides whether G contains just those edges necessary to form a Hamiltonian cycle in G (no more, no less). Prove or disprove: IS-HAM-CYCLE is NP-Complete. **(5)**

Solution A graph G on n vertices is an n-cycle if and only if G is connected with each vertex having degree 2. Connectedness of a graph can be checked in polynomial time. Also, it is straightforward to check whether each vertex in a graph has degree 2. It follows that IS-HAM-CYCLE is in P and so cannot be NP-Complete unless $\mathrm{P} = \mathrm{NP}$.

6. Consider the optimization version of the set covering problem of Exercise 1. That is, given a finite set S and a collection of k subsets $S_1, S_2, \ldots, S_k$ of S, we intend to find out a cover of S (from the given collection) of size as small as possible. Let us denote this optimization problem by MIN-SET-COVER.

(a) For instance, take $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ with five subsets $S_1 = \{2, 3, 5, 7\}$, $S_2 = \{1, 2, 3, 5, 8\}$, $S_3 = \{1, 2, 4, 8\}$, $S_4 = \{3, 6, 9\}$, and $S_5 = \{4, 6, 8, 9\}$. What is an output of MIN-SET-COVER on this input? **(2)**

Solution Since all S_i are proper subsets of S, we cannot have a cover of size 1. It is also easy to check that no two of the given subsets have a union equal to S. However, $S_1 \cup S_3 \cup S_5 = S$, so an output for MIN-SET-COVER can be the smallest count 3 along with the explicit cover S_1, S_3, S_5.

Let $S = \{x_1, x_2, \ldots, x_n\}$ be of size n, and let f_i be the count of the subsets S_j containing the element x_i. Finally, let $f = \max(f_1, f_2, \ldots, f_n)$. (The counts f_i are the frequencies of the elements, and f is the maximum frequency.)

(b) Design a polynomial-time f-approximation algorithm for MIN-SET-COVER. **(6)**

Solution The MIN-SET-COVER problem is a generalization of the MIN-VERTEX-COVER problem. We can adapt the 2-approximation algorithm for MIN-VERTEX-COVER to work for set covers as follows. The algorithm assumes that all points in S are covered by at least one of the given subsets. It is easy to check whether this condition is satisfied for the given input, and if not, we return *failure* without running the following algorithm.

```
Initialize U = ∅, and C = {S_1, S_2, ..., S_k}.
while (S ≠ ∅) {
   Choose any x ∈ S.  /* An yet uncovered member of S */
   Add to U all the subsets S_i ∈ C containing x.
   Remove from S all the points covered by all these subsets.
   Finally, remove these subsets from C.
}
Return U.
```

(c) Prove that your algorithm achieves an approximation ratio of f. **(6)**

Solution Extend the notion of matching to the case of set covers. A subset $T \subseteq S$ is called a matching (with respect to the given collection $S_1, S_2, \ldots, S_k$) if no two members of T belong to a common subset S_i. The elements x chosen in all the iterations of the **`while`** loop constitute a matching of S, since all the elements sharing subsets with a choice of x are removed (covered) before a next choice of x is made.

Since a matching contains a set of *independent* elements (each requiring its own subset for getting covered), the size of any matching of S is no larger than the size of any cover of S. If we apply this observation to the matching T computed by the algorithm and to an optimal cover U^*, we get

$$|T| \leqslant |U^*|.$$

On the other hand, each choice of x in the **`while`** loop adds at most f subsets to the cover U, since this is the maximum frequency of an element. Therefore, the computed cover U satisfies

$$|U| \leqslant f \times |T|.$$

Combining these two inequalities gives an approximation ratio of f:

$$|U| \leqslant f \times |T| \leqslant f \times |U^*|, \text{ that is, } |U|/|U^*| \leqslant f.$$

(d) Prove or disprove: The approximation ratio f achieved by your algorithm is tight. **(6)**

Solution The approximation factor f is tight. To demonstrate this, take a value $f = 2^r$, and let S contain $r + 1$ elements $x_0, x_1, x_2, \ldots, x_r$. We start with the collection of all subsets of S containing x_0 (so $k = 2^r$). The frequency of x_0 is $2^r = f$, whereas, for $1 \leqslant i \leqslant r$, the frequency of x_i is $2^{r-1} = f/2$. Thus, the maximum frequency is f.

Suppose that the approximation algorithm chooses x_0 as the first uncovered element. Since x_0 belongs to every subset in the given collection, the algorithm adds all these subsets to U. This also covers all the members of S, and the algorithm stops. We thus get $|U| = 2^r = f$.

On the other hand, $|U^*| = 1$, since the given collection of subsets of S contains the entire set S itself.

Clue: $f = 2$ for the MIN-VERTEX-COVER problem.

ROUGH WORK

ROUGH WORK