Final Examination on Algorithms

Instructor: Biing-Feng Wang **June 15, 2004**

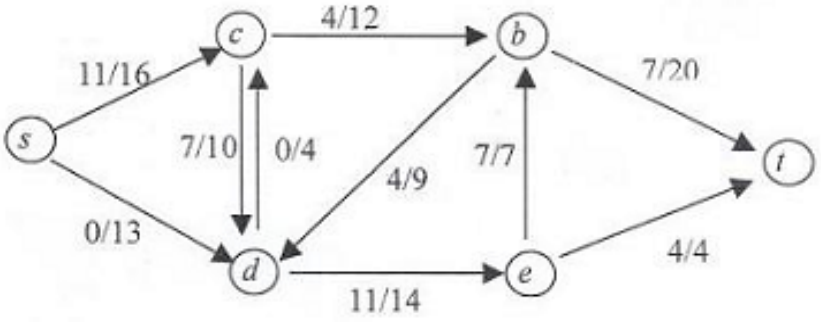
Remark: For each problem, you must justify your answers.

Problem 1: (10%) Find an upper bound on the recurrence $T(n)=T(\lfloor n/2 \rfloor)+T(\lceil n/2 \rceil)+1$ by using the substitution method. (You may assume that $T(1)=1$.)

Problem 2: (10%) Let $K=\{k_1, k_2, ..., k_n\}$ be a set of n keys. Give an $O(n)$-time algorithm to construct a binomial heap H for K?

Problem 3: (10%, Disjoint Sets) Suppose that disjoint-set forests are used to implement disjoint sets. Both *Union-by-rank* and *path compression techniques* are used. Write done the procedure FIND-SET(x) and UNION(x, y).

Problem 4: (10%, Maximum Flow) A flow network G and a flow f are given as follows. Show how Ford-Fulkerson's method finds the maximum flow from s to t, using f as the initial flow. You should describe the process step by step by giving illustrations.



Problem 5: (10%, Transitive Closure) Let A be the adjacency matrix of a directed graph $G=(V, E)$, where $A[i, j]=1$ if there is an edge (i, j). Note that $A[i, j]=0$ if $i=j$. Define $d_{i,j}^{(m)}$ as follows:

$$d_{i,j}^{(m)} = \begin{cases} 1 & \text{if there exists a path from } i \text{ to } j \text{ that contains at most } m \text{ edges, and} \\ 0 & \text{otherwise.} \end{cases}$$

(1) Give a recurrence of $d_{i,j}^{(m)}$.

(2) Based upon the above recurrence, design an $O(n^3 \log n)$ time algorithm for computing the transitive closure A^* of G, in which $A[i, j]=1$ if $i=j$ or there exists a path from i to j.

Problem 6: (10%, Number-Theoretic Algorithm) Describe an efficient algorithm to compute the value x^n for any given real x and nonnegative integer n.

Problem 7: (10%, Approximation Scheme) Consider the approximation scheme in the textbook for the subset sum problem. When two numbers $x<y$ in L_i satisfy $y \le (1+\delta)x$, we remove y, since x and y will "produce" very similar results. Can we modify the algorithm by removing x, instead of y? Please explain.

Problem 8: (10%, Minimum Spanning Trees) Show that the minimum spanning tree problem can be solved in $O(|E|+|V|\log|V|)$ time.

Problem 9: (10%, Design of Algorithms) Let $G=(V, E)$ be an undirected graph represented by adjacency lists. G may be connected or not. Design an efficient algorithm to determine whether G contains a cycle. Your algorithm should run in $O(|V|)$ time, independent of $|E|$.

Problem 10: (10%, NP-completeness) Suppose that a problem A can be reduced to another problem B in $O(n)$ time.

(a) (2%) If $A \in$ *NP-complete*, can we conclude that B is *NP-hard*?

(b) (2%) If $A \in$ *NP-complete*, can we conclude that B is *NP-complete*?

(c) (2%) If $B \in$ *NP-hard*, can we conclude that A is *NP-hard*?

(d) (2%) If $B \in$ *NP-complete*, can we conclude that A is *NP-hard*?

(e) (2%) If $B \in$ *NP-complete*, can we conclude that A is *NP-complete*?

Bonus: (10%) This problem is to verify whether you had done homework by yourself. Please answer either of the following. Proofs and analyses are unnecessary. (If you answer both, only the one getting less score will be counted.)

(a) This problem investigates the binary gcd algorithm, which avoids the remainder computations used in Euclid's algorithm. Design an efficient binary gcd algorithm for input integers a and b, where $a \ge b$, that runs in $O(\lg a)$ time. Assume that each subtraction, parity test, and halving can be performed in unit time.

(b) Suppose that we wish to maintain the transitive closure of a directed graph $G = (V, E)$ as we insert edges into E. Assume that the transitive closure is to be represented as a boolean matrix. Describe an efficient algorithm for updating the transitive closure as edges are inserted into the graph. For any sequence of n insertions, your algorithm should run in total time $\sum_{i=1}^{n} t_i = O(V^3)$, where t_i is the time to update the transitive closure when the ith edge is inserted.