Final Examination on Algorithms

Teacher: Biing-Feng Wang June 19, 2001

Problem 1: (15%, Selected problems from midterm examination-Part I)

(1)(8%) Assume that $f(n)=O(g(n))$. Using the definitions of O and Ω notations, prove or disprove that $g(n)=\Omega(f(n))$.

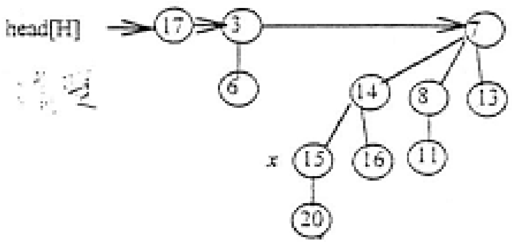
(2)(7%) Find an upper bound on the recurrence $T(n)=T(\lfloor 2n/5 \rfloor)+T(\lfloor 3n/5 \rfloor)+n$ by appealing to a recursive tree. (You may assume that $T(1)=1$.)

Problem 2: (15%, Selected problems from midterm examination-Part II)

(1)(7%) Let A=(1, 2, 3, 4, 5, 6, 7, 8). If we call *Build-Heap*(A) to make A an *max* heap, how many *exchange-operations*, each of which exchanges the contents of two elements, will be performed. Explanation is necessary.

(2)(8%, Dynamic Programming) The *resource allocation problem* is defined as follows. We are given m resources and n projects. A profit $P(i, j)$ will be obtained if i resources are allocated to project j, where $0 \le i \le m$ and $0 \le j \le n$. The problem is to find an allocation of resources to maximize the total profit. Define $A[x, y]$ as the maximum profit obtained by allocating x resources to the first y projects, where $0 \le x \le m$ and $0 \le y \le n$. Give a recurrence of $A[x, y]$, including all boundary conditions.

Problem 3: (10%, Mergeable heaps) Use the following binomial heap to explain the operation: Binomial-Heap-Delete(H, x). You must describe the operation in detail without calling any other operation as a procedure.



Problem 4: (15%, Elementary graph algorithms)

(1) (8%) Write a pseudo-code for DFS. The input is a graph G. The output includes arrays π, d, and f, where for every $v \in V(G)$, $\pi(v)$ records the parent of v, $d(v)$ records the time when v is discovered, and $f(v)$ records the time when the search finishes examining $adj[v]$, where $adj[v]$ denotes the set of vertices adjacent to v.

(2) (3%) What's the time complexity required for DFS when the input graph is represented by its adjacency lists? Justify your answer.

(3) (4%) What's the time complexity required for DFS when the input graph is represented by its adjacency matrix? Justify your answer.

Problem 5: (15%, Shortest paths) The following is Dijkstra's algorithm for the single-source shortest paths problem.

```
Dijkstra(G, w, s)
Initialize-Single-Source(G, s)
S←∅
Q←V[G]
while Q≠∅ do
      u←Extract-Min(Q)
      S←S∪{u}
      for each v∈Adj[u] do
            Relax(u, v, w)
```

(1) (3%) What does Initialize-Single-Source(G, s) do?

(2) (6%) Show that the above algorithm can be implemented in $O(E\log V)$ time.

(3) (6%) Show that the above algorithm can be implemented in $O(V^2)$ time.

(Detailed explanation is necessary and you can not use Fibonacci heaps.)

Problem 6: (10%, Convex hull) Let $Q=\{p_1, p_2, \ldots, p_n\}$ be a set of $n>3$ points. Give an efficient algorithm to compute the convex hull of Q_i for every i, where $3\le i\le n$ and $Q_i=\{p_1, p_2, \ldots, p_i\}$. Your algorithm must runs in $o(n^2\log n)$ time.

(Hint: In Graham's scan, we usually select the vertex having minimum y-coordinate as p_0. As we had indicated, with some modification, any point within the convex hull can be selected. You can use this fact without proving.)

Problem 7: (10%, NP-completeness)

(1) (7%) Define the following terms: P, NP, non-deterministic algorithms

(2) (3%) Draw a Venn diagram for NP and NP-hard

Problem 8: (10%, The subset sum problem) Let $S=\{s_1, s_2, \ldots, s_n\}$ be a set of n positive integers and $t>0$ be an integer. The subset-sum problem is to find a subset of S that adds up exactly to t. Suppose that $t=O(n^{2.5})$. How fast can we solve the subset-sum problem? Describe your algorithm in detail.

$t = s_1 + s_2 + s_3$

Bonus: (10%, A problem selected from homework) Let $G=(V, E)$ be an undirected, connected graph with weigh function $w: E\rightarrow \mathbf{R}$, and suppose that $|E|\ge|V|$. Let T be a minimum spanning tree of G. Prove or disprove that there exist edges $(u, v)\in T$ and $(x, y)\notin T$ such that $T-\{(u, v)\}\cup\{(x, y)\}$ is a second-best minimum spanning tree of G.