Graduate Algorithms	Manuel Blum, Virginia Vassilevska, Chris Wang
Problem Set 1 (Last Modified Date: January 26, 2005)	Due: January 28, 2005

You can work in groups of up to 3. Please let us know, for each question, if you have seen the question before. You will be graded out of 100 points for this assignment. Unless otherwise specified, in all graph problem m is the number of edges and n is the number of vertices of a graph. Please submit your solution in ALL of the following formats, both by email to `virgi+algo@cs.cmu.edu` and `chengwen+gradalg@cs.cmu.edu` and as a hardcopy in class: ps, pdf and tex.

Problem 1 [10pts] Prove the following property of the DFS finishing order:

Let $f(u)$ be the finishing time of vertex $u \in V$ given by DFS. For each strongly connected component C define $f(C) = \max_{u \in C} f(u)$. Then if there is a directed path from C to C' then $f(C) \geq f(C')$.

Use only the following properties of the DFS order and spanning forest:

A: Suppose v is a descendant (in the DFS forest) of u. Then $f(u) \geq f(v)$ and if w is another vertex such that $f(u) \geq f(w) \geq f(v)$, then w is also a descendant of u.

B: If (x, y) is an edge in the graph, and $f(x) \leq f(y)$, then x is a descendant of y.

Conclude that (if A and B hold) the algorithm described in class finds the strongly connected components in any directed graph.

Problem 2 [10pts] Given a tree $T = (V, E)$ rooted at a node r, a node u is said to be an *ancestor* of v if u lies on the path from v to the root r. Describe an algorithm which preprocesses T in time $O(|V| + |E|)$ so that a query of the form *"Is u an ancestor of v?"* can be answered in $O(1)$ time.

Problem 3 A large class of two-player games can be descibed as follows. There is a directed graph G with n vertices and m edges. The vertices represent game positions and the edges represent possible moves. Initially some vertex s is marked. The players alternate. In turn, each player moves the mark to one of the adjacent vertices. If there are no adjacent vertices, the player loses. Suppose both players use optimal strategies. There are three possible outcomes: the first player wins, the second player wins, or the game runs forever. Does an efficient (polynomial) algorithm for determining the outcome of the game exist? If so, give your best algorithm.

```
Scoring:
30pts the problem is undecidable
20pts no polytime algorithm exists unless P=NP
5pts for a correct polynomial algorithm
10pts for O(n+mlog n)
15pts for O(m+nlog n)
20pts for O(m+n)
30pts for O(n)
```

Problem 4 [10pts] Let T be a binary search tree with height indicators located at each node. Denote by T_1 and T_2 the subtrees rooted at the children of the root of T. Suppose T_1 and T_2 are height-balanced trees of height h_1 and h_2, respectively. Give an $O(|h_1 - h_2|)$ algorithm to balance the whole tree without increasing its height.

Problem 5

(a) Give an efficient algorithm for finding a cycle in a directed graph, if a cycle exists, or to determine that the graph is acyclic.

```
Scoring:
2pts for a correct polynomial algorithm
4pts for O((m+n) * n)
6pts for O(n^2)
8pts for O(m+n)
18pts for O(n)
```

(b) A triangle in a directed graph $G = (V, E)$ is given by three *distinct* vertices $v_0, v_1, v_2 \in V$, so that $(v_0, v_1), (v_1, v_2), (v_2, v_0) \in E$. Give an efficient algorithm to find a triangle in a directed graph, if one exists, or determine that the graph is triangle-free.

```
Scoring:
1pt a correct polynomial algorithm
2pts for O(n^3)
10pts for o(n^3)
15pts for O(n^2)
20pts fir O(n + m)
```

Problem 6 Give an efficient algorithm to compute the total number of paths in a DAG (including paths of length 0). You can assume arithmetic operations with $O(\log n)$ bit integers take $O(\log n)$ time.

```
Scoring:
5pts for a correct polynomial algorithm
10pts for O(mn)
15pts for O(m+nlog n)
30pts for O(m+n)
```

Problem 7 Let G be a directed graph with edges labeled by elements from a totally ordered set. s and t are two vertices of the graph. Give an efficient algorithm to determine whether there is a path from s to t such that the labels on the path edges form a nondecreasing sequence. What hint would you give for this problem?

```
Scoring:
5pts for a good hint
and/or
5pts for a correct polynomial algorithm
```

10pts for O(n^3)
15pts for O(n+m log n)
20pts for O(m+n log n)
25pts for O(m+n)