Basic Algorithms (V22.0310); Fall 2005; Yap

MIDTERM (with SOLUTION)

**Oct 17, 2005**

# Instructions

- You are allowed one 8"x11" sheet of notes.
- Do all quesions. Start each question on its own page.
- Read carefully, think deeply and write sparsely.
- Only write on the front side of each page – use the other side for scratch.
- Show your working for partial credit.
- Reread your answers, and rewrite!

# Question 1

(Logic and Planar Graphs, 10 points)
The following statement is a fact: *a planar graph on $n$ vertices has at most $3n-6$ edges.* Let us restate it as follows:

$$(G \text{ is a planar graph and has } n \text{ vertices}) \Rightarrow (G \text{ has } \leq 3n-6 \text{ edges}).$$

**(i)** State the contra-positive of this statement.

**(ii)** The complete graph on 5 vertices, denoted by $K_5$ is shown in Figure 1. Using the contra-positive statement in part (i), prove that $K_5$ is not planar.

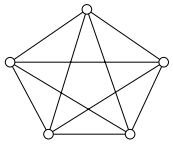

Figure 1: $K_5$, the complete graph on 5 vertices

SOLUTION:
(i) Constra-positive:

$$(G \text{ has } > 3n-6 \text{ edges}) \Rightarrow (G \text{ is not a planar graph or does not have } n \text{ vertices}).$$

(ii) Let $n = 5$. Then $K_5$ has $> 3n - 6 = 9$ edges. By the contra-positive of (i), we conclude that $K_5$ is not a planar graph or does not have 5 vertices. Since it does have 5 vertices, it must not be planar.

# Question 2

(Proving Big-Oh Relations, 10 points)
Let $f(x) = \sin x$ and $g(x) = 1$.
**(i)** Prove $f \preceq g$ or its negation.

**(ii)** Prove $g \preceq f$ or its negation.

REMEMBER: To prove $f \preceq g$, you need to show the existence of certain constants $C > 0$ and $x_0 > 0$ will satisfy a certain property involving $f$ and $g$. To prove that $f \not\preceq g$, you need to show that for *all* choices of $C > 0$ and $x_0 > 0$, this property fail.

SOLUTION:

(i) CLAIM: $f \preceq g$. Choose $C = 1$. Then for all $x \in \mathbb{R}$, we have $f(x) = \sin x \leq 1 = g(x)$. So any choice of $x_0$ will do.

(ii) CLAIM: $g \not\preceq f$. To see this, note that for all $C > 0$ and $x_0$, there exists $x > x_0$ such that $f(x) = 0$. Hence $g(x) \leq Cf(x)$ does not hold.

# Question 3

(Big-Oh Ordering of Functions, 15 points)

Order these in increasing big-Oh order:

$$n \lg n, \quad n^{-1}, \quad \lg n, \quad n^{\lg n}, \quad 10n + n^{3/2}, \quad 2^n, \quad 2^{\lg n}.$$

SOLUTION:

$$n^{-1}, \quad \lg n, \quad 2^{\lg n}, \quad n \lg n, \quad 10n + n^{3/2}, \quad n^{\lg n}, \quad 2^n.$$

Common Mistake: Most students did not realize that $2^{\lg n} = n$.

# Question 4

(Graph Search, 15 points)

Let $G$ be a digraph. A **global sink** is a node $u$ such that for every node $v \in V$, there is path from $v$ to $u$. A **global source** is a node $u$ such that for every node $v \in V$, there is path from $u$ to $v$.

**(i)** Assume $G$ is a DAG. Give an algorithm to detect if $G$ has a global sink and/or a global source. Your algorithm returns YES if both exists, and returns NO otherwise.

**(ii)** Does your algorithm work if $G$ is not a DAG? If not, give a counter example which makes your algorithm fail.

SOLUTION:

We would have liked you to use BFS or DFS, not roll your own because you tend to end up with $O(n^2)$ algorithms otherwise.

(i) The correctness of our algorithm CRUCIALLY depends on the fact that the input is DAG. In this case, if there is a global source, then there is a unique node $u$ with indegree 0. If there is a global sink, then there is a unique node $v$ with outdegree 0.

Use any graph traversal algorithm (BFS or DFS) to go through the graph. Keep two arrays, $Indegree[1..n]$ and $Outdegree[1..n]$, both initialized to 0. Each time you see an edge $(u, v)$, you increment $Indegree[v]$ and increment $Outdegree[u]$. You must use a Driver Program (as in Homework 3) to ensure that all edges are visited. At the end, you check: if there is a unique vertex $u$ such that $Indegree[u] = 0$, AND a unique vertex $v$ such that $Outdegree[v] = 0$, then output YES. Else output NO.

(ii) This algorithm would fail for the following non-DAG $G$ which consists of a cycle $(2, 3, \ldots, n, 2)$ and the node 1 is isolated.

COMMON MISTAKES:

Many student use this wrong criterion for a $u$ to be a global source: if there is an edge from $u$ to every other vertices. In terms of the array $Outdegree[]$ in the above solution, you want to conclude that $u$ is global source if $Outdegree[u] = n-1$. This is a sufficient reason for $u$ to be a global source, but it is NOT necessary. Similar error for global sink.

You should see think that the criteria above I gave for global source (i.e., $Indegree[u] = 0$) is different from yours!

Another serious problem is that many students wrote such abstract code that it is CERTAINLY cannot be considered an algorithm. E.g., "If $V = S$ then..." (where $V, S$ are sets) is NOT acceptable. As rule of thumb about granularity of detail, all steps must be reduced to $O(1)$ time operations. Sometimes you must choose your data structures appropriately to ensure this $O(1)$ property. If you are not sure, you ought to err on the side of being careful, and write pseudo-code close to Java.

Some student used a $O(n^2)$ (or worse) algorithm when an $O(m+n)$ suffice.

# Question 5

(Heaps, 15 points)

**(i)** Describe an algorithm to sort $n$ numbers in an array. You must base your solution on heaps. NO pseudo-code is needed, just give high-level description in English, using any known subroutines on heaps.

**(ii)** Give an upper bound on its worst case running time complexity.

SOLUTION:

(i) This algorithm will first create a heap from the input array by repeated insertion into the heap (e.g., as in one of the homework problems). Now, we repeatedly deleteMin from this array. The deleted element is moved into the slot in array *formerly* occupied by the last element of the heap. After $n-1$ deleteMin's, we have a sorted array.

(ii) Each deleteMin takes $O(\log n)$ time. So the total time is $O(n \log n)$. The initial building of the heap is also $O(n \log n)$, so this is the overall bound.

PITFALLS: Students assume that if the array is made into a heap, then it is already sorted. This is far from the case.

# Question 6

(MST, 15 points)

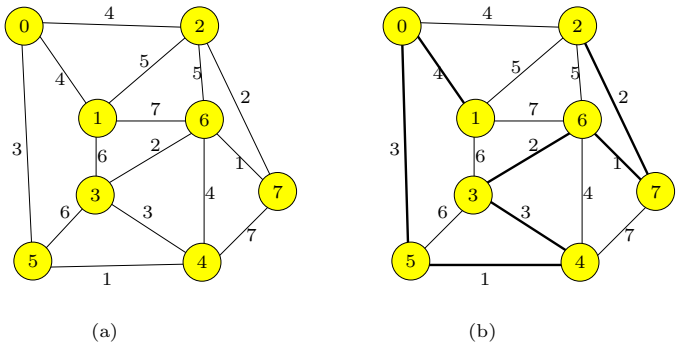


Figure 2: (a) Bigraph for MST. (b) An MST

Do a hand simulation of Kruskal's algorithm to compute a minimum spanning algorithm of the bigraph in Figure 2(a). You must (1) draw the original graph and indicate by thick lines those edges in the MST and (2) say what is the weight of the MST.

[If you do not use Kruskal's algorithm, we take off 3 points.]

SOLUTION:

(1) We sort the edges in non-decreasing weight order:

$$(4,5),(6,7),(2,7),(3,6),(0,5),(3,4),(0,1),(0,2),(4,6),(1,2),(2,6),(1,3)(1,6),(4,7),$$

then pick the edges in this order, as long as no cycles are created. The solution is seen in Figure 2(b).

(2) The weight of the MST is 16.