SAMPLE ANSWERS

Fall 2012, Second Test, Introduction to Algorithms

Name:
Section:
Email id:

5th November, 2012

This is an open books, open notes exam.
Calculator is allowed, Internet is not allowed.
Answer all six questions. Each Question is worth 10 points. You have 90 minutes to complete the exam.

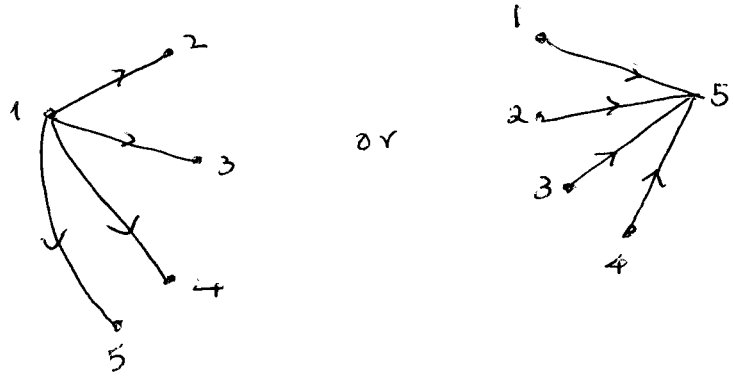
1. **Directed Graph**

(a) Assume that all nodes v are initialized with v.pre=-1 and v.post=-1. In doing a DFS on a directed graph, nodes get pre number when it gets visited first and post number when it gets visited last. If you encounter an edge (u,w) when you are visiting u, give pseudo code to check whether the edge (u,w) is a back edge or not.

if pre(w) > 0 and post(w) == -1

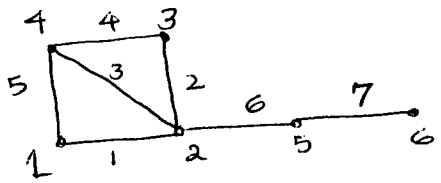
then (u,w) is a back edge

(b) Construct a directed acyclic graph with 5 nodes which has exactly 24 (4! = 24) topological orderings.



2. **Minimum Spanning Tree**

(a) Give an example of a connected weighted graph with 6 nodes (contains three cycles and all weights are distinct) such that the minimum spanning tree contains the largest weight edge.



edge (5,6) of wt. ~~of~~ 7 will be in any min. sp tree

(b) Will the minimum spanning tree algorithm work with negative weight cycles? - Give a brief explanation.

1. Yes
2. Kruskal's algorithm and Dijkstra's algorithm can be used with out any changes. This is so because of labels of nodes, does not depend upon the labels of other nodes.

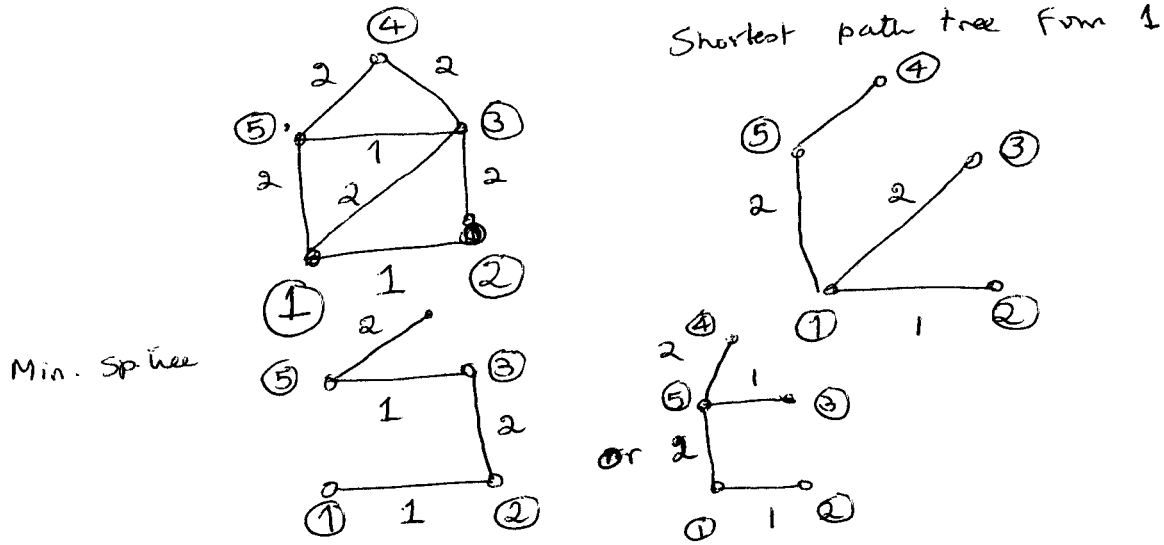
(c) Give an efficient algorithm (Pseudo Code will suffice) to find the largest weight spanning tree. (Same complexity as Minimum Spanning Tree)

$O((E \log V))$

1. sort the ~~edges~~ in decreasing (or non increasing order) in Q
2. while $n \neq n-1$ or Q is not empty ($n = 0$ and $T = \phi$)
   choose the largest weight edge in Q
   if e does not cause a cycle, add e to T. $n = n+1$
   delete e from Q
   else delete e from Q

3. **Minimum Spanning Tree /Strongly Connected Component(DFS on a Directed Graph**

(a) Give an example of an undirected graph (with 5 vertices and 7 edges , 5 edges have weight 2 and 2 edges have weight 1) such that the weight of the shortest path tree (using a starting vertex of 1) is not the same as the weight of the minimum spanning tree [**5 points**]



(b) Describe an efficient algorithm in linear time (in size of vertices and edges)(Pseudo Code will suffice) to find, a given directed graph G and a node v, whether there is a directed cycle containing node v.[**5 points**]

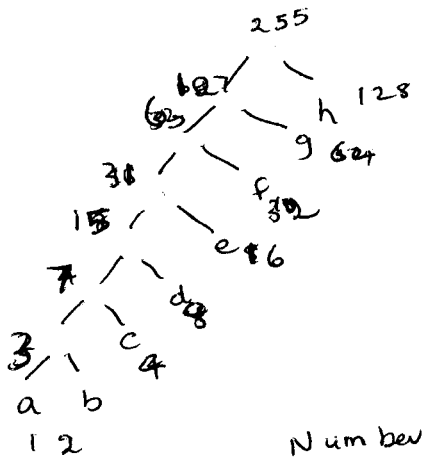
Do a DFS starting at v. for any descendents u of v, if there is a back edge $(u,v) \in E(G_1)$, there is a cycle containing v. Complexity $O(n+e)$

or

Find the Strongly connected components of G. If v is a not a singleton set among the strongly connected components, then there is a cycle containing v.

4. **Greedy Algorithm and Huffman Tree**

(a) Construct a Huffman Tree with the following characters a,b,c,d,e,f,g,h and their respective frequency of occurrences are 1,2,4,8,16,32,64,128 How many distinct Huffman Codes for this data are possible (just the number) (For instance, if the character set consists of two symbols [A, B] only, they can be coded as either [0, 1], or [1, 0]. So the number of Huffman codes is 2 in this case.). [**5 Points**]



one code

h	1
g	01
f	001
e	0001
d	00001
c	000001
b	0000001
a	0000000

Giving code is optional

Number of codes 128.

(b) You are given two sets A and B of n distinct positive integers. Describe a greedy algorithm (Pseudo Code will suffice) to find a set of X with size $\frac{n}{4}$, such that the sum of the numbers in this subset X is the smallest possible with $X = A - B$ (that is elements in X are in A but not in B) and the cardinality of X is $\frac{n}{4}$. What is its time complexity. (It is possible such a set does not exist.) [**5 points**]

Sort A and B in increasing order

X = Φ

m = 0

while m != $\frac{n}{4}$ or A is not empty

p = Choose the first element of A.

delete p from A

if (binarySearch(p, B) == 0

insert p in X

m = m + 1

Complexity is O(n log n)

5. **Dynamic Programming**

Give an $O(nt)$ algorithm for the following task (A pseudo code will suffice.)
Input : A list of n positive integers $a_1, a_2, \cdots, a_n$; a positive integer t.
Question : Does some subset of the a_i's add up to $3 \times t$? (You can use each a_i at most once) - The output is true or false. First write the recurrence relations before writing a pseudo code. Let $r = 3 \times t$ Let $SS(r, i)$ be true if there is a subset of first i elements that add up to r. Clearly $SS(y, 0) = False$ for all $1 \leq y \leq r$ and $SS(0, i) = True$ for all $1 \leq i \leq n$

(a) Define $SS(y, 1)$ for all $0 \leq y \leq r$.

(b) Define $SS(y, i)$ in terms of $SS(x, i-1)$ for all $0 \leq y \leq r$.

(c) Write a pseudo code.

[10 Points]

a) $SS(0,1) = true$

```
for (y = 0; y ≤ r; y++)
    if a_1 == y
        SS[y,1] = true
    else
        SS[y,1] = false
```

b) $SS[y,i] = SS[y,i-1] \vee SS[y-a_i, i-1]$ if $y \geq a_i$

$= SS[y,i-1]$ if $y < a_i$

for all y.

c)

```
SS[0,i] = true for i = 0 to n.
SS[j,i] = false for j = 1 to r.
for i = 1 to n
    for j = i to r
        if a_i ≤ j
            SS[j,i] = SS[j-a_i, i-1] ∨ SS[j, i-1]
        else
            SS[j,i] = SS[j,i-1]
```

6. **Dynamic Programming**
Counting tails. Given integers n and k, along with $p_1, p_2, \cdots, , p_n \in [0,1]$, you want to determine the probability of obtaining exactly k tails when n biased coins are tossed independently at random, where p_i is the probability that the ith coin comes up tails. Give a $O(nk)$ algorithm for this task. Assume that you can add and multiply numbers $\in [0,1]$ in $O(1)$ time.(Hint: Let us define P(i,j) means with i coins you get exactly j tails. P(i,j) with j > i is equal to 0).

(a) Find expressions for P(1,0), P(1,1).

(b) Find expressions for P(2,0),P(2,1) and P(2,2) in terms of P(i,j), $0 \leq j \leq 2$

(c) Write a recurrence equation for P(i,j) in terms of P(i-1,k), $0 \leq k \leq j$

(d) Write a pseudo code for solving the problem.
[**10 Points**]

a) $P[1,0] = 1 - p_1$
$P[1,1] = p_1$

b) $P[2,0] = P[1,0](1-p_2)$
$P[2,1] = P[1,1](1-p_2) + P[1,0]\, p_2$
$P[2,2] = P[1,1] \cdot p_2$

c) $P[i,j] = P[i-1,j](1-p_i) + P[i-1,j-1]\, p_i$ if $i \geq 1$

$P[0,j] = 1$ for all $j = 0$ to n.

for $i = 1$ to n
  for $j = 0$ to n
    if $j \leq i$
    {
      if $j == 0$
        $P[i,j] = P[i-1,j](1-p_i)$
      else
        $P[i,j] = P[i-1,j](1-p_i) + P[i-1,j-1]\, p_i$
    }
    else
      $P[i,j] = 0$