


National University of Computer and Emerging Sciences, Lahore Campus

	1 to	Course Name:	Design and Analysis of Algorithms			Course Code:	CS302
		Program:	BS Computer Science	12	13/14	Semester:	Spring 2018
		Duration:	180 Minutes			Total Marks:	80
		Paper Date:	25 May 2018			Weight	50
		Section:	ALL			Page(s):	10
		Exam Type:	Final				
Student : Name:				Roll No.		Section:	
Instruction/Notes:		Attempt the examination on this question paper. You can use extra sheets for rough work, do not attach extra sheets with this paper. Do not fill the table titled Question/marks.					
Question							
Marks		/45	/10	/10	10/	/5	/80

Q1) Write the tightest asymptotic upper bound for worst-case running time of following algorithms by selecting running times from following list. For sorting algorithms, n is the number of input elements. For graph algorithms, the number of vertices is n , and the number of edges is $\Theta(n)$. You need not justify your answers. Some running times may be used multiple times or not at all. **[5 Marks]**

$O(\lg n)$ $O(n)$ $O(n^2)$ $O(n^3)$ $O(n \lg n)$ $O(n^2 \lg n)$ $O(n^3 \lg n)$

- a. Insertion sort $O(n^2)$
- b. Heapsort $O(n \lg n)$
- c. BUILD-HEAP $O(n)$
- d. Topological Sort $O(n)$
- e. Bellman-Ford $O(n^2)$
- f. Depth-first search $O(n)$
- g. Floyd-Warshall $O(n^3)$
- h. Prim's $O(n \lg n)$

Q2) Give the order of growth of the running time for the following function. [5 Marks]

```
public static int f6(int N) {  
    if (N == 0) return 1;  
    return f6(N-1) + f6(N-1);  
}
```

$$T(n) = 2T(n-1) + O(1) = O(2^n)$$

Q3) You are given a function KSmall that returns the kth smallest element (location of the kth smallest element) of array A. Suppose that KSmall works in $O(n)$ in the worst case where n is the size of array A. Can you write Quick Sort Algorithm using KSmall such that it runs in worst case $O(n \log n)$ time on an n -element sequence. Briefly describe your algorithm. [5 Marks]

Use Ksmall to find the median of the array in linear time and use median as pivot. This way quick sort will run in $O(n \lg n)$ in the worst case.

Q4) Suppose we are given a set of activities/tasks specified by pairs of the start times and finish times as $T = \{(1, 2), (1, 3), (1, 4), (2, 5), (3, 7), (4, 9), (5, 6), (6, 8), (7, 9)\}$. Solve the activity selection problem for this set of tasks. Select maximum set of activities that can be scheduled without overlap. **[5 Marks]**

a) How many activities are selected?

4

b) Which activities are selected?

(1,2),(2,5),(5,6),(6,8)

Q5) Consider two strings: $X = \text{"adafd"}$, $Y = \text{"adabd"}$.

a) Show the longest common subsequence **table**, L , for strings X and Y . **[3 Marks]**

X/Y		a	d	a	b	D
	0	0	0	0	0	0
A	0	1	1	1	1	1
D	0	1	2	2	2	2
A	0	1	2	3	3	3
F	0	1	2	3	3	3
D	0	1	2	3	3	4

b) What is a longest common subsequence between these strings? **[2 Marks]**

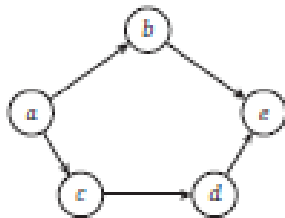
adad

Q6) How can we modify the dynamic programming algorithm from simply computing the maximum benefit/value for the 0-1 knapsack problem to selecting the items that gives specified benefit (if possible)? Write down the changed recurrence relation for this problem. **[5 Marks]**

Let S be the specified benefit

$$KS(n, W) = KS(n-1, W) == S \text{ or } KS(n-1, W - w_n) + v_n == S$$

Q7) Consider the directed acyclic graph G in following Figure. How many topological sorted orderings does it have? Write all different topological sorted orderings. **[5 Marks]**

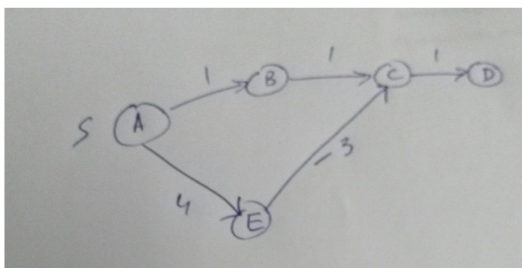


A,b,c,d,e

A,c,b,d,e

A,c,d,b,e

Q8) Give an example of a weighted directed graph, G , with negative-weight edges, but no negative-weight cycle, such that Dijkstra's algorithm incorrectly computes the shortest-path distances from some start vertex v . Mention the vertex for which Dijkstra will compute incorrect distance. **[5 Marks]**



For vertex D Dijkstra will compute incorrect distance i.e. 3 but shortest distance is 2

Q9) Suppose G is an undirected, connected, weighted graph such that the edges in G have distinct edge weights, which may be positive or negative. Will minimum spanning tree for G be changed if we square

all the edge weights in G , that is, G may have a different set of edges in its minimum spanning tree if we replace the weight, w , of each edge e in G , with w^2 . Justify your answer. **[5 Marks]**

No it will be changed as $-ve$ weight become large positive weights. So -9 would become 81 and that may not be selected as a light edge.

Q10) Suppose you are given a connected weighted undirected graph, G , with $|V|$ vertices and $|E|$ edges, such that the weight of each edge in G is an integer in the interval $[1, c]$, for a fixed constant $c > 0$. Show how to solve the single-source shortest paths problem, for any given vertex v , in G , in time $O(|V| + |E|)$.

Hint: Think about how to exploit the fact that the distance from v to any other vertex in G can be at most $O(cV) = O(V)$. Lesser credit will be awarded for less efficient solutions. **[10 Marks]**

a) Briefly describe your algorithm for solving this problem. **[3 Marks]**

Convert each edges (u,v) with weight w to w edges of weight 1 and $w-1$ additional vertices between u and v . This way the graph will be changed to an undirected graph and the set of vertices and set of edges in the changed graph will be $O(V)$ and $O(E)$ respectively. Now by applying BFS we can compute the shortest paths in linear time.

b) Write pseudocode of your algorithm **[6 Marks]**

1. For each edge (u,v) of weight w add $w-1$ dummy vertices $v_1, v_2, v_3, \dots, v_{w-1}$ and add edges as follows $(u, v_1), (v_1, v_2), \dots, (v_{w-1}, v)$
2. Now apply BFS to solve single source shortest path on changed graph.
3. The shortest distance between original nodes of the graph will be same as the one in changed graph

c) What is time complexity of your algorithm [1 Mark]

$O(V+E)$

Q11) Let A and B be two sequences of n integers each, in the range $[1, n^4]$. Given an integer x , describe an $O(n)$ -time algorithm for determining if there is an integer a in A and an integer b in B such that $x = a + b$. Lesser credit will be awarded for less efficient solutions. **[10 Marks]**

a) Briefly describe your algorithm for solving this problem. [3 Marks]

First sort both A and B using radix sort. Then compare the sum of first element of A and last element of B with x . If sum is greater than x then check compare sum of first element of A and second last element of B with x . If sum is smaller than x then compare sum of second element of A and last element of B with x . If sum is equal to x return the first and last element. And so on

b) Write pseudocode of your algorithm [6 Marks]

```
1. Sort the array A and B using radix sort with  $b = 4 \lg n$ 
2.  $i = 1, j = n$ 
3. While( $i < j$ )
4.     If( $A[i] + B[j] == x$ )
5.         Return true
6.     If( $A[i] + B[j] > x$ )
7.          $j \leftarrow j - 1$ 
8.     If( $A[i] + B[j] < x$ )
9.          $i \leftarrow i + 1$ 
```

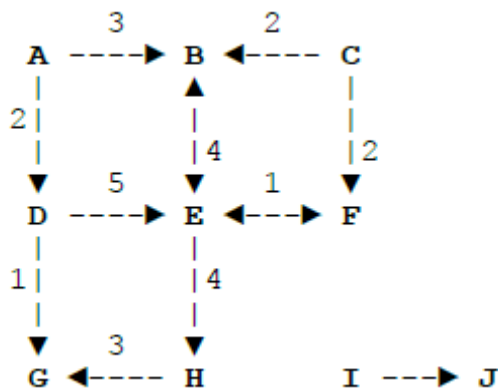
c) What is time complexity of your algorithm [1 Mark]

$O(n)$

Q12) Write a method popular that accepts a Graph G (G is represented as adjacency list) as a parameter and returns a Set of vertices in that graph that are “popular”. Assume that the graph represents users on a social network such as Facebook. A vertex represents a user, and a directed edge from A to B represents the fact that user A “likes” user B. The weight of the edge represents how much user A “likes” user B. A user v is “popular” if all of the following conditions are met:

- At least two other users “like” v
- More users “like” v than v “likes” other users. (*More arrows coming in than going out*)
- The combined weight of all "likes" toward v is more than the combined outbound weight of all the edges to other users that v "likes". (*More total edge weight coming in than going out.*)

For example, in the graph below, vertex B is “popular” because vertices A, C and E “like” him with a combined weight of $3+2+4=9$, while he “likes” only vertex E with a weight of 4. For this particular example graph, your method would return the set [B, F, G]. You may assume that the graph and its vertices are not null. **[10 Marks]**



a) Briefly describe your algorithm for solving this problem. [3 Marks]

Compute the indegree and out degree of each vertex. Also compute the sum of incoming edge weights and out going edge weight of each vertex and check condition for each vertex.

b) Write pseudocode of your algorithm [6 Marks]

1. Make four arrays Outdegree, Indegree, Outweight and Inweight of size V and initialize them to 0
2. S = null
3. For each vertex v in G
4. For each neighbor u of v
5. Outdegree[v]++
6. Indegree[u]++
7. Outweight[v] += w(v,u)
8. Inweight[u] += w(v,u)
9. For each vertex v
10. If(Indegree[v]>2 and Indegree[v]>Outdegree[v] and Inweight[v]>Outweight[v])
11. Add v to S
12. Return S

c) What is time complexity of your algorithm [1 Mark]

$O(V+E)$

Q13 is Only for Section A and B

Q13) Let $G=(V,E)$ be a connected, undirected graph and assume all edge weights are distinct. Consider a cycle $v_1, v_2, \dots, v_k, v_{k+1}$ in G , where $v_{k+1} = v_1$, and let (v_i, v_{i+1}) be the edge in the cycle with the largest edge weight. Prove that (v_i, v_{i+1}) does *not* belong to the minimum spanning tree T of G .

Hint (Use Lemmas used in proof of Prim's algorithm) **[5 Marks]**

Q14 is only for Section C, D and E

Q14) a) If the set of stack operations included a MULTIPUSH operation, which pushes k items onto the stack, along with push, pop and multipop. Would the $O(n)$ bound on the amortized cost of stack operations continue to hold? Justify your answer. **[3 Marks]**

No it will not be same because the worst sequence of operations would be multipush, multipop, multipush, multipop ... In this case amortized cost would be $O(kn)$ for n operations.

b) In B-tree, t represents its minimum degree and we say that t must be greater than 1. Why don't we allow a minimum degree of $t = 1$? **[2 Marks]**

Because if $t=1$ then it will be linked list and minimum number of nodes will be equal to zero