Proude To Be Virtulian MOAAZ SIDDIQ	**CS502_ Fundamentals of algorithm** **Solved Subjective** **From Final Term Papers**	**10 July,2013**
MC100401285	*Moaaz.pk@gmail.com* *Mc100401285@gmail.com*	**PSMD01**

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

CS502 – Final term (Fall 2012)

Q No.1 Suppose you could prove that an NP-complete problem cannot be solved in polynomial time. What would be the consequence?

Answer: Page 173

If we can solve a problem in polynomial time, we can certainly verify the solution in polynomial time. More formally, we do not need to see a certificate to solve the problem; we can solve it in polynomial time anyway.

However, it is not known whether P = NP. It seems unreasonable to think that this should be so. Being able to verify that you have a correct solution does not help you in finding the actual solution. The belief is that P 6= NP but no one has a proof for this.

Q No.3 Describe Dijkstra's algorithm working?

Answer: (Page 154)

Dijkstra's algorithm is a simple *greedy* algorithm for computing the **single-source shortest-paths** to all other vertices. Dijkstra's algorithm works on a weighted directed graph G = (V, E) in which all edge weights are non-negative, i.e., w(u, v) _ 0 for each edge (u, v) 2 E.

Negative edges weights maybe counter to intuition but this can occur in real life problems. However, we will *not allow negative cycles* because then there is no shortest path. If there is a negative cycle between, say, s and t, then we can always find a shorter path by going around the cycle one more time.

Q No.8 Explain the topological sort?

Answer: (Page 133)

A topological sort of a DAG is a linear ordering of the vertices of the DAG such that for each edge (u, v), u appears before v in the ordering. Computing a topological ordering is actually quite easy, given a DFS of the DAG. For every edge (u, v) in a DAG, the finish time of u is greater than the finish time of v (by the lemma). Thus, it suffices to output the vertices in the reverse order of finish times.

Q No.5 In the solution of edit distance technique, please describe two solution given (i) MATHS (ii) ARTS

Answer: (Page 77)

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

6.3.2 Edit Distance Algorithm

A better way to display this editing process is to place the words above the other:

S	*D*	*I*	*M*	*D*	*M*
M	A	_	T	H	S
A	_	R	T	_	S

The first word has a gap for every insertion (I) and the second word has a gap for every deletion (D). Columns with two different characters correspond to substitutions (S). Matches (M) do not count. The *Edit transcript* is defined as a string over the alphabet M, S, I, D that describes a transformation of one string into another. For example

S	*D*	*I*	*M*	*D*	*M*	
1+	1+	1+	0+	1+	0+	= 4

Q No.7 Explain the following two basic cases according to Floyd-Warshall Algorithm,
1. Don't go through vertex k at all.
2. Do go through vertex k.

Answer: (Page 162)

Don't go through k **at all**

Then the shortest path from i to j uses only intermediate vertices $\{1, 2, \ldots, k-1\}$. Hence the length of the shortest is $d_{ij}^{(k-1)}$

Do go through k

First observe that a shortest path does not go through the same vertex twice, so we can assume that we pass through k exactly once. That is, we go from i to k and then from k to j. In order for the overall path to be as short as possible, we should take the shortest path from i to k and the shortest path from k to j. Since each of these paths uses intermediate vertices $\{1, 2, \ldots, k-1\}$, the length of the path is $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$.

The following illustrate the process in which the value of $d_{3,2}^{k}$ is updated as k goes from 0 to 4.

Q No.6Variants of shortest path solution briefly?
Answer: (Page 153)

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

There are a few variants of the shortest path problem.
Single-source shortest-path problem: Find shortest paths from a given (single) *source* vertex s 2 V to every other vertex v 2 V in the graph G.
Single-destination shortest-paths problem: Find a shortest path to a given destination vertex t from each vertex v. We can reduce the this problem to a single-source problem by reversing the direction of each edge in the graph.
Single-pair shortest-path problem: Find a shortest path from u to v for given vertices u and v. If we solve the single-source problem with source vertex u, we solve this problem also. No algorithms for this problem are known to run asymptotically faster than the best single-source algorithms in the worst case.
All-pairs shortest-paths problem: Find a shortest path from u to v for *every pair* of vertices u and v. Although this problem can be solved by running a single-source algorithm once from each vertex, it can usually be solved faster.

CS502 – Final term (Fall 2012)

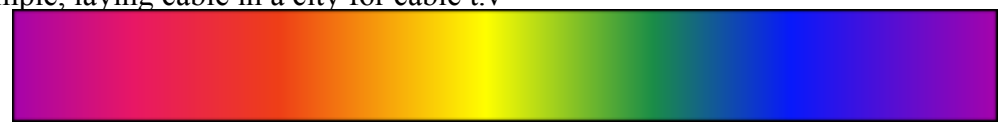
What is the Running time of Bellman Ford Algorithm?

Answer: Click here for detail

Bellman–Ford runs in $O(|V|\cdot|E|)$ time,

Define Forward Edge from ancestor to descendent (u, v) where v is a proper descendent of u in the tree.
Answer: (Page 130)
For a forward edge (u, v), v is a descendent of u and so v's start-finish interval is contained within u's implying that v has an earlier finish time. For a cross edge (u, v) we know that the two time intervals are disjoint. When we were processing u, v was not white (otherwise (u, v) would be a tree edge), implying that v was started before u. Because the intervals are disjoint, v must have also finished before u

What is the common problem in communications networks and circuit designing?
Answer: (Page 142)
A common problem is communications networks and circuit design is that of connecting together a set of nodes by a network of total minimum length. The length is the sum of lengths of connecting wires. Consider, for example, laying cable in a city for cable t.v



Q pseudo code of timestamp DFS
Answer: (Page 126)
DFS(G)
1 **for** (each u 2 V)

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

2 **do** color[u] white
3 pred[u] nil
4 time 0
5 **for** each u 2 V
6 **do if** (color[u] = white)
7 **then** DFSVISIT(u)

Prove the following lemma,
Lemma: Given a digraph G = (V, E), consider any DFS forest of G and consider any edge (u, v) ∈ E. If this edge is a tree, forward or cross edge, then f[u] > f[v]. If this edge is a back edge, then f[u] ≤ f[v]
Answer: (Page 130)
Proof: For the non-tree forward and back edges the proof follows directly from the parenthesis lemma. For example, for a forward edge (u, v), v is a descendent of u and so v's start-finish interval is contained within u's implying that v has an earlier finish time. For a cross edge (u, v) we know that the two time intervals are disjoint. When we were processing u, v was not white (otherwise (u, v) would be a tree edge), implying that v was started before u. Because the intervals are disjoint, v must have also finished before u.

How Kruskal's algorithm works ?
Answer: (Page 127)
Kruskal's algorithm works by adding edges in increasing order of weight (lightest edge first). If the next edge does not induce a cycle among the current set of edges, then it is added to A. If it does, we skip it and consider the next in order. As the algorithm runs, the edges in A induce a *forest* on the vertices. The trees of this forest are eventually merged until a single tree forms containing all vertices.

Define according to Kruskal's algorithm
creat_set(u)
find_set(U)
union(u,v)
Answer: Page 147
Create-set(u): Create a set containing a single item u.
Find-set(u): Find the set that contains u
Union(u,v): merge the set containing u and set containing v into a common set.

CS502 – Final term (Fall 2012)

Q what is free tree of 2 marks

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Answer: (Page 142)

A free tree is a tree with no vertex designated as the root vertex.

Q describe algorithm as a student of computer science 2 marks

Answer: (Page 7)

Unlike a program, an algorithm is a mathematical entity, which is independent of a specific programming language, machine, or compiler.

Q what is minimizing spanning tree of 3 marks

Answer: Page (page 142)

A minimum spanning tree is a tree of minimum weight.

The computational problem is called the *minimum spanning tree* (MST) problem. Formally, we are given a connected, undirected graph G = (V, E) Each edge (u, v) has numeric weight of cost.

CS502 – Final term (Fall 2012)

Q:A sequence of a value in a column of the dynamic programming table for an instance of the knapsack problem is always non-decreasing order (true or false) (5)

Answer: (not Sure)

This statement is true because at every step we have more weight and value than previous and according to knapsack we always prefer the value which Is maximum

Q: Describe Dijkstra's algorithm working?

Answer: Rep

Q: Explain the following two basic cases according to Floyd-Warshall Algorithm,

Answer: Rep

Q: Differentiate between back edge and forward edge

Answer: (Page 128)

Back edge: (u, v) where v is an ancestor of u in the tree.

Forward edge: (u, v) where v is a proper descendent of u in the tree.

Suppose you could prove that an NP-complete problem cannot be solved in polynomial time. What would be the consequence?

Answer: Rep

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Q: 58.How Kruskal's algorithm works?
Answer: Rep

Q:what is path? (2)
Answer: (Page 115)
A *path* in a directed graphs is a sequence of vertices hv0, v1, . . . , vki such that (vi−1, vi) is an edge for i = 1, 2, . . . , k.

Q: define free tree (2)
Answer: Rep

Q:comment whether the computational powers RAM sequential machine are less than that the parallel machine (3)
Answer: Page 10
RAM seems to go a good job of describing the computational power of most modern (non parallel) machines. It does not model some elements, such as efficiency due to locality of reference.There are some "loop-holes" (or hidden ways of subverting the rules)

Q: what is genius of warshell algorithm? (3)
Answer: Page 62
As with other dynamic programming algorithms, the genius
of the algorithm is in the clever recursive formulation of the shortest path problem. For a path
p = hv1, v2, . . . , vl, we say that the vertices v2, v3, . . . , vl−1 are the *intermediate vertices* of this path.

Q: given a graph G(V,E) any DFS forest of G and consider edge (u,v)*E* E. prove that if this edge is tree, forward or back edge then f[u]>f[v] and if this edge is backedge then f[u] ≤ [v]. (3)
Answer: Rep

CS502 – Final term (Fall 2012)

Q1-describe Asymptotic Notation

Answer: https://en.wikipedia.org/wiki/Big_O_notation
Asymptotic Notation is a formal notation for discussing and
analyzing classes of functions and it is used to classify algorithms by how they respond (*e.g.,* in their processing time or working space requirements) to changes in input size

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

Q4 - free tree

Answer: Rep

Q5- reduction and example

Answer: NP-complete problem, then ever problem in NPC will be solvable in polynomial time. For this, we need the concept of *reductions*.

2) floyd marshall running time ?

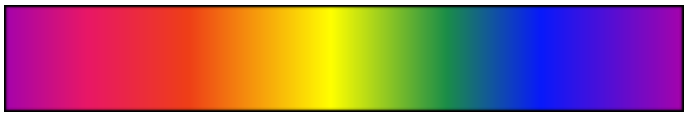
Answer: Page 164

The running time of floyd marshall is _(n3)

CS502 – Final term (Fall 2012)

Q: 1 Define common problem in communication networks.(2)
Answer: Rep

Q:2 Why we need reductions? Give Example. (5)
Answer: Page 173
The class NP-complete (NPC) problems consists of a set of decision problems (a subset of class NP) that no one knows how to solve efficiently. But if there were a polynomial solution for even a single NP-complete problem, then ever problem in NPC will be solvable in polynomial time. For this, we need the concept of *reductions*.



EXAMPLE:
3-color: Given a graph G, can each of its vertices be labelled with one of 3 different colors such that two adjacent vertices have the same label (color).
Coloring arises in various partitioning problems where there is a constraint that two objects cannot be assigned to the same set of partitions. The term "coloring" comes from the original application which was in map drawing. Two countries that share a common border should be colored with different colors.
It is well known that planar graphs can be colored (maps) with *four colors*. There exists a polynomial time algorithm for this. But determining whether this can be done with 3 colors is hard and there is no polynomial time algorithm for it. In Figure 9.3, the graph on the left can be colored with 3 colors while the graph on the right cannot be colored.

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

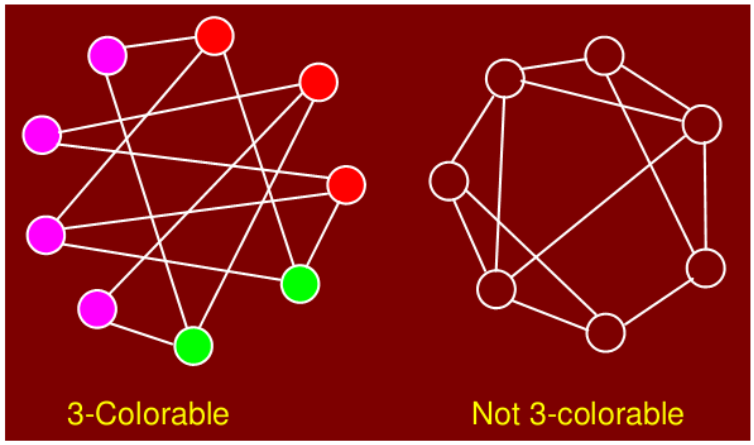
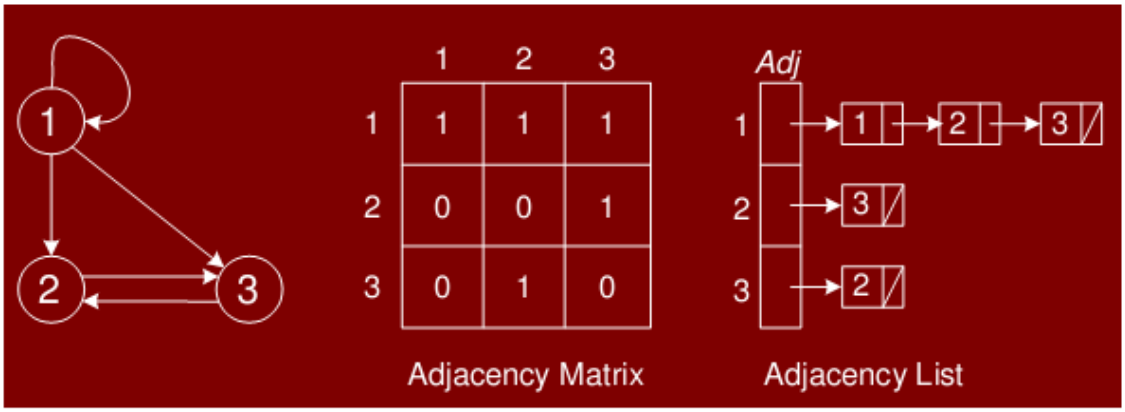
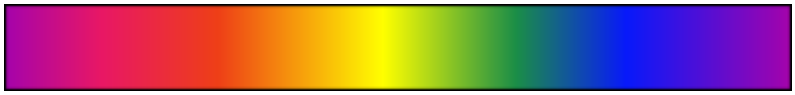


Figure 9.3: Examples of 3-colorable and non-3-colorable graphs

Q: 3 Make adjacency list of given graph. (5)
Answer: Page 116





Q:4 Kruskal's algorithm can return different spanning trees for the same input graph G, depending on how ties are broken when the edges are sorted into order. Show that for each minimum spanning tree T of G, there is a way to sort the edges of G in Kruskal's algorithm so that the algorithm returns T.

Solution:
We would start by sorting the edges in of G in non-descending order. In addition, we would want that among the edges of same weight, the edges which are contained in T are placed in first positions.
The claim here is that Kruskal's algorithm will return T when run on E if sorted in the above mentioned manner.
Proof:

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

T: e1 ≤ e2 ≤ , … , ≤ em
T': e'1 ≤ e'2 ≤ , … , ≤ e'm
Weight of ei = Weight of e'i where 1 ≤ i ≤ m
Since the algorithm always places edges of T first, the edges of T will be chosen to T'.

Q:5 How to get Knapsack optimal solution with dynamic programming algorithm table ? (5)
Answer: (Page 96)
The algorithm for computing V[i, j] does not keep record of which subset of items gives the optimal solution. To compute the actual subset, we can add an auxiliary boolean array keep [i, j] which is 1 if we decide to take the ith item and 0 otherwise. We will use all the values keep[i, j] to determine the optimal subset T of items to put in the knapsack as follows:
• If keep[n,W] is 1, then n 2 T. We can now repeat this argument for keep [n − 1,W − wn].
• If kee[n,W] is 0, the n 62 T and we repeat the argument for keep[n − 1,W].

We will add this to the knapsack algorithm:

```
KNAPSACK(n, W)
for w = 0, W
do V[0, w] ← 0
for i = 0, n
do V[i, 0] ← 0
    for w = 0, W
    do if (w_i ≤ w & v_i + V[i − 1, w − w_i] > V[i − 1, w])
            then V[i, w] ← v_i + V[i − 1, w − w_i]; keep[i, w] ← 1
            else  V[i, w] ← V[i − 1, w]; keep[i, w] ← 0
// output the selected items
k ← W
for i = n downto 1
do if (keep[i, k] = 1)
        then output i
                k ← k − w_i
```

Q:6 Define the following in Kruskal algorithm
Create Set-u
Find Set-u
Union(u,v)
Answer: Rep

Q:7 In divide-conquer strategy which step have main processing?
Answer: (Page 27)
The main elements to a divide-and-conquer solution are
Divide: the problem into a small number of pieces
Conquer: solve each piece by applying divide and conquer to it *recursively*

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

Combine: the pieces together into a global solution.

Q:8 Q No.7 Explain the following two basic cases according to Floyd-War shall Algorithm?
Answer: Rep

Q:9:
Chain matrix multiplication? (2)
Answer: (Page 85)
Matrix multiplication is an associative but not commutative operation. We are free to add parenthesis the above multiplication but the order of matrices cannot be changed. The *Chain Matrix Multiplication Problem* is stated as follows:
Given a sequence A1,A2, . . . ,An and dimensions p0, p1, . . . , pn where Ai is of dimension pi−1 × pi, determine the order of multiplication that minimizes the number of operations.

Q:11 How to propagate shortest path in Bell men Ford theorem?
Answer: (Page 160)
The shortest path information is propagated sequentially along each shortest path in the graph.

CS502 – Final term (Fall 2012)

Q: What is the common problem in communications networks and circuit designing?
Answer: Rep

Q: How Plagiarism detection can be done with edit distance?
Answer: Page 76
Plagiarism Detection
If someone copies, say, a C program and makes a few changes here and there, for example, change variable names, add a comment of two, the edit distance between the source and copy may be small. The edit distance provides an indication of similarity that might be too close in some situations.

Q: Consider the following

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

0	1	2	3
1	2	4	0
2	0	0	2
3	4	0	2

Can an adjacency matrix for a directed graph ever not be square in shape? Why or why not?
Answer: click here 4 detail
No. since we want to describe the relationship between each node and each other node, we need precisely n^2 matrix entries.

Q: What do you mean by polynomial time algorithm? Explain what kind of problems can be solved by using polynomial time algorithm?
Answer: (Page 169)
A *polynomial time algorithm* is any algorithm that runs in $O(n^k)$ time.
A problem is solvable in polynomial time if there is a polynomial time algorithm for it.

Q: Run Radix sort on the following array of integers, show first two passes of sorting:
8081, 7342, 9287, 9583, 3202, 5215, 8397, 8001, 972, 5315, 1983, 283, 1664, 8107
Answer :Page 71

8081		808[1]		32[0]2
7342		800[1]		80[0]1
9287		734[2]		81[0]7
9583		320[2]		52[1]5
3202		198[3]		53[1]5
5215		958[3]		97[2]
8397	=>	166[4]	=>	28[3]
8001		521[5]		73[4]2
972		531[5]		16[6]4
5315		928[7]		19[8]3
1983		839[7]		80[8]1
283		810[7]		92[8]7
1664		283		95[8]3
8107		972		83[9]7

CS502 – Final term (Spring 2012)

How can we make it possible for an array of "n" elements that every element has equal probability of '1/n' to be selected as pivot elements?
Answer: (Page 50)

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

To analyze the average running time, we let T(n) denote the average running time of QuickSort on a list of size n. It will simplify the analysis to assume that all of the elements are distinct. The algorithm has n random choices for the pivot element, and each choice has an equal probability of 1/n of occurring. So we can modify the above recurrence to compute an average rather than a max, giving:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ \frac{1}{n}\sum_{q=1}^{n}(T(q-1) + T(n-q) + n) & \text{otherwise} \end{cases}$$

The time $T(n)$ is the weighted sum of the times taken for various choices of q. I.e.,

$$\begin{aligned} T(n) = \Big[&\frac{1}{n}(T(0) + T(n-1) + n) \\ &+ \frac{1}{n}(T(1) + T(n-2) + n) \\ &+ \frac{1}{n}(T(2) + T(n-3) + n) \\ &+ \cdots + \frac{1}{n}(T(n-1) + T(0) + n)\Big] \end{aligned}$$

write two steps of dynamic programming

Answer: Page 75

Formulate problem recursively. Write down a formula for the whole problem as a simple combination of answers to smaller sub problems.

• **Build solution to recurrence from bottom up**. Write an algorithm that starts with base cases and works its way up to the final solution.

pseoudo code for strong component

Answer: Page 139

STRONGCOMPONENTS(G)
1 Run DFS(G) computing finish times f[u]
2 Compute GT
3 Sort vertices of GT in decreasing f[u]
4 Run DFS(GT) using this order
5 Each DFS tree is a strong component

CS502 – Final term (Spring 2012)

Difference b/w back ward and forward 2 marks

Answer: Rep

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Polynomial time algorithm 2 marks
Answer: Rep

Code that fib memorization ka. 3 marks
Answer: Page 74
MEMOFIB(n)
1 **if** (n < 2)
2 **then return** n
3 **if** (F[n] is undefined)
4 **then** F[n] MEMOFIB(n − 1) + MEMOFIB(n − 2)
5 **return** F[n]

CS502 – Final term (Spring 2012)

What is difference between O (n log n) and theta (n log n)? (2)
Answer:
The Theta-notation asymptotically bounds a function from above and below. When we have only an asymptotic upper bound, we use O-notation.

7. Consider the following code:

```
for (j=1; j<n;j++)
for (k=1; k<15;k++)
for(l=5; l<n; l++)
{
Do_something_constant();
}
```

What order is the execution of this code? (3)
Answer:
The execution will be in inside-out order

8. How Dijkstra's algorithm works? (3)
Answer: Rep

9. Explain the topological sort? (5)
Answer: (Page 133)

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

A topological sort of a DAG is a linear ordering of the vertices of the DAG such that for each edge (u, v), u appears before v in the ordering.
Computing a topological ordering is actually quite easy, given a DFS of the DAG. For every edge (u, v) in a DAG, the finish time of u is greater than the finish time of v (by the lemma). Thus, it suffices to output the vertices in the reverse order of finish times.

CS502 – Final term (Spring 2012)

1. Heap sort Algorithm
Answer: (Page 41)

```
HEAPSORT( array A, int n)
BUILD-HEAP(A, n)
m   n
while (m _ 2)
do SWAP(A[1],A[m])
m   m− 1
HEAPIFY(A, 1,m)
```

3. Prim's Algorithm
Answer: Page 151

```
PRIM((G,w, r))
for ( each u 2 V)
do key [u]   1; pq.insert (u, key[u])
color [u]   white
key [r]   0; pred [r]   nil; pq.decrease key (r, key [r]);
while ( pq.not empty ())
do u   pq.extract min ()
for ( each u 2 adj [u])
do if ( color [v] == white )and( w (u, v) < key [v])
then key [v] = w (u, v)
pq.decrease key (v, key [v])
pred [v] = u
color [u] = black
```

7. Prove that the generic TRAVERSE (S) marks every vertex in any connected graph exactly once and the set of edges (v, parent (v)) with parent (v) ¹F form a spanning tree of the graph.
Answer: Page 125

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

Lemma:

The generic TRAVERSE(s) marks every vertex in any connected graph exactly once and the set of edges $(v, \mathtt{parent}(v))$ with $\mathtt{parent}(v) \neq \varnothing$ form a spanning tree of the graph.

Proof:

First, it should be obvious that no vertex is marked more than once. Clearly, the algorithm marks s. Let $v \neq s$ be a vertex and let $s \to \cdots \to u \to v$ be a path from s to v with the minimum number of edges.

Since the graph is connected, such a path always exists. If the algorithm marks u, then it must put (u, v) into the bag, so it must take (u, v) out of the bag at which point v must be marked. Thus, by induction on the shortest-path distance from s, the algorithm marks every vertex in the graph.

Call an edge $(v, \mathtt{parent}(v))$ with $\mathtt{parent}(v) \neq \varnothing$, a *parent edge*. For any node v, the path of parent edges $v \to \mathtt{parent}(v) \to \mathtt{parent}(\mathtt{parent}(v)) \to \ldots$ eventually leads back to s. So the set of parent edges form a connected graph.

Clearly, both end points of every parent edge are marked, and the number of edges is exactly one less than the number of vertices. Thus, the parent edges form a *spanning tree*.

2) what is a run time analysis and its two criteria

Answer: (Page 13)

The running time depends upon the input size, e.g. n Different inputs of the same size may result in different running time. For example, breaking out of the inner loop in the brute-force algorithm depends not only on the input size of P but also the structure of the input.

Two criteria for measuring running time are *worst-case time* and *average-case time*.

Worst-case time is the maximum running time over all (legal) inputs of size n. Let I denote an input instance, let $|I|$ denote its length, and let $T(I)$ denote the running time of the algorithm on input I. Then

$$T_{worst}(n) = \max_{|I|=n} T(I)$$

Average-case time is the average running time over all inputs of size n. Let $p(I)$ denote the probability of seeing this input. The average-case time is the weighted sum of running times with weights

CS502 – Final term (Spring 2012)

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Q1) answer the follwing according to Floyd Warshall 1) runing time 2) space use
Answer: Page 164
the running time is theta (n3). The space used by the algorithm is Theta (n2).

2) write suedo code of dijkstra algorithm? 5marks
Answer: Page 156
DIJKSTRA((G,w, s))
1 **for** (each u 2 V)
2 **do** d[u] 1
3 pq.insert (u, d[u])
4 d[s] 0; pred [s] nil; pq.decrease key (s, d[s]);
5 **while** (pq.not empty ())
6 **do** u pq.extract min ()
7 **for** (each v 2 adj[u])
8 **do if** (d[u] + w(u, v) < d[v])
9 **then** d[v] = d[u] + w(u, v)
10 pq.decrease key (v, d[v])
11 pred [v] = u

3)Give a detailed example for 2-d maxima problem 2marks
Answer: (Page 10)
Suppose you want to buy a car. You want the pick the fastest car. But fast cars are expensive; you want the cheapest. You cannot decide which is more important: speed or price. Definitely do not want a car if there is another that is both faster and cheaper. We say that the fast, cheap car *dominates* the slow, expensive car relative to your selection criteria. So, given a collection of cars, we want to list those cars that are not dominated by any other.
Here is how we might model this as a formal problem.
• Let a point p in 2-dimensional space be given by its integer coordinates, p = (p.x, p.y).
• A point p is said to be dominated by point q if p.x _ q.x and p.y _ q.y.
• Given a set of n points, P = {p1, p2, . . . , pn} in 2-space a point is said to be maximal if it is not dominated by any other point in P.
The car selection problem can be modelled this way: For each car we associate (x, y) pair where x is the speed of the car and y is the negation of the price. High y value means a cheap car and low y means expensive car. Think of y as the money left in your pocket after you have paid for the car. Maximal points correspond to the fastest and cheapest cars.
The *2-dimensional Maxima* is thus defined as
• Given a set of points P = {p1, p2, . . . , pn} in 2-space, output the set of maximal points of P, i.e., those points pi such that pi is not dominated by any other point of P.

5) how generic algorithms work with minimum spaning tree 3marks
Answer: Page 143

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

CS502 - Fundamentals of Algorithms (Lecture 1 - 45).pdf - Adobe Reader

8.5.1 Computing MST: Generic Approach

Let $G = (V, E)$ be an undirected, connected graph whose edges have numeric weights. The intuition behind greedy MST algorithm is simple: we maintain a subset of edges E of the graph . Call this subset A. Initially, A is empty. We will add edges one at a time until A equals the MST.

A subset $A \subseteq E$ is *viable* if A is a subset of edges of *some* MST. An edge $(u, v) \in E - A$ is *safe* if $A \cup \{(u, v)\}$ is viable. In other words, the choice (u, v) is a safe choice to add so that A can still be extended to form a MST.

Note that if A is viable, it cannot contain a cycle. A generic greedy algorithm operates by repeatedly adding any *safe* edge to the current spanning tree.

When is an edge safe? Consider the theoretical issues behind determining whether an edge is safe or not. Let S be a subset of vertices $S \subseteq V$. A *cut* $(S, V - S)$ is just a partition of vertices into two disjoint subsets. An edge (u, v) *crosses* the cut if one endpoint is in S and the other is in $V - S$.

Given a subset of edges A, a cut *respects* A if no edge in A crosses the cut. It is not hard to see why respecting cuts are important to this problem. If we have computed a partial MST and we wish to know which edges can be added that *do not* induce a cycle in the current MST, any edge that crosses a respecting cut is a possible candidate.

7) Which points should be supposed to prove the correctness of the Dijkstra's Algorithm 3marks

Answer: (Page 158 – 159)

CS502 - Fundamentals of Algorithms (Lecture 1 - 45).pdf - Adobe Reader

We will prove the correctness of Dijkstr's algorithm by Induction. We will use the definition that $\delta(s, v)$ denotes the minimal distance from s to v.

For the base case

1. $S = \{s\}$
2. $d(s) = 0$, which is $\delta(s, s)$

Assume that $d(v) = \delta(s, v)$ for every $v \in S$ and all neighbors of v have been relaxed. If $d(u) \leq d(u')$ for every $u' \in V$ then $d(u) = \delta(s, u)$, and we can transfer u from V to S, after which $d(v) = \delta(s, v)$ for every $v \in S$.

We do this as a proof by contradiction. Suppose that $d(u) > \delta(s, u)$. The shortest path from s to u, $p(s, u)$, must pass through one or more vertices exterior to S. Let x be the last vertex inside S and y be the first vertex outside S on this path to u. Then $p(s, u) = p(s, x) \cup \{(x, y)\} \cup p(y, u)$.

CS502 - Fundamentals of Algorithms (Lecture 1 - 45).pdf - Adobe Reader

The length of $p(s, u)$ is $\delta(s, u) = \delta(s, y) + \delta(y, u)$. Because y was relaxed when x was put into S, $d(y) = \delta(s, y)$ by the convergence property. Thus $d(y) \leq \delta(s, u) \leq d(u)$. But, because $d(u)$ is the smallest among vertices not in S, $d(u) \leq d(y)$. The only possibility is $d(u) = d(y)$, which requires $d(u) = \delta(s, u)$ contradicting the assumption.

By the upper bound property, $d(u) \geq \delta(s, u)$. Since $d(u) > \delta(s, u)$ is false, $d(u) = \delta(s, u)$, which is what we wanted to prove. Thus, if we follow the algorithm's procedure, transferring from V to S, the vertex in V with the smallest value of $d(u)$ then all vertices in S have $d(v) = \delta(s, v)$

8.6.3 Bellman-Ford Algorithm

Dijkstra's single-source shortest path algorithm works if all edges weights are non-negative and there are

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

CS502 – Final term (Spring 2012)

4- What are the minimum and maximum number of elements in a heap of height h (5 marks)

Answer: Page 44

At level h, there will be 2^h or less nodes.

5- Explain Floyd Warshell algorithm (3 marks)

Answer: (Page 164)

```
FLOYD-WARSHALL(n,w[1..n, 1..n])
for (i = 1, n)
do for (j = 1, n)
do d[i, j]   w[i, j]; mid[i, j]   null
for (k = 1, n)
do for (i = 1, n)
do for (j = 1, n)
do if (d[i, k] + d[k, j] < d[i, j])
then d[i, j] = d[i, k] + d[k, j]
mid[i, j] = k
```

7- Compare bellman ford algorithm with dijikstr's algorithm. Also give the time complexity of bellman ford algorithm (3 marks)

Answer: Page 159

Dijkstra's single-source shortest path algorithm works if all edges weights are non-negative and there are no negative cost cycles. Bellman-Ford *allows* negative weights edges and no negative cost cycles. The algorithm is slower than Dijkstra's, running in _(VE) time.

psedo code of timestamp DFS

Answer: Page 126

```
DFS(G)
for (each u 2 V)
do color[u]   white
pred[u]   nil
time   0
for each u 2 V
do if (color[u] = white)
then DFSVISIT(u)
```


Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari

Total running time of edit distance

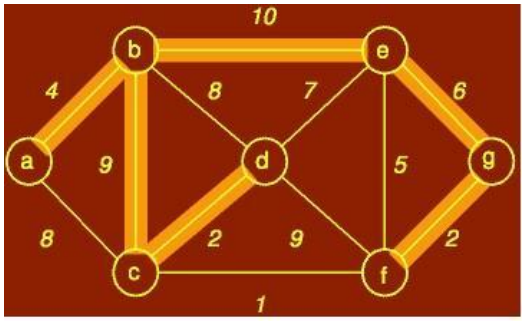
Answer: (Page 84)

The total running time is θ (n2).

CS502 – Final term (Spring 2012)

Question No: 49 (Marks: 5)

What is the cost of the following graph?



Answer: Page 142

Cost =33

Question No: 51 (Marks: 5)

Floyd Warshal algorithm with three recursive steps

Answer: Page 167

PATH(i, j)
1 **if** (mid[i, j] == null)
2 **then** output(i, j)
3 **else** PATH(i,mid[i, j])
4 PATH(mid[i, j], j)

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies
(IEMS) Samundari