Cs 502 Subjective Mid Term
Prepared by

SAIF

For Past Papers visit:

http://vupkpastpapers.blogspot.com

REMEMBER ME IN YOUR PRAYERS

MID TERM EXAMINATION

Heap sort Algorithm

```
HEAPSORT( array A, int n)
1  BUILD-HEAP(A, n)
2  m ← n
3  while (m ≥ 2)
4  do SWAP(A[1], A[m])
5     m ← m − 1
6     HEAPIFY(A, 1, m)
```

Page #41

BFS pseudo code in graph

Start with s and visit its adjacent nodes. Label them with distance 1. Now consider the neighbours of neighbours of s. These would be at distance 2. Now consider the neighbours of neighbours of neighbours of s. These would be at distance 3. Repeat this until no more unvisited neighbours left to visit.

heapify

There is one principal operation for maintaining the heap property. It is called Heapify. The idea is that we are given an element of the heap which we suspect may not be in valid heap order, but we assume that all of other the elements in the subtree rooted at this element are in heap order. In particular this root element may be too small. To fix this we “sift” it down the tree by swapping it with one of its children. Which child? We should take the larger of the two children to satisfy the heap ordering property. This continues recursively until the element is either larger than both its children or until its falls all the way to the leaf level.

Polynomial time algorithm

A *polynomial time algorithm* is any algorithm that runs in O(nk) time. A problem is solvable in polynomial time if there is a polynomial time algorithm for it.

What is a run time analysis and its two criteria

The main purpose of our mathematical analysis will be measuring the execution time. The running time of an implementation of the algorithm would depend upon the speed of the computer, programming language, optimization by the compiler etc. 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 being the probabilities:

$$T_{avg}(n) = \sum_{|I|=n} p(I)T(I)$$

We will almost always work with worst-case time. Average-case time is more difficult to compute; it is difficult to specify probability distribution on inputs. Worst-case time will specify an upper limit on the running time.**Page #13**

Dijkstra algorithm correctness criteria two conditions

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 $\delta(s, s)$

Page #158

Floyd algorithm

A graph analysis algorithm for finding shortest paths in a weighted graph with positive or negative edge weights. A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between *all* pairs of vertices.

write suedo code of dijkstra algorithm? 5marks

```
DIJKSTRA((G, w, s))
 1  for ( each  u ∈ V)
 2  do d[u] ← ∞
 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 ∈ 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
```

Page #156

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

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. Like Dijkstra's algorithm, Bellman-Ford is based on performing repeated relaxations. Bellman-Ford applies relaxation to *every edge* of the graph and repeats this V − 1 times.

Page #159

psedo code of timestamp DFS

DFS(G)
1 **for** (each u 2 V)
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)

Page #126

variants of shortest path solution

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.

Page #153

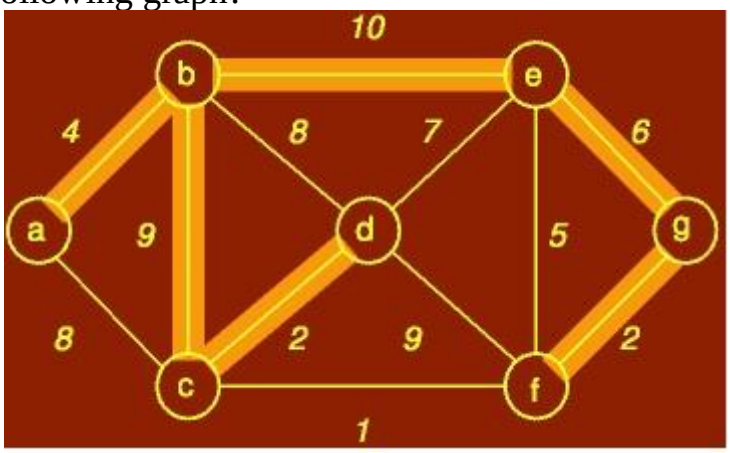
Total running time of edit distance

The total running time is _(n2).

Page #84

Question No: 49 (Marks: 5)

What is the cost of the following graph?



Cost =33

Question No: 50 (Marks: 5)

Let the adjacency list representation of an undirected graph is given below. Explain what general property of the list indicates that the graph has an isolated vertex.

a → b → c → e
b → a → d
c → a → d → e → f
d → b → c → f
e → a → c → f
f → c → d → e
g

If any of the vertex do not have any adjacent vertices, It is shown in the list a single entity.

Question No: 48 (Marks: 3)

Recursive explanation of dynamic programming.

Dynamic programming is essentially recursion without repetition. Developing a dynamic programming
algorithm generally involves two separate steps:

• **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.

Question No: 28 (Marks: 2)

Differentiate between back edge and forward edge.

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.

Question No: 29 (Marks: 2)
How the generic greedy algorithm operates in minimum spanning tree?
A generic greedy algorithm operates by repeatedly adding any *safe* edge to the current spanning tree.

Question No: 30 (Marks: 2)
What are two cases for computing assuming we already have the previous matrix $d^{(k-1)}$ using Floyed-Warshall algorithm?
There are two basic cases:
1. Don't go through vertex k at all.
2. Do go through vertex k.

Question No: 31 (Marks: 3)
Describe Minimum Spanning Trees Problem with examples.
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.

Question No: 32 (Marks: 3)
What is decision problem, also explain with example?
A problem is called a *decision problem* if its output is a simple "yes" or "no" (or you may this of this as true/false, 0/1, accept/reject.)

Question No: 33 (Marks: 3)
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.
First, it should be obvious that no vertex is marked more than once. Clearly, the algorithm marks s. Let v 6= s be a vertex and let s ! · · · ! u ! v be a path from s to v with the minimum number of edges.
Page #125 lemma

Question No: 35 (Marks: 5)
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]
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.

1)RAM(Random Access memory)and its Applications?

A RAM is an idealized machine with an infinitely large random-access memory. Instructions are executed one-by-one (there is no parallelism). Each instruction involves performing some basic operation on two values in the machines memory. Basic operations include things like assigning a value to a variable, computing any basic arithmetic operation (+, - , × , integer division) on integer values of any size, performing any comparison (e.g. x _ 5) or boolean operations, accessing an element of an array (e.g. A[10]). We assume that each basic operation takes the same constant time to execute.

2)Describe Dijkstra's algorithm working?

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.

3)Prim algorithm graph?

Prim's algorithm builds the MST by adding leaves one at a time to the current tree. We start with a root vertex r; it can be any vertex. At any time, the subset of edges A forms a single tree. We look to add a single vertex as a leaf to the tree. Graph on page #150.

4) Give a detailed example for 2-d maxima problem

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.

8)Pseudo code algorithm for DFS Timestamp?

```
DFS(G)
1 for (each u 2 V)
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)
```

(2) what is common problem in communication networks and circuit designing? 2
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.

(3) write suedo code of relaxing a vertex 5
RELAX((u, v))
1 **if** (d[u] + w(u, v) < d[v])
2 **then** d[v] d[u] + w(u, v)
3 pred[v] = u
Page #155

(4) write suedo code of dijkstra algorithm? 5
Page #156

(5) define NP completeness 5
The set of NP-complete problems is all problems in the complexity class NP for which it is known that if any one is solvable in polynomial time, then they all are. Conversely, if any one is not solvable in polynomial time, then none are.
Definition: A decision problem L is NP-Hard if
L0 _P L for all L0 2 NP.
Definition: L is NP-complete if
1. L 2 NP and
2. L0 _P L for some known NP-complete problem L0.
Page #176

(6) define floyed warshall algorithm in these two cases 5
do not go through a vertex k at all
do go through a vertex k
Don't go through k **at all**
Then the shortest path from i to j uses only intermediate vertices {1, 2, . . . , k − 1}. Hence the length of the shortest is d(k−1)
ij
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, . . . , k − 1}, the length of the path is
d(k−1)
ik + d(k−1)
kj .
Page #162

(7) Define DAG 3
A directed graph that is acyclic is called a *directed acyclic graph (DAG).*

4 Questions of 5 Marks

1) Write steps of sieve techniques

We then *partition* A into three parts.

1. A[q] contains the pivot element x,
2. A[1..q − 1] will contain all the elements that are less than x and
3. A[q + 1..n] will contains all elements that are greater than x.

Page #36

Prove the Lemma:

Consider a diagraph G = (V,E) and any DFS forest for G. G has a cycle if and only if the DFS forest has a back edges

Proof: If there is a back edge (u, v) then v is an ancestor of u and by following tree edge from v to u, we get a cycle.

Where the cliquer cover problem is used?

In clustering applications.

2) Analyze the brute force maxima algorithm

Assume that the input size is n, and for the running time we will count the number of time that any element of P is accessed. Clearly we go through the outer loop n times, and for each time through this loop, we go through the inner loop n times as well. The condition in the if-statement makes four accesses to P. The output statement makes two accesses for each point that is output. In the worst case every point is maximal so these two access are made for each time through the outer loop.

3) How Kruskal's algorithm works ?

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.

4) Which points should be supposed to prove the correctness of the Dijkstra's Algorithm

We will prove the correctness of Dijkstr's algorithm by Induction. We will use the definition that _(s, v)

denotes the minimal distance from s to v.

For the base case

1. S = {s}
2. d(s) = 0, which is _(s, s)

Page #158

What are two steps generally involved while developing a dynamic programming algorithm?

Dynamic programming is essentially recursion without repetition. Developing a dynamic programming algorithm generally involves two separate steps:

• **Formulate problem recursively**. Write down a formula for the whole problem as a simple combination of answers to smaller subproblems.

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

What is the running time of the Dijkstra's algorithm?

_(E log V)

The following adjacency matrix represents a graph that consists of four vertices labeled 0, 1, 2 and 3. The entries in the matrix indicate edge weights.

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

Answer the following question:

Can an adjacency matrix for a directed graph ever not be square in shape? Why or why not?

NO. Because number of rows and number of columns are always same.

What is the application of edit distance technique?

Edit Distance: Applications

There are numerous applications of the Edit Distance algorithm. Here are some examples:

Spelling Correction

If a text contains a word that is not in the dictionary, a 'close' word, i.e. one with a small edit distance, may be suggested as a correction. Most word processing applications, such as Microsoft Word, have spelling checking and correction facility. When Word, for example, finds an incorrectly spelled word, it makes suggestions of possible replacements.

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.

Computational Molecular Biology

Speech Recognition

Algorithms similar to those for the edit-distance problem are used in some speech recognition systems. Find a close match between a new utterance and one in a library of classified utterances.

7. What are the minimum and maximum numbers of elements in a heap of height h?

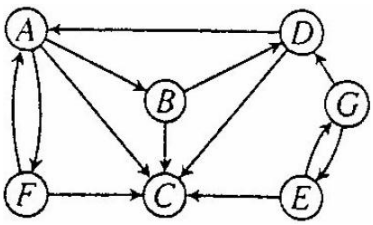
min = 2^h

max = 2^(h+1) - 1

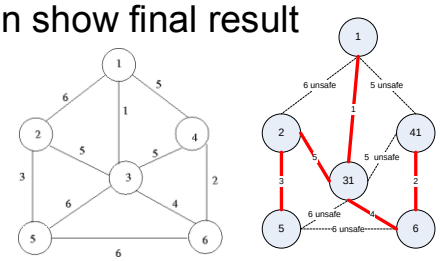
8. Where clique cover problem arises?

The clique cover problem arises in applications of clustering. We put an edge between two nodes if they are similar enough to be clustered in the same group. We want to know whether it is possible to cluster all the vertices into k groups.

9. Show the strongly connected components of the following graph using DFS algorithm. Take node E as a starting node. [You can show final result in exam software and need not to show all intermediate steps].



10. Apply Kruskal's algorithms on the following graph.[You can show final result in exam software and need not to show all intermediate steps].



8. Apply Kruskal's algorithms on the following graph.[You can show final result in exam software and need not to show all intermediate steps].

5) a table is given and we have to make the adjacency list and matrix

6) an mst graph is given and we have to calculate its weigth

Q1) answer the follwing according to Floyd Marshall 1) runing time 2) space use

6) Floyd-Warshall matrix diya tha os py 3 itration apply karny thy 5marks

1- Given a graph run dijikstr's algorithm to find shortest path from vertex "S" to all other vertices (5 marks)

2- Given a graph run prim's algorithm to find minimum spanning tree. Only show the order of edges to be added to make spanning tree (5 marks)

3- Given a graph run DFS and label timestamping (5 marks)

8- Given an adjacency list, make the graph. State whether the graph is cyclic or not? (3 marks)

Is Dijksratra is dynamic problem. Answer with yes or no with reason

Question No: 34 (Marks: 5)

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

Question No: 51 (Marks: 5)

Floyd Warshal algorithm with three recursive steps

5)Floyd-Warshall matrix complexity?

6)Convert Shortest path in to single source shortest path problem?

1) Given a matriz of graph G

Is G directed or undirected graph ?

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

Answer yes or no and give a brief explanation for your choice. If problem A reduces (is polynomial-time reducible) to problem B and B is NP-complete then A is NP-complete

5. Consider the following two problems. In P1 we are given as input a set of n squares (specified by their corner points), and a number k. The problem is to determine whether there is any point in the plane that is covered by k or more squares. In P2 we are given as input an n–vertex graph, and a number k; the problem is to determine whether there is a set of k mutually adjacent vertices. (E.g. for k = 3 we are just looking for a triangle in the graph.). Obviously, the problems are both in NP. There exists a simple translation from P1 to P2: just make a graph vertex for each square, and add an edge between a pair of vertices if the corresponding two squares overlap. If P1 is NP-complete, would this translation imply that P2 is NP-complete? (Give your Answer in Yes or No)