D & A of Algorithms (CS2009)

Date: Dec 21 2024

Course Instructor(s)

Dr. Maryam, Dr. Asim, Miss Abeeda

Final- II Exam

Total Time (Hrs): 3
Total Marks: 49
Total Questions: 8

_____________ Roll No   ____________ Section   _________________________ Student Signature

Instructions: Attempt all questions. Answer in the space provided. Do not attach rough sheets with this exam.

Question	1	2	3-4	5-6	7-8	Total
Marks	/6	/14	/9	/11	/9	/49

CLO #2: *Analyze the time and space complexity of different algorithms by using standard asymptotic notations for recursive and non-recursive algorithms.*

Q1) Usman and Hassan each have come up with a divide-and-conquer algorithm to solve a problem.

- Usman's algorithm (algorithm A) splits a problem of size n into four pieces, each of size n/2, solves the pieces recursively, and takes time 30n to combine the results.
- Hassan's algorithm (algorithm B) takes time n^3 to turn a problem of size n into a smaller problem of size n/2, which it then solves recursively.

(a) Write a recurrence for the runtime of Usman's algorithm, TA(n), and a recurrence for the runtime of Hassan's algorithm, TB(n). [2 Marks]

$TA(n) = 4T(n/2) + 30n$
$TB(n) = T(n/2) + n^3$

(b) Which algorithm has an asymptotically better runtime? Justify your answer by solving each recurrence using the recursion tree method. [4 Marks]

$TA(n) = O(n^2)$
$TB(n) = O(n^3)$

CLO 4: **Implement** the algorithms, compare the implementations empirically, and apply fundamental algorithms knowledge to solve practical problems related to the program.

Q2 (a) Let T be a minimum spanning tree of a graph G. Then for any two vertices u,v the path from u to v in T is a shortest path from u to v in G. Is this statement True or False? Justify your answer with an example. [2 Marks]

False

b) Suppose we have a O(n) time algorithm that finds the median of an unsorted array. Now consider a QuickSort implementation where we first find the median using the above algorithm, and then use the median as the pivot. What will be the worst-case time complexity of this modified QuickSort? [1 Mark]

O(n lg n)

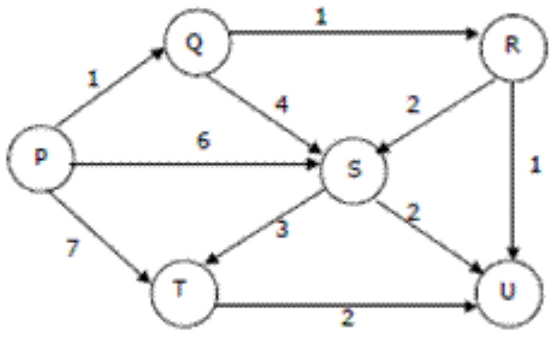
c) When does the worst case of Quick sort occur? [1 Mark]

Solution: When an array is already sorted and the first or last position is selected as pivot.

d) Instead of using counting sort to sort digits in the radix sort algorithm, we can use any valid sorting algorithm and radix sort will still sort correctly. (True/False) [1 Mark]

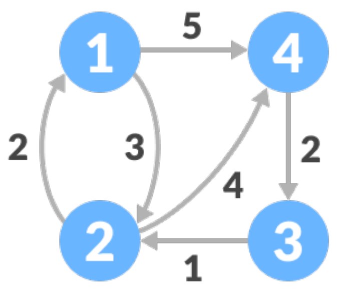
Solution: False. Need stable sort.

(e) Suppose we run Dijkstra's single source shortest-path algorithm on the following edge weighted directed graph with vertex P as the source. In what order do the nodes get included into the set of vertices for which the shortest path distances are finalized? [1 Marks]



a) P Q T S R U
b) P Q T S U R
c) P Q R S U T
d) P Q R U S T
e) P S Q R U T

(f) Consider the graph given below and run Floyd Warshall's algorithm to find all pairs shortest paths. Show the results after 3rd iteration (Fill matrix D^2). For your ease matrices (D^0, D^1) are provided for the first two iterations. The Floyd-Warshall's algorithm uses the following recurrence relation for the shortest path d(i, j, k) from vertex i to vertex j, considering vertex k as an intermediate vertex:
d(i, j, k) = min(d(i, j, k-1), d(i, k, k-1) + d(k, j, k-1)) [2 Marks]



D^0	1	2	3	4
1	0	3	∞	5
2	2	0	∞	4
3	∞	1	0	∞
4	∞	∞	2	0

D^2	1	2	3	4
1	0	3	∞	5
2	2	0	∞	4
3	3	1	0	5
4	∞	∞	2	0

D^1	1	2	3	4
1	0	3	∞	5
2	2	0	∞	4
3	∞	1	0	∞
4	∞	∞	2	0

(g) Suppose we apply RADIX-SORT on an array of n integers in the range $0, 1, \ldots, n^5 - 1$.

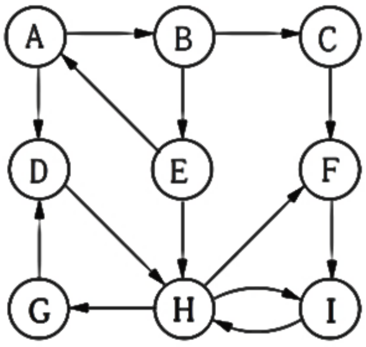
i. What is the running time when using a base-n representation? [1 Mark]

Using base n, each integer has $d = \log n^5 = 5$ digits, so the running time of RADIX- SORT is $\Theta(5n) = \Theta(n)$.

ii. What is the running time when using a base-n representation?

Cancelled as this is same as part 1.

(h) Run the strongly connected components algorithm on the directed graph G, and answer the following questions. Whenever there is a choice of vertices to explore, always pick in alphabetical order. [1 + 1+ 1 = 3 Marks]



I. How many SCCs does this graph have, list the vertices of each SCC?
3
(A, B, E)
(C)
(D, F, G, H, I)

II. How many sink SCCs does this graph have? List the vertices of sink SCCs.
1 (D, F, G, H, I)

III. What is the minimum number of edges you must add to this graph to make it a strongly connected graph? Write the name of the edge or edges.
1 (H, E) or (D, A)

i) If graph G has some negative edge weights, Dijkstra's algorithm does not guarantee computing the shortest paths. However, would finding the minimum weight (most negative weight) w and adding the absolute value to every edge's weight solve this problem? In other words, if we normalize all edge weights by adding a constant absolute value to each edge such that the smallest edge weight in the new graph is 0, then will Dijkstra's algorithm correctly compute the shortest paths on this new graph? (True/False) [1 Mark]

False

j) What is space complexity of count sort? [1 Mark]
O(n + k)
CLO #1: *Design algorithms using different algorithm design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program*

Q3) You are given an extremely large array A in which the first n cells contain non-negative integers in sorted order and the rest of the cells are filled with -1. You do not know the value of n. The size of the entire array is also not known. Describe an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in O (log n) time – again, you do not know the value of n. [5 Marks]

If the array is infinite, then we cannot directly apply binary search due to the unknown upper bound (high). To find the position of integer x, first set the lower and upper bounds, initially low = 0 and high = 1, and compare the key with the high index element, if it is smaller, then apply the binary search on elements between high and low indices, if it is greater than the high index element then update the low = high and high = high*2 move forward by doubling the high (2, 4, 6, 8 ...).

Q4) You have N homework due right now, but you haven't started any of them, so they are all going to be late. Each homework requires d_i days to complete, and has a cost penalty of c_i per day. So if homework i ends up being finished t days late, then it incurs a penalty of $t.c_i$. Assume that once you start working on a homework, you must work on it until you finish it, and that you cannot work on multiple homework at the same time.
For example, suppose you have three homework: HW1 takes 3 days and has a penalty of 12 points/day, HW2 takes 4 days and has a penalty of 20 points/day, and HW3 takes 2 days and has a penalty of 4 points/day. The best order is then HW2, HW1, HW3 which results in a penalty of
$20 \cdot 4 + 12 \cdot (4 + 3) + 4 \cdot (3 + 4 + 2) = 200$ points.
Give a greedy algorithm that outputs an ordering of the homework that minimizes the total penalty for all the homework. Analyze the running time. [4 Marks]

Solution: Sort by increasing d_i/c_i and do the homework in that order. This takes O(n log n) time.

Q5) Given a set of n people, the problem is to determine how many different ways you can select a committee of k members, which means finding the number of possible k-element subsets. The order in which you choose the people who end up on the committee doesn't matter: that is, first choosing Alice, then Bob, and then Carol is the same as first choosing Carol, then Bob, and then Alice.

The base case when k = 0 or the value of k and n is equal
$C(n, 0) = C(n, n) = 1$
$C(n, k) = C(n-1, k-1) + C(n-1, k)$

i. A recursive code is provided below what is the time complexity of this function in terms of Big-O? [1 Mark]

```
int C (n, k) {
    if (k > n)
      return 0
     if (k == 0 || k == n)
      return 1
   return C (n - 1, k - 1) + C (n - 1, k);
}
```

Exponential T(n) = O(2^n)

ii. Write the iterative Dynamic Programming solution with memoization for this problem and provide the time complexity of your solution. [5 Marks]

```
int C (n, k) {
      if (k > n)
        return 0;
      mem[n+1][k+1] = {-1} //mem initialized with -1 values
      for i =0 to n
            for j = 0 to min(k,i)
                  if(j==0 || j==i)
                        mem[i][j] = 1
                  else
                        mem[i][j] = Mem[i-1][j-1] + Mem[i-1][j]
      return mem[n][k];
}
```

Time = O(n^2)

Q6) The power company needs to lay distribution lines connecting the six cities below to the power grid. How can they minimize the amount of new lines to lay? The following table presents the length of the line between each pair of city.

a) Give an efficient algorithm for solving this problem. Note: If you are using an algorithm discussed in class as a subroutine, you can simply call it and do not need to include its pseudocode. However, if you need to modify the algorithm, you must provide its pseudocode. [4 Marks]

b) Dry run the algorithm given in part (a) on the given problem and calculate the amount of new lines to lay. [1 Mark]

	A	**B**	**C**	**D**	**E**	**F**
A	0	4	7	2	9	3
B	4	0	6	5	8	1
C	7	6	0	10	11	12
D	2	5	10	0	15	13
E	9	8	11	15	0	14
F	3	1	12	13	14	0

Solution:
This problem can be solved by mapping to a graph. Use any MST algorithm to minimize the amount of lines to lay.
If we apply Kruskal, we get the following MST edges with total length = 20:
(B, F) = 1
(A, D) = 2
(A, F) = 3
(B, C) = 6
(B,E) = 8

Q7) Given a directed unweighted graph G= (V, E) with vertex set V = {1, 2, …, n}. We want to determine whether G contains a path from vertex i to vertex j for all vertex pairs (i, j).
How you will use the BFS algorithm for this problem: explain your answer in 2 to 3 lines. What is the time complexity of this solution in terms of Big-θ?
Note: If you are using BFS as a subroutine, you can simply call it and do not need to include its pseudocode. However, if you need to modify the BFS algorithm, you must provide its pseudocode. [4 Marks]

Run BFS total V times by taking every vertex as a source and check which vertices are visited marking them 1 and unvisited 0.

Adjacency List = $O(V(V + E)) = O(V^2 + VE)$

Q8) Write an efficient algorithm for solving the following graph problem.
Input: a directed graph G = (V,E), with a positive length l(e) on each edge e; vertices s,w,t
Output: the shortest path length from *s* to *t* that goes through *w*. [5 Marks]
Note: If you are using an algorithm discussed in class as a subroutine, you can simply call it and do not need to include its pseudocode. However, if you need to modify the algorithm, you must provide its pseudocode.
Solution:
1. Call Dijkstra(G,l,w), to get $d(w,v)$ for each $v \in V$.
2. Reverse the direction of all the edges; call the result *Gr*.
3 Call Dijkstra(Gr,l,w), to get $d(v,w)$ for each $v \in V$.
4. Return $d(s,w)+d(w,t)$.