SRM Institute of Science and Technology Set A
College of Engineering and Technology
School of Computing

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

**Academic Year: 2021-22 (Even)**

| | |
|---|---|
| **Test: CLA-T2** | **Date: 27-05-2022** |
| **Course Code & Title: 18CSC204J Design and Analysis of Algorithms** | **Duration:** 100 min |
| **Year & Sem: II Year / IV Sem** | **Max. Marks:** 50 |

| Q. No | Part A |
|---|---|
| 1 | The best-case and worst-case complexities of Binary Search are<br>a) O(1) and O(log n)<br>b) O(1) and O(n)<br>c) O(log n) and O(log n)<br>d) Both best and worst case is O(log n) |
| 2 | Algorithm A1 can compute min-max in a1 comparison without divide and conquer. Algorithm A2 can compute min-max in a2 comparison with divide and conquer. What could be the relation between al and a2 considering the worst-case scenarios?<br>a) a1 < a2<br>b) a1 > a2<br>c) a1 = a2<br>d) Depends on the input |
| 3 | Time complexity of Strassen's matrix multiplication problem is:<br>a) $T = \theta(N^{\log 2})$<br>b) $T = \theta(7^{\log 2})$<br>c) $T = \theta(7^{\log n})$<br>d) $T = \theta(N^{\log 7})$ |
| 4 | Find the Euclidean distance between the po<br>ints (4,3) and (7,5)<br>a) $\sqrt{3}$<br>b) $\sqrt{13}$<br>c) $\sqrt{19}$<br>d) $\sqrt{5}$ |
| 5 | Which algorithm strategy builds up a solution by choosing the option that looks the best at every step.<br>a) greedy method<br>b) branch and bound<br>c) dynamic programming<br>d) divide and conquer |
| 6 | The total running time of fractional Knapsack problem using simple approach.<br>a) O(n)<br>b) O(log n)<br>c) $O(2^n \log n)$ |

| | |
|---|---|
| | d) $O(2^n)$ |
| **7** | Suppose the letters a, b, c, d, e has the probabilities 0.3, 0.3, 0.2, 0.1, 0.1, which of the following is the Huffman code for the above letters?<br>a) 100, 110, 00, 10, 01<br>b) 10, 11, 00, 010, 011<br>c) 100, 110, 11, 10, 11<br>d) 10, 111, 00, 01, 01 |
| **8** | Dynamic programming is characterized by<br>(1) Distinct sub-problems<br>(2) Overlapping sub-problems<br>(3) Optimal substructures<br>(4) Sub-optimal or near optimal substructures<br>a) 1 and 3<br>b) 2 and 3<br>c) 1 and 4<br>d) 2 and 4 |
| **9** | Time complexity of Longest common subsequence using dynamic programming is<br>a) O(m!)<br>b) O(mn)<br>c) O(n!)<br>d) O(n) |
| **10** | Let X = ABRACADABRAAB and Y = YABBADABBADOO. The Longest Common Subsequence, Z, of X and Y is<br>a) Z = DAB<br>b) Z = AAB<br>c) Z = ABADABA<br>d) Z = ABADABAAB |
| | **Part B** |
| **11** | What is Master theorem to solve recurrence relations and solve the below relations using master method.<br>a) $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$<br>b) $T(n) = 7T\left(\frac{n}{2}\right) + n^2$<br><br><u>Solution:</u><br>a)<br>$T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$<br>$a = 2 \quad b = 4 \quad f(n) = \sqrt{n}$<br>$n^{\log_b a} = \sqrt{n} \quad f(n) = n^{\log_b a}$<br>$\therefore \; T(n) = \Theta(\sqrt{n} \lg n)$<br><br>b) |

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$$a = 7 \quad b = 2 \quad f(n) = n^2$$

$$n^{\log_b a} = n^{\log 7} \approx n^{2.8} \quad f(n) = O(n^{\log 7 - \epsilon}) \text{ where } \epsilon \approx 0.8$$

$$\therefore \; T(n) = \theta(n^{\log 7})$$

**12** Develop a straightforward and recursive algorithm using divide and conquer to find a maximum and minimum number in a set of n elements. Explain with an example.

<u>Solution:</u>

Straightforward Method:

```
Algorithm: Max-Min-Element (numbers[])
max := numbers[1]
min := numbers[1]

for i = 2 to n do
if numbers[i] > max then
max := numbers[i]
if numbers[i] < min then
min := numbers[i]
return (max, min)
```

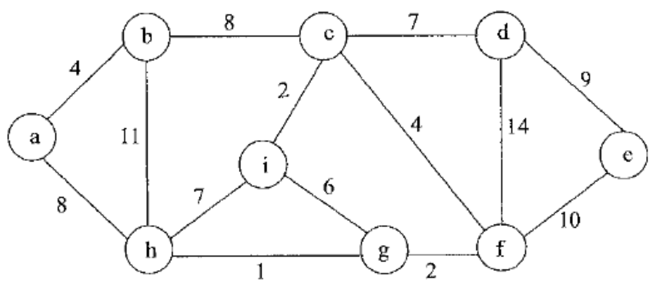
Divide and conquer method:

```
Algorithm: Max - Min(x, y)
if y – x ≤ 1 then
return (max(numbers[x], numbers[y]), min((numbers[x], numbers[y]))
else
(max1, min1):= maxmin(x, ⌊((x + y)/2)⌋)
(max2, min2):= maxmin(⌊((x + y)/2) + 1)⌋,y)
return (max(max1, max2), min(min1, min2))
```

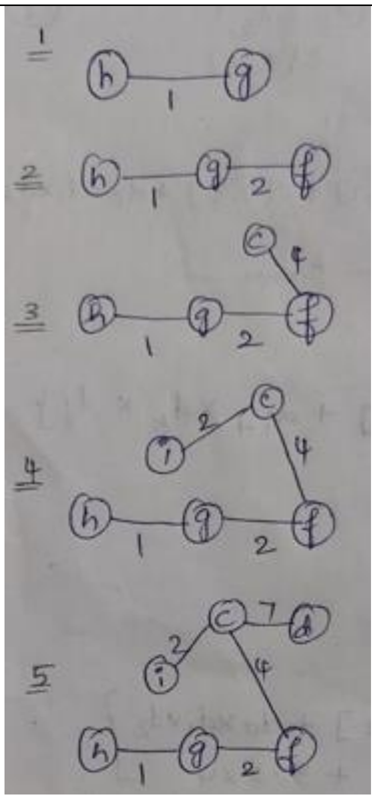
Compared to Naïve method, in divide and conquer approach, the number of comparisons is less.

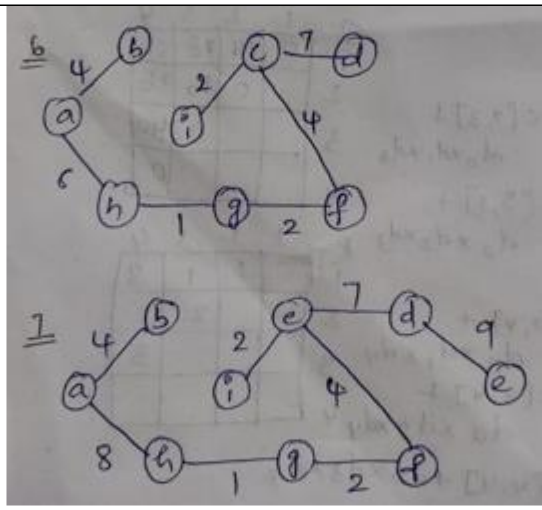
Examples to be given.

**13** Determine the minimum cost spanning tree using Kruskal's method.




<u>Solution:</u>




Cost = 37

**14** Explain 0/1 Knapsack problem. Solve the following problem using Dynamic programming. Max weight = 7, (P1, P2, P3, P4) = (1, 4, 5, 7) and (W1, W2, W3, W4) = (1, 3, 4, 5).

<u>Solution:</u>

| max wt = 7 | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| $P_i$ | $w_i$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 3 | 2 | 0 | 1 | 1 | 4 | 5 | 5 | 5 | 5 |
| 5 | 4 | 3 | 0 | 1 | 1 | 4 | 5 | 6 | 6 | 9 |
| 7 | 5 | 4 | 0 | 1 | 1 | 4 | 5 | 7 | 8 | 9 |

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k]+b_k\} & \text{else} \end{cases}$$

Items: {I1, I2, I3, I4} = {0, 1, 1, 0}

**15** Construct an Optimal Binary Search Tree for the given values:

| Key | A | B | C | D |
|---|---|---|---|---|
| Probability | 0.1 | 0.2 | 0.4 | 0.3 |

<u>Solution:</u>

$$C[i,j] = \min_{i \le k \le j} \{C[i,k-1] + C[k+1,j]\} + \sum_{s=i}^{j} p_s \text{ for } 1 \le i \le j \le n.$$

Example computations of C[1,2], C[1,3], C[1,4] to be given.

E.g.:

$$C[1,2] = \min \begin{cases} k=1: C[1,0] + C[2,2] + \sum_{s=1}^{2} p_s = 0 + 0.2 + 0.3 = 0.5 \\ k=2: C[1,1] + C[3,2] + \sum_{s=1}^{2} p_s = 0.1 + 0 + 0.3 = 0.4 \end{cases} = 0.4$$

main table

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1 | 0 | 0.1 | 0.4 | 1.1 | 1.7 |
| 2 | | 0 | 0.2 | 0.8 | 1.4 |
| 3 | | | 0 | 0.4 | 1.0 |
| 4 | | | | 0 | 0.3 |
| 5 | | | | | 0 |

root table

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1 | | 1 | 2 | 3 | 3 |
| 2 | | | 2 | 3 | 3 |
| 3 | | | | 3 | 3 |
| 4 | | | | | 4 |
| 5 | | | | | |

Optimal Binary Search Tree: