Name Roll number Marks:

CS4050 (and CSU 301) DESIGN AND ANALYSIS OF ALGORITHMS

MIDSEM TEST II - MONSOON SEMESTER 2014

Time: 1 Hour **Marks: 20**

All questions are compulsory,

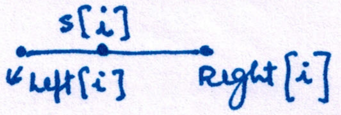
Answer only in the space provided after the question. Additional Sheets are for ROUGH WORK ONLY

Q1. A very long heavy iron chain (weighing tonnes!) has to be cut into a number of pieces. However cutting involves lifting the chain, the cost of which is proportional to the weight of the chain, which is proportional to its length (hence assume cost of a cut=length of the chain before the cut). The integral length of the chain is given, and the places in the chain which are to be cut are specified by integers representing length from the start of the chain. However we can observe that the order of cutting affects the cost. For example, if length of the chain is 10, and the cuts are to be made at 2 and 6, then it may be possible to cut in two ways. Either cut at 2 first and at 6 next, resulting in a total cost of 10+ 8=18. Or cut it at 6 first and then at 2, resulting in a cost of 10+6=16. Thus the second option is better. (In case more explanation is needed --- to make the first cut, you need to lift the whole chain, whereas in order to make the next cut, you only have to lift the relevant piece)

The problem can be stated as, given the chain length L and a set S of n integers (positions of cuts), find the optimal order of cutting, i.e., the permutation of S which minimizes the cost of cutting.

a. Outline a brute force algorithm to solve the problem and analyze its complexity [2]

Assume that there is a function to generate different permutations and it returns the permutations into an array P[1..n]. Thus P is a permutation of S. Right[ ] & Left[ ] store the left end & right end distances of each cut's segments.



Brute-force (L,S)

```
1.  n = S.length
2.  COST = ∞
3.  Best-permutation = [0,0....0]
4.  For each permution P of S
5.      temp=0
6.      for i=1 to ~~n do~~
7.          Left[i]=0
8.          Right[i]=L
9.      for i=1 to n
10.         temp = temp + right[i] - left[i]
11.         for j = i+1 to n
12.             if s[i] < s[j]
13.                 left[j] = s[i]
14.             else right[j] = left[i]
15.     if temp < COST
```

Complexity.
Assuming O(1) amortized time for generating a permutation, it is $O(n!\,n^3)$

$\swarrow$ outer loop (lno:4) $\searrow$ (loop at line no:9)

b. Identify the optimal substructure in the problem and prove it. [2]

Consider the sorted sequence of the required cuts $S[1..n]$. Let $S[0] = 0$ and $S[n+1] = L$ (to formally represent the end points of the original string)

Then, $S = (S_0 \; S_1 \; S_2 \; \dots \; S_{n+1})$

Let S_i be the first cut in the optimal solution.

to S, called $B[1..n]$ i.e $B_1 = S_i$

Consider the subproblems $(S_0 \dots S_i)$ and $(S_i \dots S_{n+1})$

The solutions to these subproblems contained i.e subsequences of B, in the original solution, must be optimal solutions to $(S_0 \dots S_i)$ and $(S_i \dots S_{n+1})$ respectively as otherwise one get ~~g~~ a better ~~optimal~~ solution to S (changing the order of cuts in one segment will not affect the cost of cuts in another segment).

c. Write a recursive solution to the problem and analyze its time complexity [2]

Let $S[0 \dots n+1]$ be the sorted sequence of cuts as above. Let $M[i,j]$ denote the minimum cost of cutting a string with left end at $S[i]$ and right end at $S[j]$. Thus $M[0..n+1]$ is to be found.

$$M[i,j] = \begin{cases} i > j, 0 \\ i = j, 0 \\ i = j-1, 0 & \text{\# because no cut is to be made there.} \\ \min\limits_{k=i+1}^{j-1} \Big(\underbrace{S[j] - S[i]}_{\text{cost of}} + \underbrace{M[i,k]}_{\text{opt solution}} + \underbrace{M[k,j]}_{\text{opt solution}} \Big) \end{cases}$$

Let n = ~~S.length~~ no: of cuts to be made.

$T(n) \leq T(1) + T(n-1) + T(2) + T(n-2) + \cdots T(n-1) + T(1)$

$\leq 2(T(1) + T(2) + \cdots T(n-1))$

$\leq 4(T(1) + T(2) + \cdots T(n-2))$

$\leq 8(T(1) + T(2) + \cdots T(n-3))$

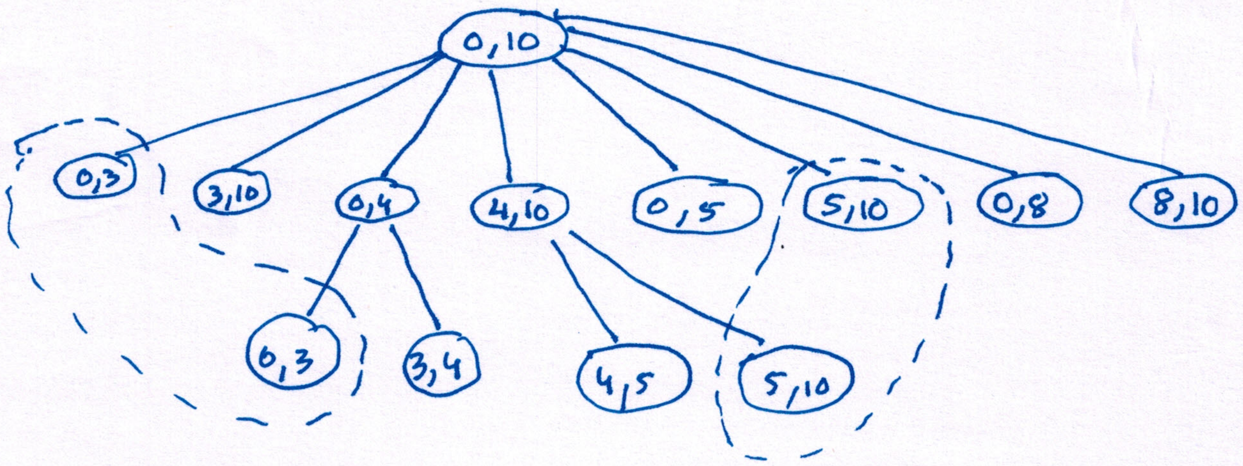
$\leq 2^{n:}(T(n))$ Hence $O(2^n)$.

d. Does the problem exhibit repeated subproblems in the recursive solution. Demonstrate. [2]

Let L = 10

S = [3, 4, 5, 8]

An example



e. Formulate a bottom up dynamic programming solution to the problem [2]

```
Dynamic Programming ~~(S)~~ (S, L)
1. sort S[1..n]  # where n = S.length
2. S[0] = 0
3. S[n+1] = L
4. Initialize arrays M[0..n+1][0..n+1] and B[0..n+1][0..n+1]
   to ∞ and 0 respectively.
5. for i = 0 to n        # f is offset
6.         M[i, i] = 0
7.         M[i, i+1] = 0
8. for f = 2 to n+1
9.         for i = 0 to n+1 - offset
```