ASSIGNMENT : HW1

NAME : PRASANTH BHAGAVATULA

CWID : A20355611

FALL 2015

1. (a) The operation STRING-SPLIT (S, k, f, b) is used to cut/split a string 's' into two pieces. The first piece, called 'f', will have first 'k' characters of 's' and the remaining characters will be called as 'b'. Our aim is to cut the string 's' into multiple pieces with the position of cuts provided as input. Note that, cutting a string of 'n' character length requires copying of 'n' characters and the length of string 's' is represented as 'O(|s|)'.

Lets say $k_1, k_2, k_3 \cdots k_m$ be the positions of cuts to be made, arranged in ascending order, on the string

$S_i\ S_{i+1}\ S_{i+2} \cdots S_j$. S_0, the optimal cost of cutting this string into two pieces at position k_l is defined as

$$f(S_{ij}, k_l, f_l, b_l) = \begin{cases} O(|S_{ij}|) + f(f_l, k_{l-1}, f_{l-1}, b_{l-1}) \\ \quad + f(b_l, (k_{l+1} - k_i), f_{l+1}, b_{l+1}); \text{ if } i \le k_l < j \\ O(|S_{ij}|) \text{ ; if } k_l = 0 \text{ or } k_l = j \text{ (i.e. length of input string)} \end{cases}$$

The optimal cost of cutting string 's' into 'm' pieces is ~~cutting~~ involves three steps.

(a) cutting the ~~and~~ string ~~as~~ into two pieces
(b) cutting the first sub-string optimally
(c) cutting the second sub-string optimally

If the position of the next cut of the sub-string is "0" or same as the length of the sub-string, then it there is no cut can be performed and hence the cost will be ~~0 if~~ $O(|s_{ij}|)$.

Now, replacing 'i' and 'j' with 'l' and 'r'

$$f(s, k_l, f_l, b_l) = \begin{cases} 0|s| + f(f_l, k_{l-1}, f_{l-1}, b_{l-1}) \\ \quad + f(b_l, k_{l+1} - k_l, f_{l+1}, b_{l+1}); \text{ if } 1 \le k_l \le n \\ 0|s|; \text{ if } k_l = 0 \text{ or } k_l = n \end{cases}$$

Below is the algorithm

```
STRING-SPLIT(s, k, f, b)
1.  IF k_l == 0 OR k_l == s.length
2.      RETURN O(|s|)
3.  C = ∞
4.  FOR i = l to s.length
5.      C = min(C, STRING-SPLIT(f, k_{i-1}, f_{i-1}, b_{i-1})
                + STRING-SPLIT(b, k_{i+1} - k_i, f_{i+1}, b_{i+1})
                + O(|s|))
6.  RETURN C.
```

ASSIGNMENT : HW1

NAME : PRASANTH BHAGAVATULA

CWID : A20355611

FALL 2015

The time taken/cost by of this algorithm is similar to that of the ROD-CUT problem mentioned in the text book in Chapter 15. as since this involves splitting the string into two-halves and so we can create a binary tree out of it. The running time of this algorithm is

$$T(n) = O(2^n).$$

1.(b) Since, we are recursively calling the STRING-SPLIT function and the time/cost for splitting a string of 'n' character is $O(n)$, we might end up calculating the cost for an already calculated sub-string length.

Consider an example wherein, we are splitting a string of 50 characters on after 25 characters. So, the cost of this split is '50' characters. Now, if we are further split the first and second sub-strings, we can simply calculate the cost of splitting the first string and the same can be applied for the second sub-string because the size of both strings is same (25 characters)

So, we memoize the cost ~~incurred~~ incurred for cutting/splitting a string of 'n' characters in an array 'L', at position 'n'. So, the modified algorithm (memoized algorithm) is.

```
STRING-SPLIT-MEMOIZED ( S, k_i, f, b)
1.  IF L[S.LENGTH] = exists
2.      return L[S.LENGTH]
3.  ELSE
4.      IF k_i == 0  OR  k_i = S.LENGTH
5.          RETURN Θ(|s|)
6.      ELSE
7.          c = ∞
8.          FOR i = 1 to s.length
9.              c = min (c , STRING-SPLIT-MEMOIZED(f, k_{i-1}, f_{i-1}, b_{i-1})
                          + STRING-SPLIT-MEMOIZED(b, k_{i+1} - k_i, f_{i+1}, b_{i+1})
                          + Θ(|s|))
10.             ~~L[S.LENGTH] = c~~
10.         L[S.LENGTH] = c
11.         RETURN c
```

This is similar to the TOP-DOWN-APPROACH of the ROD-CUT problem and the cost of their algorithm would be

$$T(n) = \Theta(n^2)$$

ASSIGNMENT : HW1

NAME: PRASANTH BHAGAVATULA FALL 2015

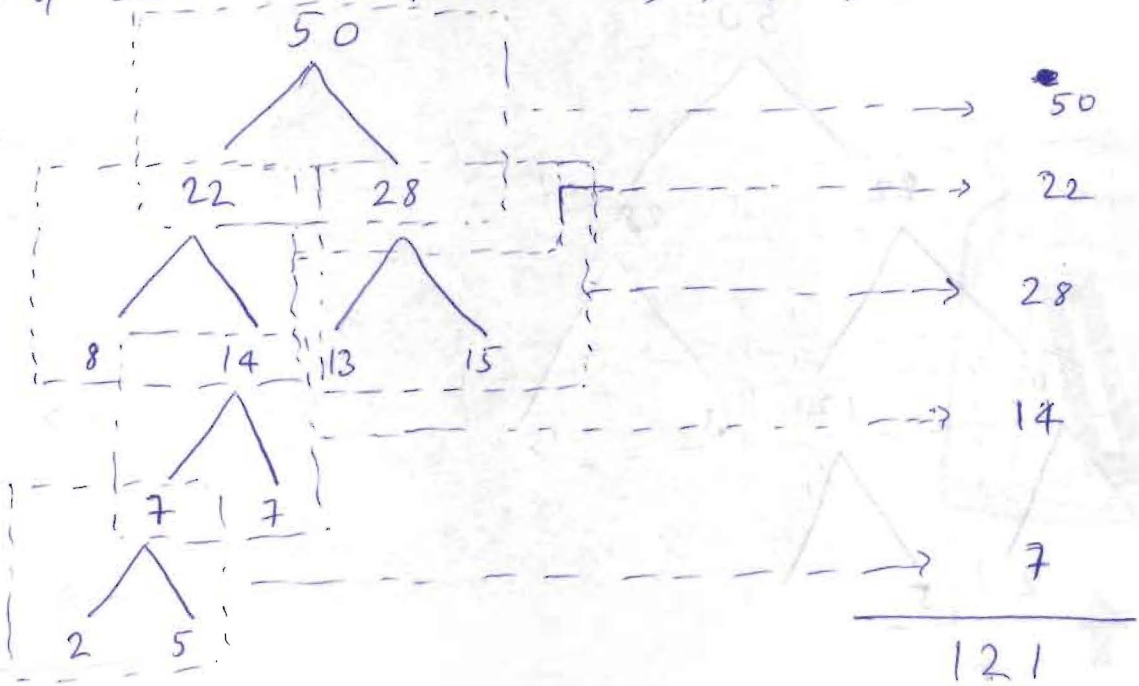
CWID: A20355611

(2) ~~First~~ Consider a string of 50 characters and we make cuts at positions ~~3~~ 8, 10, 15, 22, 35. First we will calculate the cost of optimal cuts and then we will go for each and every order given.

Optimal cost.

We will represen the cuts in a binary-tree format.

The order of cuts are at position 22, 8, 35, 15, 10.



The total cost of this cut order is 121

(a) Cutting the string and sub-strings closes to middle. So, the order of cuts have to be

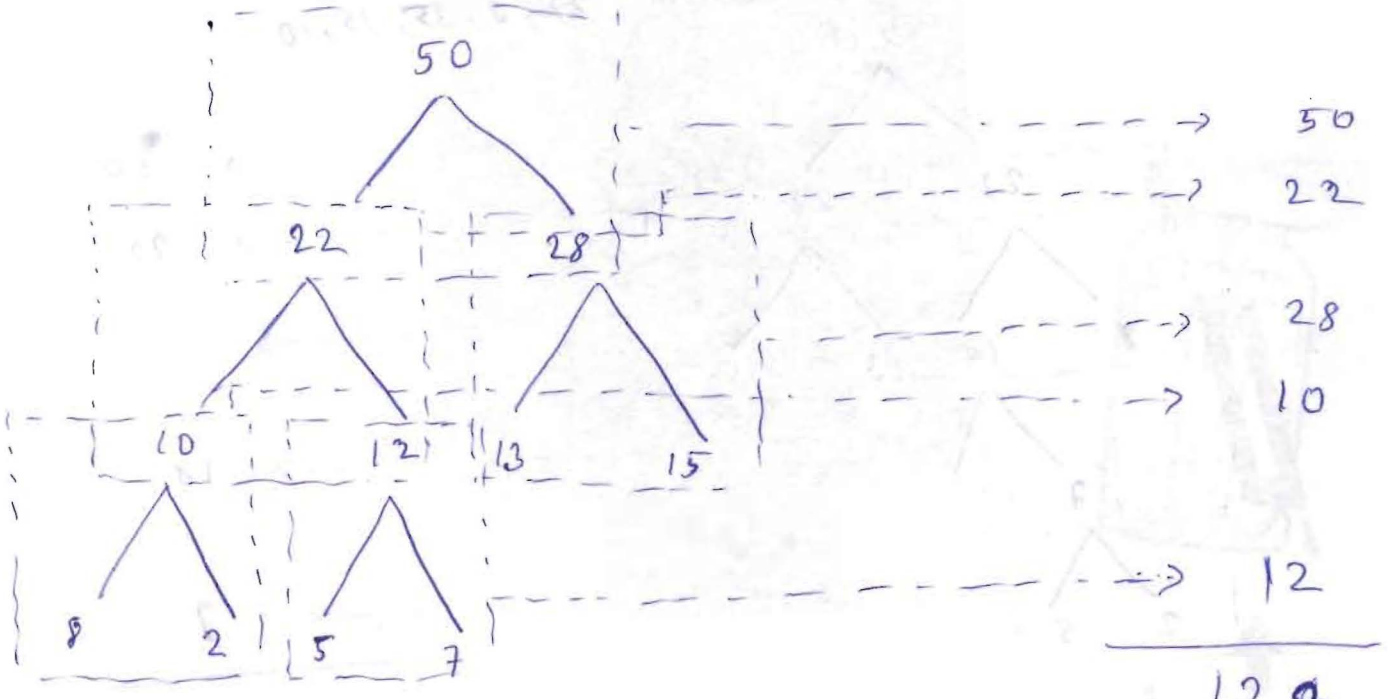
(a) cut 1 at 22

(b) cut 2 at 10 (first sub-string is of 22 ~~bytes~~ characters)

(c) cut 3 at 35 (second substring is of 28 characters)

(d) cut 4 at 8 (The only one cut ~~and~~ can be made for string of ~~first~~ 10 character)

(e) cut 5 at 15 (only one cut can be made for string from 11th character to 22nd character)



122

ASSIGNMENT : HW1

FALL 2015

NAME : PRASANTH BHAGAVATULA

CWID : A20355611

② (b) We need ~~can~~ make at most 2 cuts, to seperate out the smallest substring. For the example, the smallest sub-string is of length 2 character ~~bytes~~ and is from 9th character to 10th character.

The order of cuts are.

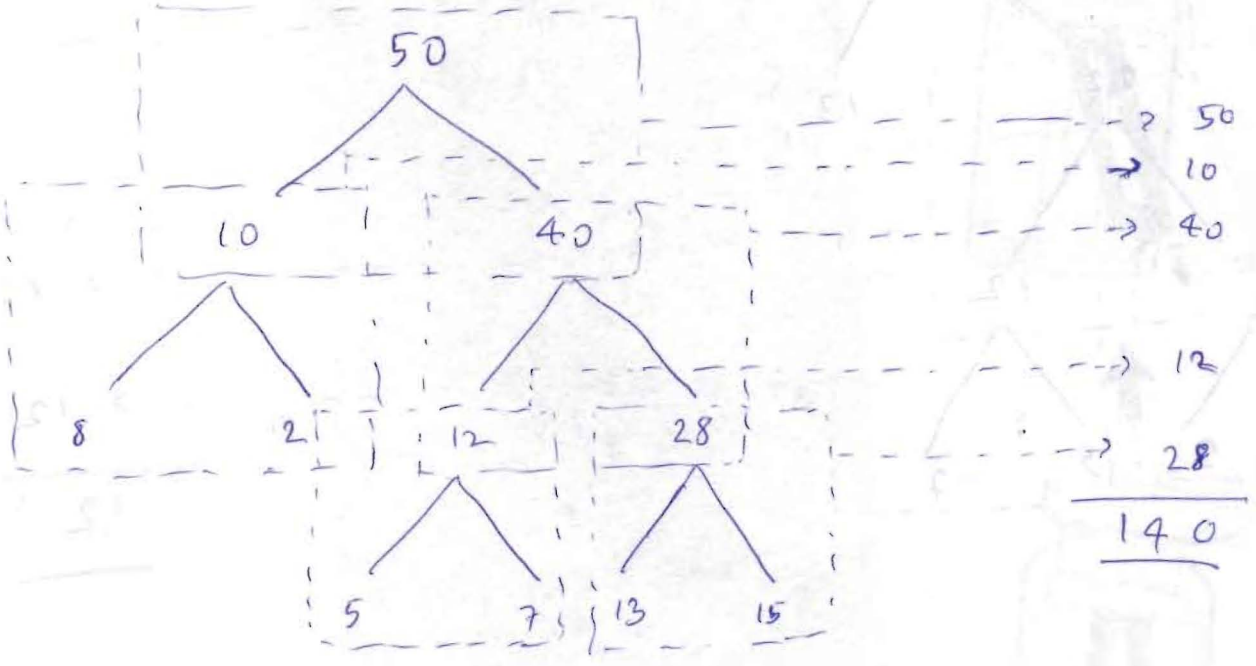
(a) cut 1 at position 10

(b) cut 2 at position 8 (obtained the smallest sub-string)

(c) cut 3 at position ~~8~~ 22

(d) cut 4 at position 15

(e) cut 5 at position 35



(2) (c) We need to make atmost 2 cuts, to get the largest sub-string

In the given example the largest substring is of length 15 and is from position 36 to 50. ~~The example provided in solution 2(a) serves as a solution to this problem as if we make the cut at 35th position before making cut at position 8 &~~

Consider the cuts made at positions 35, 22, 10, 8, 15.

So the cost of these cuts