National University of Computer & Emerging Sciences, Karachi

Fall-2022 Department of Computer Science

Mid Term-2 Solution

03 November 2022, 11:30 AM – 01:00 PM

Course Code: CS2009	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Dr. Farrukh Saleem, Dr. Waheed Ahmed, Miss Anaum Hamid, Miss Aqsa Zahid, Sohail Afzal	
Student Roll No:	**Section:**

Time: 90 min **Max Marks: 17.5**

<u>Question # 1</u> **(15 min)** **[3 marks] [CLO 4]**

Find the Shortest Common Super sequence (SCS) for the strings: X=NOOR, Y=ABDULLAH.
First dry run to find the solution of Longest Common Subsequence (LCS) and then further dry run the below algorithm to find the SCS.

```
//Let m and n contain the length of strings X and Y,
and the matrix dp[m + 1][n + 1] contains the LCS
solution.

string printShortestSuperSeq(string X, string Y) {
string str;     int i = m, j = n;     while (i > 0 && j >
0) {         if (X[i - 1] == Y[j - 1]) {
str.push_back(X[i - 1]); i--, j--; }         else if (dp[i
- 1][j] > dp[i][j - 1]) {             str.push_back(Y[j -
1]); j--; }         else{
            str.push_back(X[i - 1]); i--; } }

   while (i > 0) {
       str.push_back(X[i - 1]); i--; }
while (j > 0) {
       str.push_back(Y[j - 1]); j--; }

   reverse(str.begin(), str.end());
return str;}
```

<u>Solution:</u>

Grading scheme: (Two parts, LCS = 1.5, SCS = 1.5) -
If dry run of LCS is shown: 1.5 marks

- If only answer of SCS is shown i.e. NOORABDULLAH or ABDULLAHNOOR, then +0.5mark to +1 mark.
- If length of desired SCS is shown i.e. 12 in this case, and also if all intermediate steps, either in terms of arrows traversed (as in table below) or by showing values at each iteration of loops, then full marks of this part (i.e. 1.5 marks)

Length of SCS = Total length of all substrings – Length of LCS
Length of SCS = (4 + 8) – 0
Length of SCS = 12

LCS Table:

	j	0	1	2	3	4	5	6	7	8
i			A	B	D	U	L	L	A	H
0		0	0	0	0	0	0	0	0	0
1	N	0	0	0	0	0	0	0	0	0
2	O	0	0	0	0	0	0	0	0	0
3	O	0	0	0	0	0	0	0	0	0
4	R	0	0	0	0	0	0	0	0	0

The SCS solution will be traversed, by only two highlighted conditions shown above.

Answer: NOORABDULLAH

In case of slight error, by misunderstanding, **> as > =**, i.e. (dp[i - 1][j] >= dp[i][j - 1]) then answer is: ABDULLAHNOOR. This is also considered right.

Question # 2 **(5 min)** **[2 marks] [CLO 1]**

Suppose we have two different searching algorithms having complexity of $2n^2$ and $100n$. We are running $2n^2$ complexity algorithm on Computer A (Intel i7 10-core) which executes 200 billion instruction per second and we are running $100n$ complexity algorithm on Computer B (Intel i860) which execute only 50 million instruction per second. Suppose Input length is 1 billion; identify the time taken by Computer A and Computer B.

Solution:

Computer A takes:

$$\frac{2 \times (10^9)^2 \text{ Instructions}}{200 \times 10^9 \text{ Instructions/second}} = 10^7 \text{ seconds (277 hours)}$$

Computer B takes:

$$\frac{100 \times (10^9) \text{ Instructions}}{50 \times 10^6 \text{ Instructions/second}} = 2 \times 10^3 = 2000 \text{ seconds (33 minutes)}$$

Question # 3 **(15 min)** **[2.5 marks] [CLO 3]**

a) Suppose we are given two unsorted integers sequences *A* and *B*, which may contain duplicate entries. Design an ***O(n)*** time algorithm for computing an integer sequence representing the set AUB (with no duplicates) where U stands for union.

> **Example:** Suppose A is [22, 350, 1, 350, 22, 350, 1] and B is [31, 13, 350, 35, 111, 22]. The resulting list we would like to have is [1, 13, 22, 31, 111, 350]

b) Discuss space complexity of proposed algorithm briefly

Solution

Sort integer arr1[] and arr2[] using Bucket Sort or Count Sort. This will take O(n) time. Combine two sorted array (Avoiding repeated element) in Linear Time: Steps are given below:

I. Use two index variables i and j, initial values i = 0, j = 0
II. If arr1[i] is smaller than arr2[j] then print arr1[i] and increment i.
III. If arr1[i] is greater than arr2[j] then print arr2[j] and increment j.
IV. If both are same then print any of them and increment both i and j. V. Print remaining elements of the larger array.

Question # 4 **(10 min)** **[1+1 = 2 marks] [CLO 4]**

1) Figure (a) shows graph of game of Pick up Sticks. An edge indicates that one sticks overlaps another. Write valid order for picking up the sticks
2) Figure (b) shows recipe steps to make pasta. Write valid order for making this dish

Figure (a)

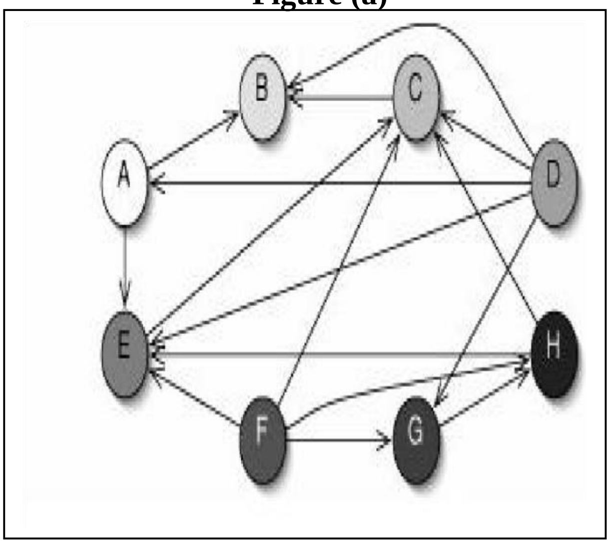
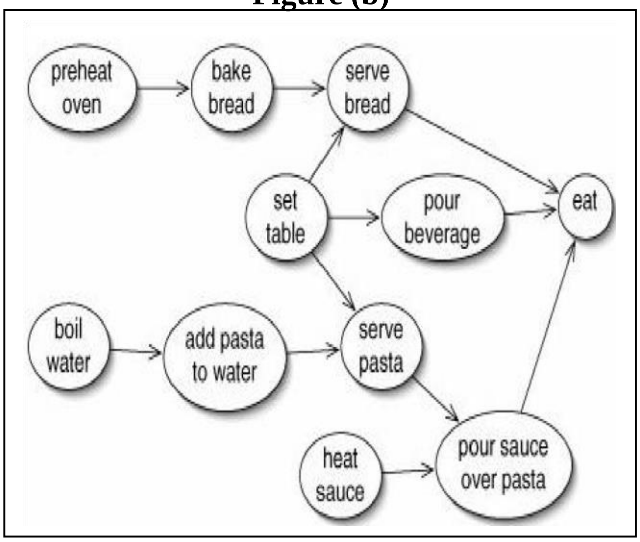
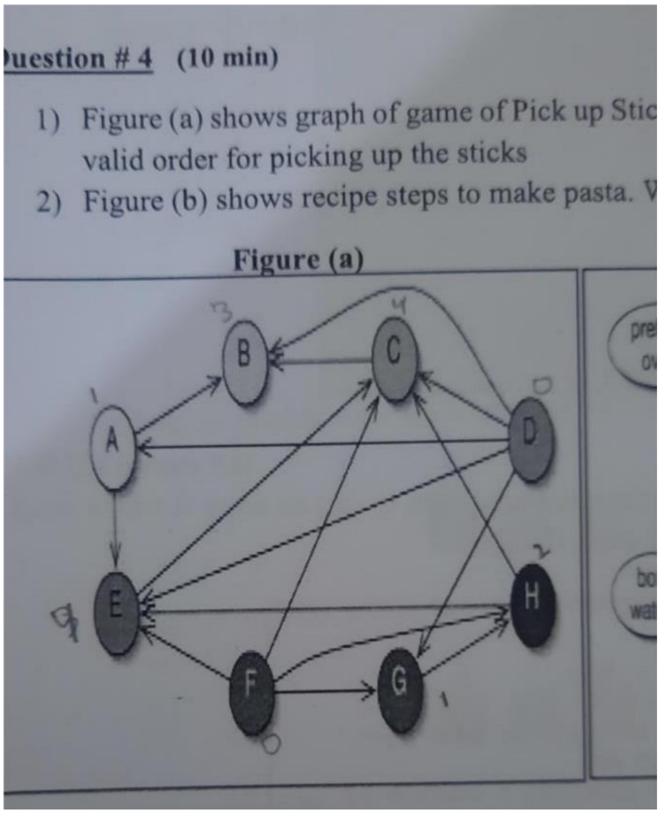


Figure (b)



a)

Expected Order:

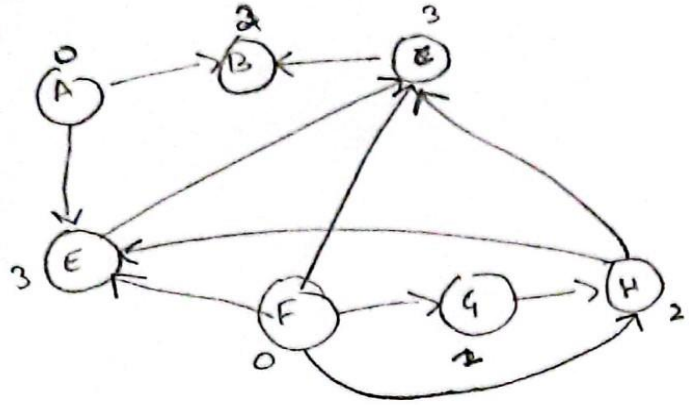
- **D,F,G,H,A,E,C,B**
- **F,D,G,A,H,E,C,B**
- **D,A,F,G,H,E,C,B**
- 

0 inorder nodes :- D, F

chose D

D

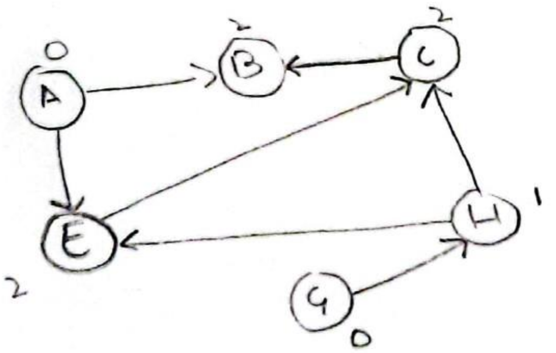
1



2 inorder degree 0 nodes :- A, F

chose F

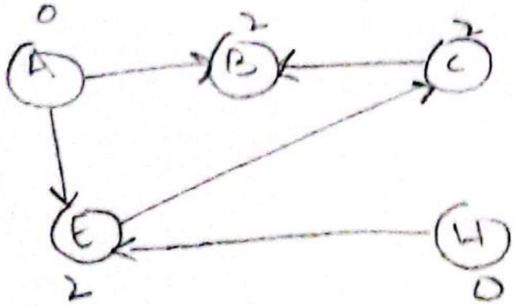
D, F



3 inorder degree 0 nodes A, G

chose G

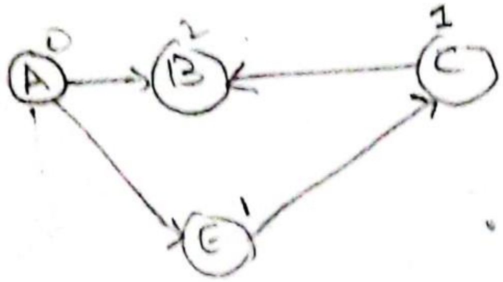
D, F, G



(4) inorder degree 0 nodes

chose H

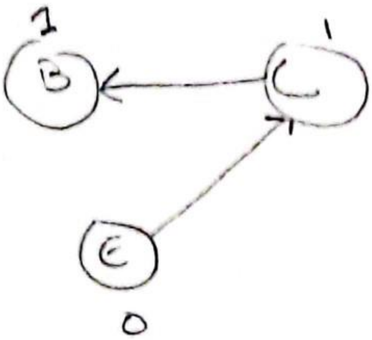
A, H



D,F, G, H

(5) inorder degree 0 nodes

chose A

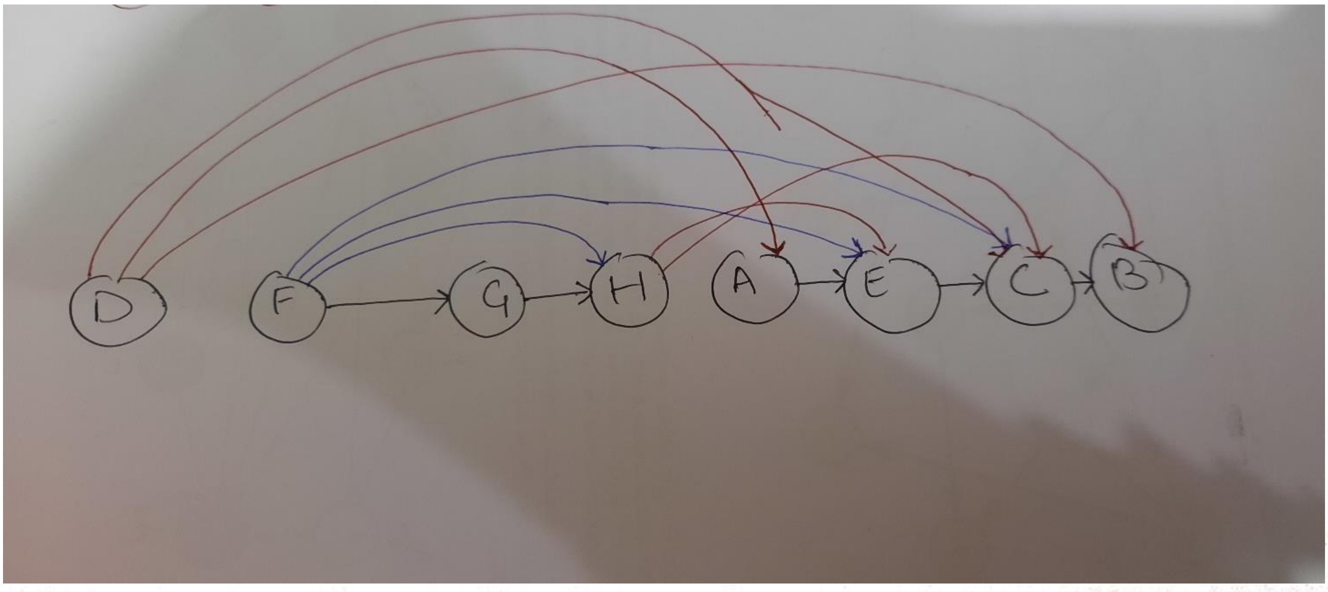


D F G H, A

chose C

(6)

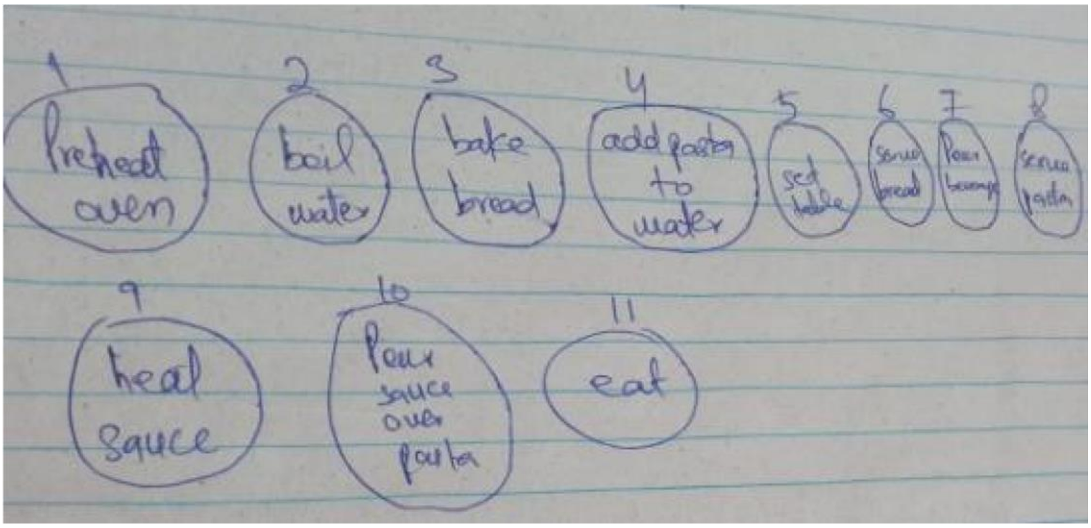
D F G H A.E



b)

•

•



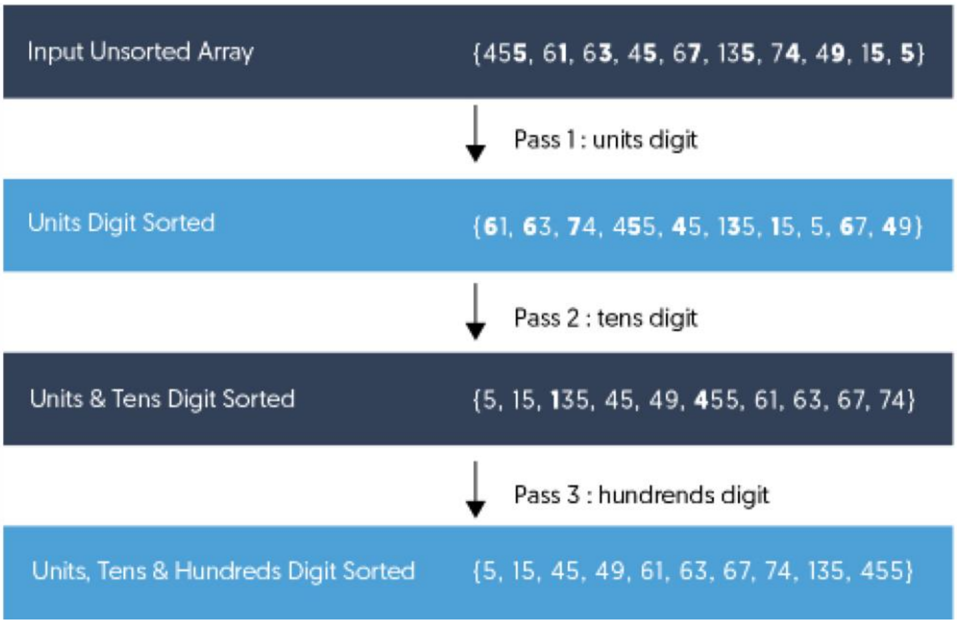
- **Boil water, add pasta, preheat oven, bake bread, set table, serve bread, heat sauce, serve pasta, pour sauce, pour bev, eat**
- **Set table, preheat oven, boil water, pour bev, heat sauce, bake bread, add pasta ,serve bread, serve pasta, pour sauce , eat**

<u>Question # 5</u> (15 min) [2.5 marks] [CLO 3]

You need to dry run the following algorithm to show all the necessary steps to sort an array A where array A= (455, 61, 63, 45, 67, 135, 74, 49, 15, 5) and d= 3. Show all steps clearly.

```
Sort(A, d)
        for j = 1 to d do
            int count[10] = {0};
            for i = 0 to n do
                count[key of(A[i]) in pass j]++
            for k = 1 to 10 do
                count[k] = count[k] + count[k-1]
            for i = n-1 downto 0 do
                result[ count[key of(A[i])] ] = A[j]
                count[key of(A[i])]--
            for i=0 to n do
                A[i] = result[i]
        end for(j)
 end func
```

<u>Solution</u>



ALL STEPS

PASS 1: UNITS DIGIT

0	1	2	3	4	5	6	7	8	9
44**5**	6**1**	6**3**	4**5**	6**7**	13**5**	7**4**	4**9**	1**5**	**5**

Count array:

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

Count array after storing the count of each unit digit in the original array:

0	1	2	3	4	5	6	7	8	9
0	1	0	1	1	5	0	1	0	1

Count array after storing the cumulative count so that their values indicate their sorted positions:

0	1	2	3	4	5	6	7	8	9
0	1	1	2	3	8	8	9	9	10

Array after pass 1:

0	1	2	3	4	5	6	7	8	9
61	63	74	455	45	135	15	5	67	49

PASS 2: TENS DIGIT

0	1	2	3	4	5	6	7	8	9
61	63	74	455	45	135	15	5	67	49

Count array:

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

Count array after storing the count of each tens digit in the resultant array of pass 1:

0	1	2	3	4	5	6	7	8	9
1	1	0	1	2	1	3	1	0	0

Count array after storing the cumulative count so that their values indicate their sorted positions:

0	1	2	3	4	5	6	7	8	9
1	2	2	3	5	6	9	10	10	10

After pass 2:

0	1	2	3	4	5	6	7	8	9
5	15	135	45	49	455	61	63	67	74

PASS 3: HUNDREDS DIGIT

0	1	2	3	4	5	6	7	8	9
5	15	135	45	49	455	61	63	67	74

Count array:

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

Count array after storing the count of each tens digit in the resultant array of pass 1:

0	1	2	3	4	5	6	7	8	9
8	1	0	0	1	0	0	0	0	0

Count array after storing the cumulative count so that their values indicate their sorted positions:

0	1	2	3	4	5	6	7	8	9
8	9	9	9	10	10	10	10	10	10

After pass 3:

0	1	2	3	4	5	6	7	8	9
5	15	45	49	61	63	67	74	135	455

<u>Question # 6</u> (30 min) **[1+1.5+3 = 5.5 marks] [CLO 4]**

Suppose you're running a lightweight consulting business — just you, two associates, and some rented equipment. Your clients are distributed between Karachi and Lahore, and this leads to the following question.

Each month, you can either run your business from an office in Karachi (KH) or from an office in Lahore (LR). In month $\boldsymbol{i}$, you'll incur an operating cost of $\mathbf{K_i}$ if you run the business out of KH; you'll incur an operating cost of $\mathbf{L_i}$ if you run the business out of LR. The costs depend on the distribution of client demands for that month. However, if you run the business out of one city in month $\boldsymbol{i}$, and then out of the other city in month $\boldsymbol{i+1}$, then you incur a fixed moving cost of $\boldsymbol{M}$ to switch base offices.

Given a sequence of $\boldsymbol{n}$ months, a plan is a sequence of $\boldsymbol{n}$ locations—each one equal to either KH or LR—such that the $\boldsymbol{i^{th}}$ location indicates the city in which you will be based in the $\boldsymbol{i^{th}}$ month. The cost of a plan is the sum of the operating costs for each of the **n** months, plus a moving cost of $\boldsymbol{M}$ for each time you switch cities. The plan can begin in either city.

The problem. Given a value for the moving cost M, and sequences of operating costs $K_1, \ldots, K_n$ and $L_1, \ldots, L_n$, find a plan of minimum cost. (Such a plan will be called optimal)

Example. Suppose n = 4, M = 10, and the operating costs are given by the following table.

	Month 1	Month 2	Month 3	Month 4
K_i	1	3	20	30
L_i	50	20	2	4

Then the plan of minimum cost would be the sequence of locations [KH, KH, LR, LR], with a total cost of 1 + 3 + 2 + 4 + 10 = 20, where the final term of 10 is the moving cost of changing locations once.

a) Show that the following algorithm does not correctly solve this problem, by giving an instance (example) on which it does not return the correct answer. Assume that n = 4 and M = 10 as it was in the example above. Give the optimal solution and its cost, as well as what the algorithm finds.

```
for i = 1 to n
        if Ki < Li then
                output "KH in Month i"
        else output "LR in Month i"
end
```

<u>Solution</u>

Here, NY = KH and SF = LR

Suppose that M = 10, {K1, K2, K3} = {1,4,1}, and {L1, L2, L3} = {20,1,20}. Then the optimal plan would be {KH, KH, KH} while this greedy method would return {KH, LR, KH}

b) Give an example of an instance (again with n = 4 and M = 10) in which the optimal plan must move (i.e., change locations) at least three times. Provide a brief explanation, saying why your example has this property.

<u>Solution</u>

Suppose that M = 10, $\{K_1,K_2,K_3,K_4\} = \{1,100,1,100\}$, and $\{L_1,L_2,L_3,L_4\} = \{100,1,100,1\}$

The plan {KH, LR, KH, LR} has cost 34 and it moves 3 times. Any other plan pays at least 100, and so is not optimal

c) Give a pseudo code for an efficient dynamic programming algorithm with complexity O(n) that takes values for n, M, and sequences of operating costs K1, . . . , Kn and L1, . . . , Ln, and returns the cost of an optimal plan.

Hint: What is the table of intermediate results with optimal substructure property?

Solution

SF = LR, and NY = KH

(c) The basic observation is: The optimal plan either ends in NY, or in SF. If it ends in NY, it will pay N_n plus one of the following two quantities:

- The cost of the optimal plan on $n-1$ months, ending in NY, or
- The cost of the optimal plan on $n-1$ months, ending in SF, plus a moving cost of M.

An analogous observation holds if the optimal plan ends in SF. Thus, if $OPT_N(j)$ denotes the minimum cost of a plan on months $1,\ldots,j$ ending in NY, and $OPT_S(j)$ denotes the minimum cost of a plan on months $1,\ldots,j$ ending in SF, then

$$OPT_N(n) = N_n + \min(OPT_N(n-1), M + OPT_S(n-1))$$

$$OPT_S(n) = S_n + \min(OPT_S(n-1), M + OPT_N(n-1))$$

This can be translated directly into an algorithm:

$OPT_N(0) = OPT_S(0) = 0$
For $i = 1,\ldots,n$
  $OPT_N(i) = N_i + \min(OPT_N(i-1), M + OPT_S(i-1))$
  $OPT_S(i) = S_i + \min(OPT_S(i-1), M + OPT_N(i-1))$
End
Return the smaller of $OPT_N(n)$ and $OPT_S(n)$

The algorithm has n iterations, and each takes constant time. Thus the running time is $O(n)$.

1	3 + min (1, 10+50) = 4	20 + min (4, 10+31) = 24	30 + min (24, 10+16) = 54
50	20 + min (50, 10+1) = 31	2 + min (31, 10+4) = 16	4 + min(16, 24+10) = 20