Analysis of Algorithm

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

❖ Longest Common Subsequence (LCS)

LCS Problem Statement: Given two sequences, find the length of longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous. For example, “abc”, “abg”, “bdf”, “aeg”, ‘”acefg”, .. etc are subsequences of “abcdefg”. So a string of length n has 2^n different possible subsequences.

It is a classic computer science problem, the basis of diff (a file comparison program that outputs the differences between two files), and has applications in bioinformatics.

Examples:

LCS for input Sequences “ABCDGH” and “AEDFHR” is “ADH” of length 3.

LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length 4.

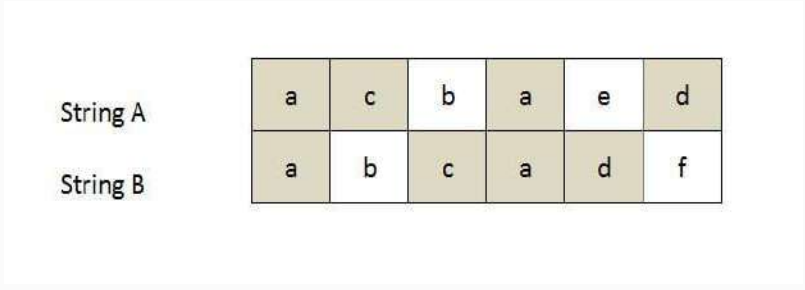
Objective:

Given two string sequences, write an algorithm to find the length of longest subsequence present in both of them.

These kind of dynamic programming questions are very famous in the interviews like Amazon, Microsoft, Oracle and many more. A longest subsequence is a sequence that appears in the same relative order, but not necessarily contiguous (not substring) in both the string.

Example:

String A = "acbaed"; String B = "abcadf";



Longest Common Subsequence (LCS): **acad**, Length: 4

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

Example

Given two sequences of symbols, X and Y, determine the longest subsequence of symbols that appears in both X and Y.

Example:

X = <A, B, C, B, D, A, B>

Y = <B, D, C, A, B, A>

LCS of X, Y?

<B, C, B, A>

<B, D, A, B>

1. Enumerate all subsequences of X.
2. Each subsequence of X corresponds to a subset of indices {1, 2, ….,m} of X.

There are 2^m subsequences of X, so this approach requires exponential time.

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0↑	0↑	0↑	↖1	1←	↖1
2	B	0	↖1	1←	1←	1←	↖2	2←
3	C	0	1↑	1↑	↖2	2←	2↑	2↑
4	B	0	↖1	1↑	2↑	2↑	↖3	3←
5	D	0	1↑	↖2	2↑	2↑	3↑	3↑
6	A	0	1↑	↖2	2↑	↖3	3↑	↖4
7	B	0	↖1	2↑	2↑	3↑	↖4	4↑

Algorithm to compute the above table is given below

Analysis of Algorithm

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

- **LCS – Algorithm: Constructing an LCS**
- Table b[1..m, 1..n] can be used to construct an LCS.
- Initial invocation is **Print_LCS(b, X, length[X], length[Y]).**

```
Print_LCS(b[], X[], i, j)
{   if(i = = 0 or j = = 0)
        return 0;
    if(b[i, j] = = "↖"                    // common elements
    {   Print_LCS(b, X, I-1, j-1);
        print xi;
    }
    else if (b[I, j] = = "↑")
        Print_LCS(b, X, I-1, j);
    else Print_LCS(b, X, I, j-1);
}
```

- Devise a memoized version of LCS_Length();

I. **Characterize the LCS problem**

- The LCS problem has an optimal substructure property.
- **Can we formulate optimal substructure?**
- ith preifx of a sequence X is X_i, e.g.,
- X_4 of sequence X = <A, B, C, B, D, A, B> is <A, B, C, B>

Optimal Substructure of an LCS:

Let $X = \langle x_1, x_2, x_3, \ldots, x_m \rangle$ and

$Y = \langle y_1, y_2, y_3, \ldots, y_n \rangle$ and let

$Z = \langle z_1, z_2, z_3, \ldots z_k \rangle$ be any LCS of X and Y

1. If $x_m = y_n$ then $z_k = x_m = y_n$ and z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m \neq y_n$ then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.
3. If $x_m \neq y_n$ then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}

Step 1: Characterizing an LCS

- LCS problem has an optimal substructure property. I.e.
- There are either one or two sub-problems to examine when finding an LCS of $X = \langle x_1, x_2, \ldots, x_m \rangle$, $Y = \langle y_1, y_2, \ldots, y_n \rangle$
- if $x_m = y_n$
  - find an LCS of X_{m-1} and Y_{n-1} appending x_m and y_n ($x_m = y_n$) to this LCS yields an LCS of X and Y.
- If $x_m \neq y_n$
  - solve two sub-problems:
  - Finding an LCS of Xm-1 and Y
  - Finding an LCS of X and Yn-1
  - Longer of I and II is an LCS of X and Y.
- This shows overlapping sub-problmes property in LCS.

Step 2:

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

Define a recursive definition

- Establish a recurrence for the cost of an optimal solution
- Let LCS[i,j] to be the length of the longest common subsequences (LCS) of X_i an Y_j.

Recursive Definition for LCS

$$LCS(i,j) = \begin{cases} 0 & \text{if } (i = 0 \text{ or } j = 0) \\ LCS(i-1, j-1) + 1 & \text{if } (i, j > 0 \text{ and } X_i = Y_j) \\ MAX(LCS(i, j-1), LCS(i-1, j)) & \text{if } (i, j > 0 \text{ and } X_i \neq Y_j) \end{cases}$$

$X = \langle X_1, X_2, X_3, \ldots, X_m \rangle$

$Y = \langle Y_1, Y_2, Y_3, \ldots, Y_n \rangle$

Where i varies from 0 to m and j varies from 0 to n.

Step 3:

Devise an algorithm to compute the length of an LCS and then construct the LCS.

- Development of an algorithm
- Input two sequences X and Y and a two dimensional table LCS[0..m , 0..n]
- According to the definition

Memorized Algorithm

```
for(i =1;   i<=m; i++)
{
   for(j =1;   j<=n; j++)
    {
          if (X[i] = =Y[j])
                LCS[ i , j ] = LCS[ i-1 , j-1 ] + 1;
          else if (LCS[ i-1,j] >= LCS[ i , j-1 ])          /* It means to compute LCS[i,j]
                LCS[ i , j ] = LCS[ i-1 , j ];                We need to know
```

```
                else                                                    LCS[ i-1 , j-1 ]
    , LCS[ i, j-1 ],
                        LCS[ i , j ] = LCS[ i , j-1 ];          and LCS[ i-1 , j ]
        }                                                               */
    }
```

❖ Assembly Line Scheduling (ALS)

A car factory has two assembly lines, each with n stations. A station is denoted by $S_{i,j}$ where i is either 1 or 2 and indicates the assembly line the station is on, and j indicates the number of the station.

The time taken per station is denoted by $a_{i,j}$. Each station is dedicated to some sort of work like engine fitting, body fitting, painting and so on. So, a car chassis must pass through each of the n stations in order before exiting the factory.

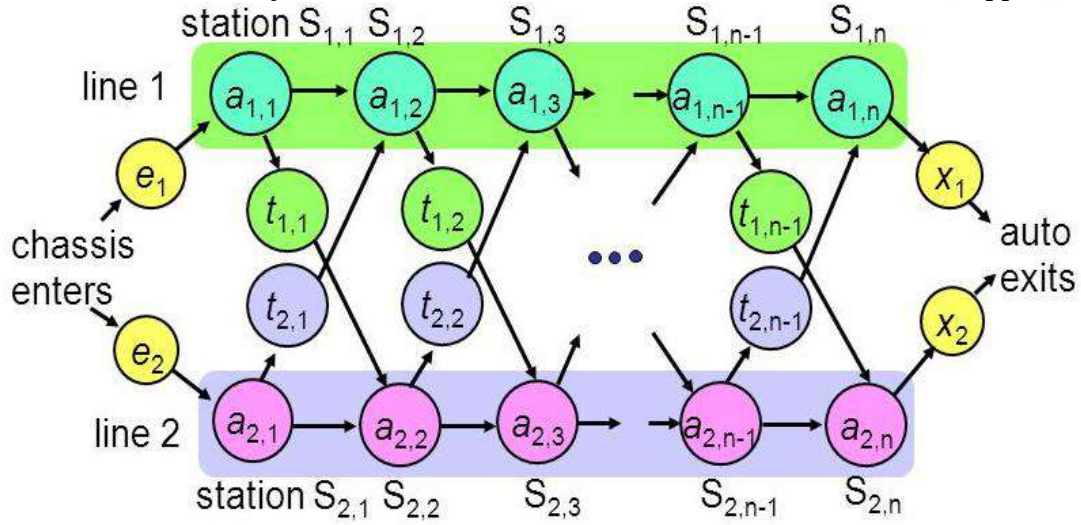
The parallel stations of the two assembly lines perform the same task. After it passes through station $S_{i,j}$, it will continue to station $S_{i,j+1}$ unless it decides to transfer to the other line. Continuing on the same line incurs no extra cost, but transferring from line i at station j – 1 to station j on the other line takes time $t_{i,j}$.

Each assembly line takes an entry time e_i and exit time x_i which may be different for the two lines. Give an algorithm for computing the minimum time it will take to build a car chassis.

- **Automobile factory with two assembly lines**
  - **Each line has n stations: $S_{1,1}, \ldots, S_{1,n}$ and $S_{2,1}, \ldots, S_{2,n}$**
  - **Corresponding stations $S_{1,j}$ and $S_{2,j}$ perform the same function but can take different amounts of time $a_{1,j}$ and $a_{2,j}$**
  - **Entry times are: e_1 and e_2; exit times are: x_1 and x_2**

Analysis of Algorithm

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010



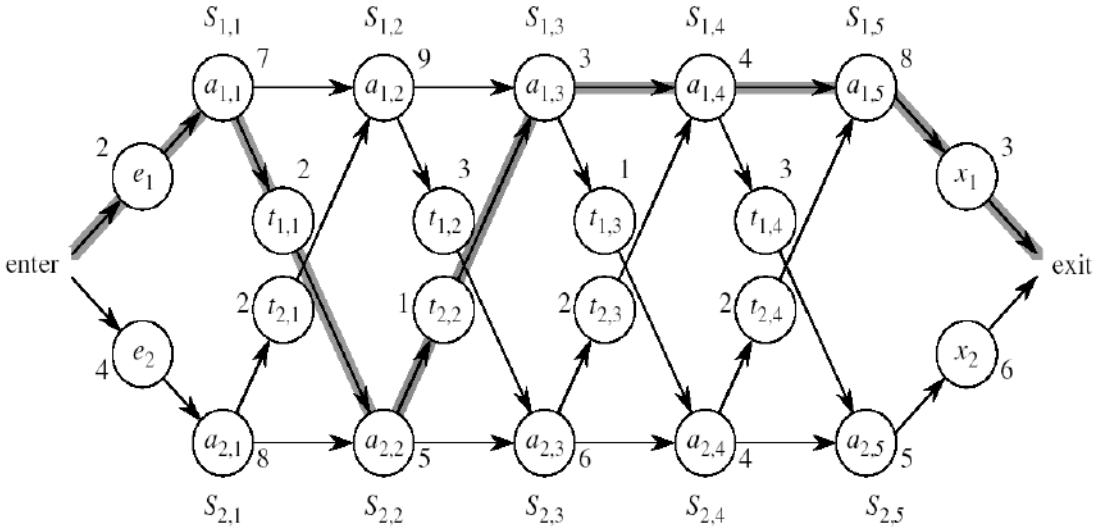
The following information can be extracted from the problem statement to make it simpler:

- Two assembly lines, 1 and 2, each with stations from 1 to n.
- A car chassis must pass through all stations from 1 to n in order(in any of the two assembly lines). i.e. it cannot jump from station i to station j if they are not at one move distance.
- The car chassis can move one station forward in the same line, or one station diagonally in the other line. It incurs an extra cost $t_{i,j}$ to move to station j from line i. No cost is incurred for movement in same line.
  - **transfer to other line: cost after $S_{i,j}$ is $t_{i,j}$, $j = 1, \ldots, n - 1$**
- The time taken in station j on line i is $a_{i,j}$.
- $S_{i,j}$ represents a station j on line i.

Problem:
What stations should be chosen from line 1 and which from line 2 in order to minimize the total time through the factory for one car?

Analysis of Algorithm

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

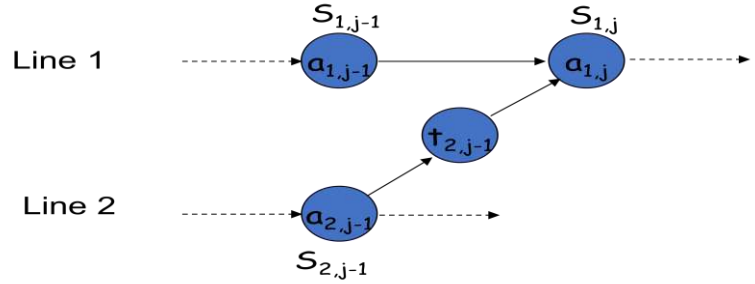


Brute Force

- Enumerate all possibilities of selecting stations
- Compute how long it takes in each case and choose the best one
- There are 2^n possible ways to choose stations
- Infeasible when n is large!!

Optimal Solution

- How do we compute the minimum time of going through a station?
- Let's consider all possible ways to get from the starting point through station $S_{1,j}$
  - We have two choices of how to get to $S_{1,j}$:
    - Through $S_{1,j-1}$, then directly to $S_{1,j}$
    - Through $S_{2,j-1}$, then transfer over to $S_{1,j}$



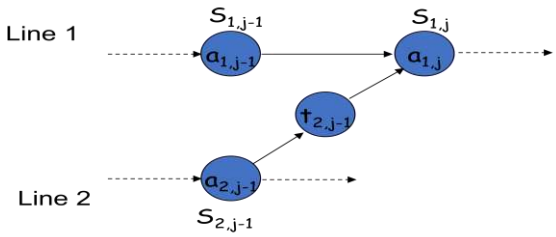
Structure of the optimal solution

Suppose that the fastest way through $S_{1,j}$ is through $S_{1,j-1}$

We must have taken a fastest way from entry through $S_{1,j-1}$

If there were a faster way through $S_{1,j-1}$, we would use it instead

Similarly for $S_{2,j-1}$



- **Generalization**: an optimal solution to the problem "find the fastest way through $S_{1,j}$" contains within it an optimal solution to subproblems: "find the fastest way through $S_{1,j-1}$ or $S_{2,j-1}$".
- We use this property to construct an optimal solution to a problem from optimal solutions to sub-problems

Recursive Solution

Define the value of an optimal solution in terms of the optimal solution to sub-problems

Definitions:

- f^* : the fastest time to get through the entire factory
- $f_i[j]$: the fastest time to get from the starting point through station $S_{i,j}$

$$f^* = \min (f_1[n] + x_1, f_2[n] + x_2)$$

Base case: j = 1, i=1, 2 (getting through station 1)

$$f_1[1] = e_1 + a_{1,1}$$

$$f_2[1] = e_2 + a_{2,1}$$

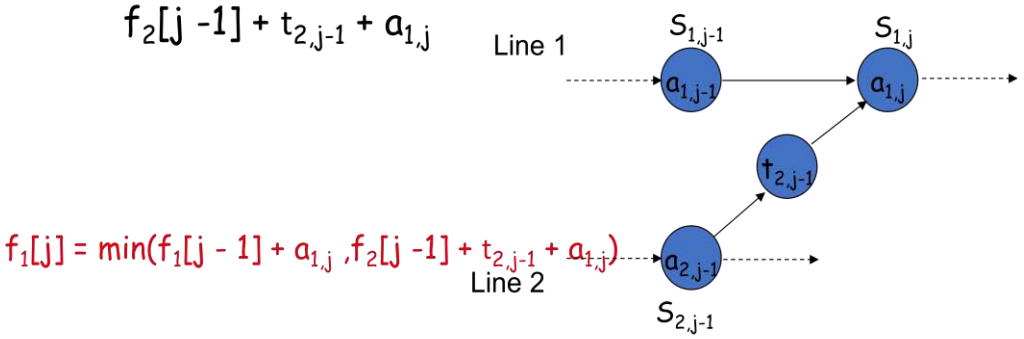
Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

- General Case: j = 2, 3, …,n, and i = 1, 2
- Fastest way through $S_{1,j}$ is either:
  - the way through $S_{1,j-1}$ then directly through $S_{1,j}$, or

$$f_1[j-1] + a_{1,j}$$

  - the way through $S_{2,j-1}$, transfer from line 2 to line 1, then through $S_{1,j}$

$$f_2[j-1] + t_{2,j-1} + a_{1,j}$$



Recursive Definition

$$f[i,j] = \begin{cases} e_i + a_{i,j} & \text{if } (i=1 \text{ or } 2 \ \&\ j = 1) \\ \text{MIN}(f[i,j-1] + a_{i,j},\ f[i+1,j-1] + t_{i+1,j-1} + a_{i,j}) & \text{if } (i=1 \ \&\ j \geq 2) \\ \text{MIN}(f[i,j-1] + a_{i,j},\ f[i-1,j-1] + t_{i-1,j-1} + a_{i,j}) & \text{if } (i=2 \ \&\ j \geq 2) \end{cases}$$

$$\text{result} = \min(f[1,n] + x_1,\ f[2,n] + x_2)$$

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

$$f^* = \min(f_1[n] + x_1, f_2[n] + x_2)$$

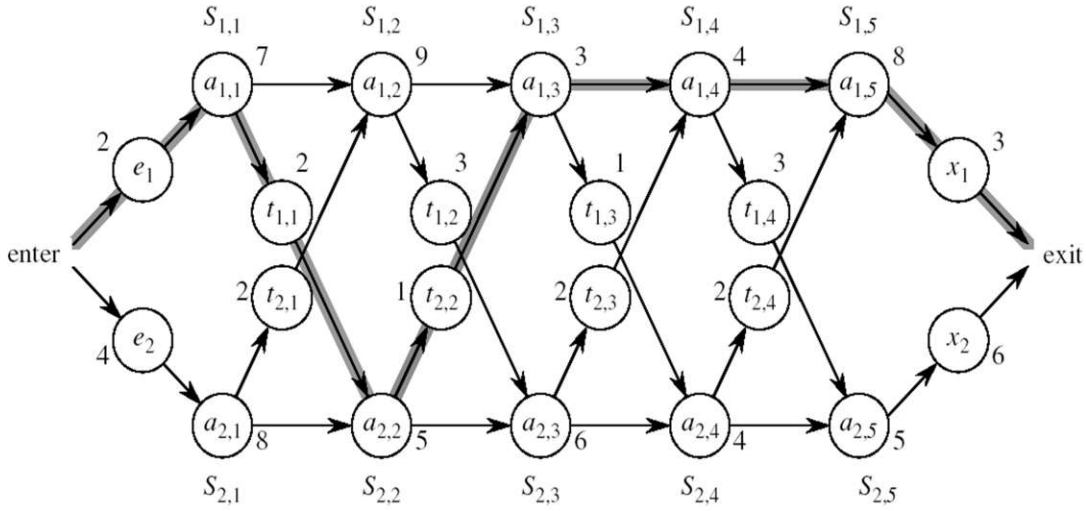
$$f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$$

$$f_2[j] = \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{2,j})$$

	1	2	3	4	5
$f_1[j]$	$f_1(1)$	$f_1(2)$	$f_1(3)$	$f_1(4)$	$f_1(5)$
$f_2[j]$	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$	$f_2(5)$
			4 times	2 times	

- Solving top-down would result in exponential running time

Example



$$f_1[j] = \begin{cases} e_1 + a_{1,1}, & \text{if } j = 1 \\ \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j}) & \text{if } j \geq 2 \end{cases}$$

	1	2	3	4	5
$f_1[j]$	9	$18^{[1]}$	$20^{[2]}$	$24^{[1]}$	$32^{[1]}$
$f_2[j]$	12	$16^{[1]}$	$22^{[2]}$	$25^{[1]}$	$30^{[2]}$

$f^* = 35^{[1]}$

Recursive Algorithm

Analysis of Algorithm

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

ALS(i , j) {

if(i=1 or 2 and j=1)

return $e_{ij}+a_{ij}$

else if (i=1 and j>1)

return MIN(ALS(i, j-1) + $a_{i,j}$, ALS(i+1, j-1) + $t_{i+1,j-1}$ + $a_{i,j}$)

else if (i=2 and j>1)

return MIN(ALS(i, j-1) + $a_{i,j}$, ALS(i-1, j-1) + $t_{i-1,j-1}$ + $a_{i,j}$)

}

FASTEST-WAY(a, t, e, x, n)

1. $f_1[1] \leftarrow e_1 + a_{1,1}$
2. $f_2[1] \leftarrow e_2 + a_{2,1}$

} Compute initial values of f_1 and f_2

3. **for** $j \leftarrow 2$ **to** n — **O(N)**
4. **do if** $f_1[j-1] + a_{1,j} \leq f_2[j-1] + t_{2,j-1} + a_{1,j}$
5. **then** $f_1[j] \leftarrow f_1[j-1] + a_{1,j}$
6. $l_1[j] \leftarrow 1$
7. **else** $f_1[j] \leftarrow f_2[j-1] + t_{2,j-1} + a_{1,j}$

} Compute the values of $f_1[j]$ and $l_1[j]$

8. $l_1[j] \leftarrow 2$
9. **if** $f_2[j-1] + a_{2,j} \leq f_1[j-1] + t_{1,j-1} + a_{2,j}$
10. **then** $f_2[j] \leftarrow f_2[j-1] + a_{2,j}$
11. $l_2[j] \leftarrow 2$
12. **else** $f_2[j] \leftarrow f_1[j-1] + t_{1,j-1} + a_{2,j}$
13. $l_2[j] \leftarrow 1$

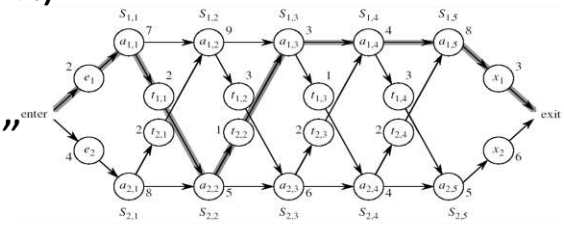
} Compute the values of $f_2[j]$ and $l_2[j]$

```
14. if f1[n] + x1 ≤ f2[n] + x2
15.    then f* = f1[n] + x1
16.         l* = 1
17.    else f* = f2[n] + x2
18.         l* = 2
```

Compute the values of the fastest time through the entire factory

Alg.: PRINT-STATIONS(l, n)

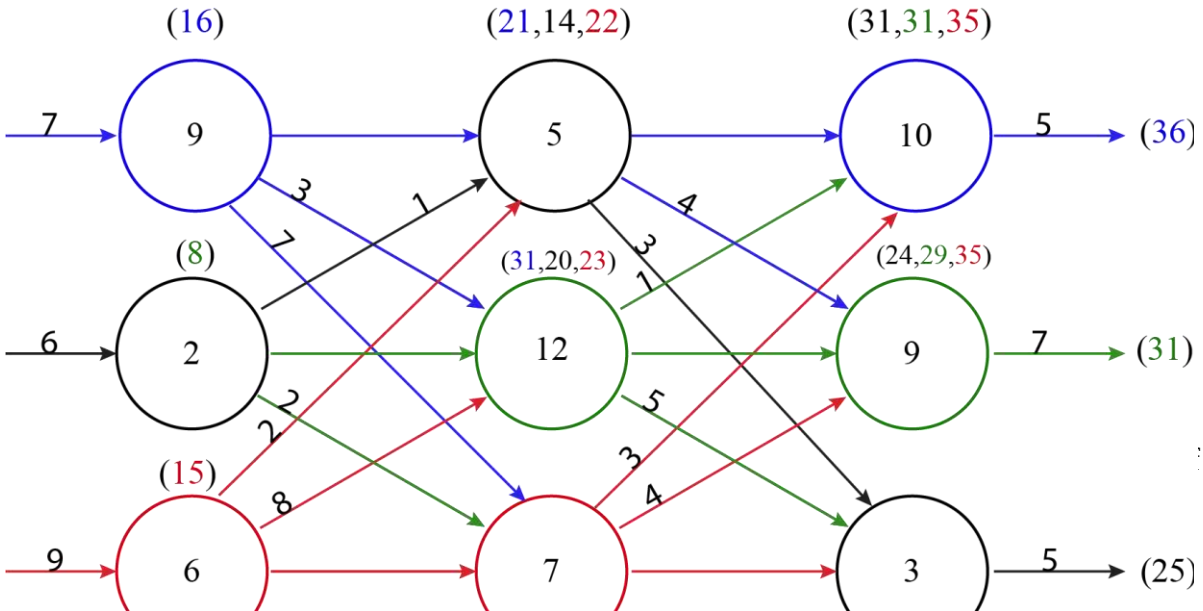
```
i ← l*
print "line " i ", station "
for j ← n downto 2
    do i ← li[j]
       print "line " i ", station " j
```



	1	2	3	4	5
$f_1[j]/l_1[j]$	9	18[1]	20[2]	24[1]	32[1]
$f_2[j]/l_2[j]$	12	16[1]	22[2]	25[1]	30[2]

l* = 1

Example:



Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk WhatsApp# 0346-5100010

Sol:

Line	S_1	S_2	S_3	Cost
L_1	16[1]	14[2]	31[1]	36[1]
L_2	8[2]	20[2]	24[1]	31[2]
L_2	15[3]	17[2]	20[1]	25[3]