21BDS0340

Abhinav Dinesh Srivatsa

Design and Analysis of Algorithms Lab

Digital Assignment 2

<u>Question 1</u>

<u>Algorithm</u>

Given jobs with respective deadlines, penalties, and completion times

1. Input selected jobs as Jobs and U
2. If sum Jobs' completion time > maximum deadline, return (branch is killed)
3. Calculate Ĉ as the sum of penalties of jobs less than maximum Jobs' index
4. If Ĉ > U, return (branch is killed)
5. Calculate U as the sum of the penalties of the left over jobs
6. Add a job to Jobs and repeat from step 1

<u>Problem Solving</u>

Id	Penalty	Deadline	Time to Complete
1	5	1	1
2	10	3	2
3	6	2	1
4	3	1	1

Computation Table

Node	Selected Jobs	Validity	ĉ	u
1	{}	Yes	0	5 + 10 + 6 + 3 = **24**
2	{ 1 }	Yes	0	10 + 6 + 3 = **19**
3	{ 2 }	Yes	5	5 + 6 + 3 = **14**
4	{ 3 }	Yes	5 + 10 = 15 (> min(u), discarded)	
5	{ 4 }	Yes	5 + 10 + 6 = 21 (> min(u), discarded)	
6	{ 1, 2 }	Yes	0	6 + 3 = **9**
7	{ 1, 3 }	Yes	10 (> min(u), discarded)	
8	{ 1, 4 }	Yes	10 + 6 = 16 (> min(u), discarded)	
9	{ 2, 3 }	Yes	5	5 + 3 = **8**
10	{ 2, 4 }	Yes	5 + 6 = 11	

			(> min(u), discarded)	
11	{ 3, 4 }	No		

Minimum cost at **node 9,** therefore the solution is to complete jobs **{2, 3}**

Question 2

Algorithm

Given a string S and Pattern

1. Create a prefix table for the string Pattern as Pi
2. Keep track of letters in S as I
3. Set J as 0
4. If I equals Pattern[J + 1], then increment J
5. If the letters do not match, then set J as Pattern[J]'s Pi and set I as the next letter in S
6. If J is equal to the length of the Pattern, sequence is found, set J to Pattern[J]'s Pi

Problem Solving

Pi Table:

Letter	a	b	c	d	f
Index	1	2	3	4	5
Pi	0	0	0	0	0

Iterations:

1. Pattern[J + 1] = a

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I	X											

Match! Increment I, J

2. Pattern[J + 1] = b

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I		X										

Match! Increment I, J

3. Pattern[J + 1] = c

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I			X									

Match! Increment I, J

4. Pattern[J + 1] = d

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I				X								

Match! Increment I, J

5. Pattern[J + 1] = f

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I					X							

Not match, J = 0

6. Pattern[J + 1] = a

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I					X							

Match! Increment I, J

7. Pattern[J + 1] = b

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I						X						

Match! Increment I, J

8. Pattern[J + 1] = c

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I							X					

Match! Increment I, J

9. Pattern[J + 1] = d

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I								X				

Not match, J = 0

10. Pattern[J + 1] = a

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I								X				

Match! Increment I, J

11. Pattern[J + 1] = b

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I									X			

Match! Increment I, J

12. Pattern[J + 1] = c

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I										X		

Match! Increment I, J

13. Pattern[J + 1] = d

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I											X	

Match! Increment I, J

14. Pattern[J + 1] = f

Letter	a	b	c	d	a	b	c	a	b	c	d	f
I												X

Match! Increment I, J

J = length of Pattern, **sequence found**!

Time Complexity

The time complexity if the algorithm depends on the length of the string S and Pattern with lengths m and n. The order is of **O(m + n)**

Question 3

Algorithm

Assuming course 0 has already been done
Given that the Prerequisites array consists of [ai, bi]

1. Create a list Courses to store the courses that can be done
2. Loop from 0 to length of Prerequisites
3. Loop through the Prerequisites array's elements
4. If bi exists in Courses, then add ai to Courses

5. If all ai exists in Courses, then return true
6. Otherwise return false

Problem Solving

Course	1	2	3	3
Prerequisite	0	0	1	2

1. Current course = [1, 0]

Courses = [0]

Prerequisite 0 satisfied, add 1

2. Current course = [2, 0]

Courses = [0, 1]

Prerequisite 0 satisfied, add 2

3. Current course = [3, 1]

Courses = [0, 1, 2]

Prerequisite 1 satisfied, add 3

4. Current course = [3, 2]

Courses = [0, 1, 2, 3]

Prerequisite 1 satisfied, add 3

Since all courses have been added to the Courses array, return **true**

Time Complexity

The time complexity depends on the number of courses and prerequisites, n and e respectively, the time complexity is of order **O(n + e)**

Question 4

Algorithm

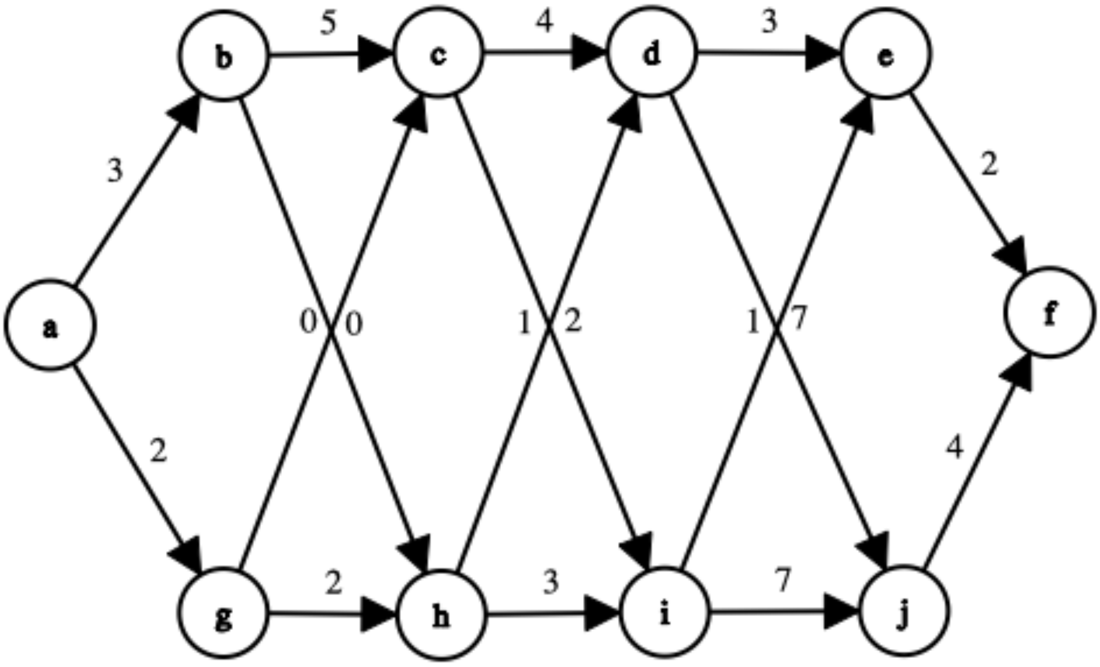
Given the line times, transfer times, entry and exit times

1. Create a dynamic programming table with two rows and the number of units as columns
2. Traverse the following steps from the left columns to right

3. For the first column, add the entry time to the line time
4. For every column after, find the minimum of staying on the line vs transferring from the other line
5. Complete the table to find the minimum time at the last column and add the exit times
6. The least time is the minimum time taken for a car to finish assembling

Problem Solving

Converting the given problem into a graph:



Solution:

	b, g	c, h	d, i	e, j	f
Line 1	3	Min(2, 8) = 2	Min(4, 6) = 4	Min(5, 7) = 5	5 + 3 = 8
Line 2	2	Min(3, 4) = 3	Min(4, 6) = 4	Min(6, 11) = 6	6 + 4 = 10

From this, the least time for a car to be assembled is **10 units**

Time Complexity

The time complexity of this algorithm depends on the length of the lines 1 and 2, which are equal as n, the complexity is of order **O(n)**