GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

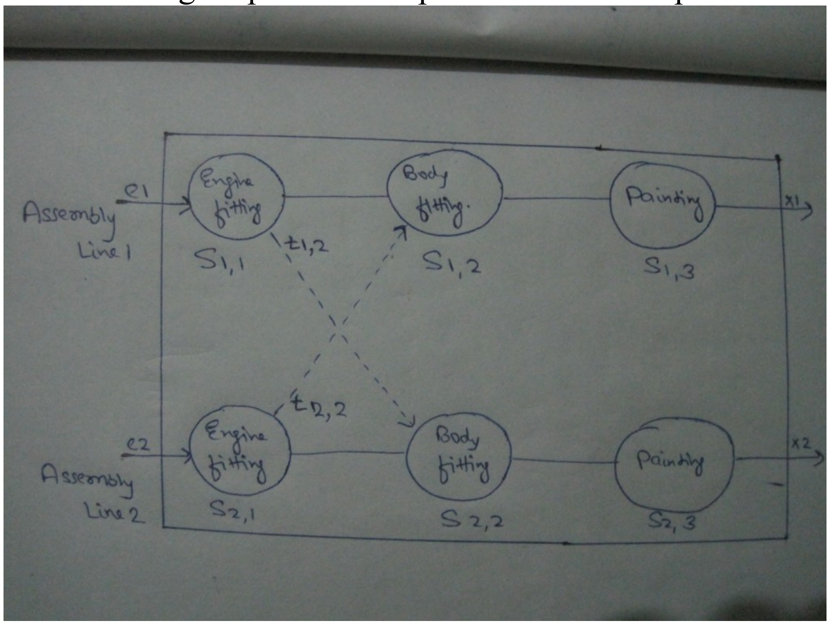
- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Dynamic Programming | Set 34 (Assembly Line Scheduling)

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.

The below figure presents the problem in a clear picture:



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 ti, j to move to station j from line i. No cost is incurred for movement in same line.
- The time taken in station j on line i is $a_{i, j}$.
- $S_{i, j}$ represents a station j on line i.

Breaking the problem into smaller sub-problems:
We can easily find the ith factorial if (i-1)th factorial is known. Can we apply the similar funda here?
If the minimum time taken by the chassis to leave station $S_{i, j-1}$ is known, the minimum time taken to leave station $S_{i, j}$ can be calculated quickly by combining $a_{i, j}$ and $t_{i, j}$.

T1(j) indicates the minimum time taken by the car chassis to leave station j on assembly line 1.

T2(j) indicates the minimum time taken by the car chassis to leave station j on assembly line 2.

Base cases:
The entry time e_i comes into picture only when the car chassis enters the car factory.

Time taken to leave first station in line 1 is given by:
T1(1) = Entry time in Line 1 + Time spent in station $S_{1,1}$
$T1(1) = e_1 + a_{1,1}$
Similarly, time taken to leave first station in line 2 is given by:
$T2(1) = e_2 + a_{2,1}$

Recursive Relations:
If we look at the problem statement, it quickly boils down to the below observations:

The car chassis at station $S_{1,j}$ can come either from station $S_{1, j-1}$ or station $S_{2, j-1}$.

Case #1: Its previous station is $S_{1, j-1}$
The minimum time to leave station $S_{1,j}$ is given by:
T1(j) = Minimum time taken to leave station $S_{1, j-1}$ + Time spent in station $S_{1, j}$
T1(j) = T1(j-1) + $a_{1, j}$

Case #2: Its previous station is $S_{2, j-1}$
The minimum time to leave station S1, j is given by:
T1(j) = Minimum time taken to leave station $S_{2, j-1}$ + Extra cost incurred to change the assembly line + Time spent in station $S_{1, j}$
T1(j) = T2(j-1) + $t_{2, j}$ + $a_{1, j}$

The minimum time T1(j) is given by the minimum of the two obtained in cases #1 and #2.
T1(j) = min((T1(j-1) + $a_{1, j}$), (T2(j-1) + $t_{2, j}$ + $a_{1, j}$))
Similarly the minimum time to reach station S2, j is given by:
T2(j) = min((T2(j-1) + $a_{2, j}$), (T1(j-1) + $t_{1, j}$ + $a_{2, j}$))

The total minimum time taken by the car chassis to come out of the factory is given by:
Tmin = min(Time taken to leave station $S_{i,n}$ + Time taken to exit the car factory)
Tmin = min(T1(n) + x_1, T2(n) + x_2)

Why dynamic programming?
The above recursion exhibits overlapping sub-problems. There are two ways to reach station $S_{1, j}$:

1. From station $S_{1, j-1}$
2. From station $S_{2, j-1}$

So, to find the minimum time to leave station $S_{1, j}$ the minimum time to leave the previous two stations must be calculated(as explained in above recursion).
Similarly, there are two ways to reach station $S_{2, j}$:

1. From station $S_{2, j-1}$
2. From station $S_{1, j-1}$

Please note that the minimum times to leave stations $S_{1, j-1}$ and $S_{2, j-1}$ have already been calculated.

So, we need two tables to store the partial results calculated for each station in an assembly line. The table will be filled in bottom-up fashion.

Note:
In this post, the word “leave” has been used in place of “reach” to avoid the confusion. Since the car chassis must spend a fixed time in each station, the word leave suits better.

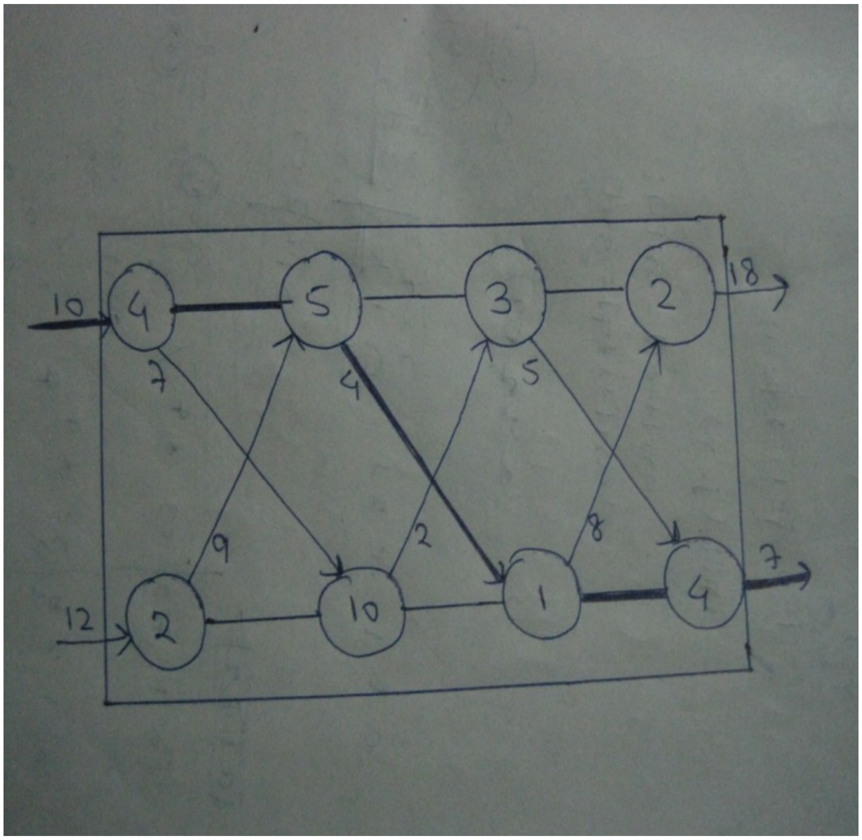
Implementation:

```
// A C program to find minimum possible time by the car chassis to complete
```

```
#include <stdio.h>
#define NUM_LINE 2
#define NUM_STATION 4

// Utility function to find minimum of two numbers
int min(int a, int b) { return a < b ? a : b; }

int carAssembly(int a[][NUM_STATION], int t[][NUM_STATION], int *e, int *x)
{
    int T1[NUM_STATION], T2[NUM_STATION], i;

    T1[0] = e[0] + a[0][0]; // time taken to leave first station in line 1
    T2[0] = e[1] + a[1][0]; // time taken to leave first station in line 2

    // Fill tables T1[] and T2[] using the above given recursive relations
    for (i = 1; i < NUM_STATION; ++i)
    {
        T1[i] = min(T1[i-1] + a[0][i], T2[i-1] + t[1][i] + a[0][i]);
        T2[i] = min(T2[i-1] + a[1][i], T1[i-1] + t[0][i] + a[1][i]);
    }

    // Consider exit times and retutn minimum
    return min(T1[NUM_STATION-1] + x[0], T2[NUM_STATION-1] + x[1]);
}

int main()
{
    int a[][NUM_STATION] = {{4, 5, 3, 2},
                {2, 10, 1, 4}};
    int t[][NUM_STATION] = {{0, 7, 4, 5},
                {0, 9, 2, 8}};
    int e[] = {10, 12}, x[] = {18, 7};

    printf("%d", carAssembly(a, t, e, x));

    return 0;
}
```

Output:

```
35
```



The bold line shows the path covered by the car chassis for given input values.

Exercise:
Extend the above algorithm to print the path covered by the car chassis in the factory.

References:
[Introduction to Algorithms 3rd Edition by Clifford Stein, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest]

This article is compiled by **Aashish Barnwal**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

Writing code in comment? Please use **ideone.com** and share the link here.

19 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

prashant jha • a year ago

the complexity of naice recursive algo will be 0(2^n)
for dp it wil be 0(n) as there cant be more than 2*n numbers of subproblems unless no of parallel assemblies be >2
here is my dp implementation
http://ideone.com/JTIX3J

• Reply • Share ›

prashant jha • a year ago

here is my dyanamic programming solution

```
#include<iostream>
#include<string.h>
using namespace std;
int min(int a,int b)
{
return a>b?b:a;
}
int min(int a,int b,int c)
{
return min(min(a,b),c);
}
int fun(int i,int j,int a[][4],int t[][4],int x[],int arr[][4],int high)
{
if(arr[i][j]!=-1)
return arr[i][j];
if(j==high)
```

see more

• Reply • Share ›

anonymous • a year ago

The naive n recursive solution of the above problem is O(2^n) ... however, the dp solution suggested above is linear time . Am I wrong?
asking because i mocked by an interviewer at a reputed company upon saying so,
and had a hard time convincing him of my point.

3 • Reply • Share ›

3 ^ | v • Reply • Share ›

prashant jha ↪ anonymous • a year ago

u r ryt

^ | v • Reply • Share ›

reshma ↪ anonymous • a year ago

it takes more than that i think.....bcz it is like two arrays now... we need to check to arrays combination... O(2^n) when only one arry with n elements... of combinations..... but now what is time complexity i too don't exactly...tell us if anybody knowss Thanks..

1 ^ | v • Reply • Share ›

Scott Ellis • a year ago

You can take the '+a[x][i]' outside the 'min' call.

1 ^ | v • Reply • Share ›

xiaoguangye • a year ago

I think this is not a good example of DP. It is more abt single source shortest path in a DAG that you can easily achieve within time O(E+v) by textbook algo.

2 ^ | v • Reply • Share ›

the_c0der ↪ xiaoguangye • a year ago

exactly what i thought , but time complexity will NOT be O(E+V) as we would have to use dijkstra's algo and use HEAP .

1 ^ | v • Reply • Share ›

meh • a year ago

Shouldn't this section:

T1[i] = min(T1[i-1] + a[0][i], T2[i-1] + t[1][i] + a[0][i]);
T2[i] = min(T2[i-1] + a[1][i], T1[i-1] + t[0][i] + a[1][i]);

be instead like this:

T1[i] = min(T1[i-1] + a[0][i], T2[i-1] + t[1][i - 1] + a[0][i]);
T2[i] = min(T2[i-1] + a[1][i], T1[i-1] + t[0][i - 1] + a[1][i]);

In other words, the transition time considered should be the one from the previous station and not the same one.

^ | v • Reply • Share ›

meh ↪ meh • a year ago

Ah, never mind, you're considering t[i][j] to be the transition time of station j - 1 from line i :P

^ | v • Reply • Share ›

Viki • 2 years ago
Use of just four variables T1, T2, T1_pre, T2_pre will suffice our purpose.

Why to use array ?
1 • Reply • Share ›

parbays • 2 years ago
Isn't it Bellman ford algorithm applied on the graph formed by stations on the assembly line?
3 • Reply • Share ›

gg • 2 years ago
why dont we just keep track of lowest cost at every position and proceed rather than computing lowest cost from line 1 and line 2 and then taking minimum at last?

```
/* Paste your code here (You may delete these lines if not writing code) */
```

1 • Reply • Share ›

Aashish → gg • 2 years ago
Because lowest cost at any position can be either through choosing previous station from line 1 or line 2. We recommend you to go through the explanation to make things clear.
• Reply • Share ›

Bala Sravan • 2 years ago
Solution to exercise :

```
#include<stdio.h>
int train(int* a,int* t,int* x,int source,int n,int *flag,int* path,int count);
void printpath(int source,int* path,int n,int line);
void main()
{
int n,i,j=0;//number of stations
int a[2][4],t[2][4];
int e[2],x[2];
n=4;
e[0]=10;
e[1]=12;
x[0]=18;
x[1]=7;
a[0][0]=4;
a[0][1]=5;
a[0][2]=3;
```

see more

• Reply • Share ›

Lakshminarayana K • 2 years ago

Good Explanation

1 • Reply • Share ›

Venki • 2 years ago

Nice work Aashish. Following my style to create effective pictures? Good algo, I haven't tried earlier.

Does anyone of our readers knew about graphology? Sorry, it is off the topic.

• Reply • Share ›

Aashish → Venki • 2 years ago

Thanks Venki. I liked your style as it takes less efforts(pen and paper) and readers can understand it in the same way we do. The problem is a bit different from other DP problems. So, you should try it once. An interesting problem, you will enjoy it for sure.

• Reply • Share ›

abhishek → Aashish • 2 years ago

This is very nice prob. I happened to read that in the ref book one year ago. Aashish you explained here very well.

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 2 hours ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours ago

- lucy

because we first fill zero in first col and...

Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago

-