GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

Weighted Job Scheduling

Given N jobs where every job is represented by following three elements of it.
1) Start Time
2) Finish Time.
3) Profit or Value Associated.
Find the maximum profit subset of jobs such that no two jobs in the subset overlap.

Example:

```
Input: Number of Jobs n = 4
       Job Details {Start Time, Finish Time, Profit}
       Job 1:  {1, 2, 50}
       Job 2:  {3, 5, 20}
       Job 3:  {6, 19, 100}
```

```
       Job 4:  {2, 100, 200}
Output: The maximum profit is 250.
We can get the maximum profit by scheduling jobs 1 and 4.
Note that there is longer schedules possible Jobs 1, 2 and 3
but the profit with this schedule is 20+50+100 which is less than 250.
```

A simple version of this problem is discussed here where every job has same profit or value. The Greedy Strategy for activity selection doesn't work here as the longer schedule may have smaller profit or value.

The above problem can be solved using following recursive solution.

```
1) First sort jobs according to finish time.
2) Now apply following recursive process.
   // Here arr[] is array of n jobs
   findMaximumProfit(arr[], n)
   {
     a) if (n == 1) return arr[0];
     b) Return the maximum of following two profits.
         (i) Maximum profit by excluding current job, i.e.,
             findMaximumProfit(arr, n-1)
         (ii) Maximum profit by including the current job
   }
```

How to find the profit excluding current job?

```
The idea is to find the latest job before the current job (in
sorted array) that doesn't conflict with current job 'arr[n-1]'.
Once we find such a job, we recur for all jobs till that job and
add profit of current job to result.
In the above example, for job 1 is the latest non-conflicting
job for job 4 and job 2 is the latest non-conflicting job for
job 3.
```

The following is C++ implementation of above naive recursive method.

```
// C++ program for weighted job scheduling using Naive Recursive Method
#include <iostream>
#include <algorithm>
using namespace std;

// A job has start time, finish time and profit.
struct Job
{
    int start, finish, profit;
};

// A utility function that is used for sorting events
// according to finish time
bool myfunction(Job s1, Job s2)
{
    return (s1.finish < s2.finish);
}

// Find the latest job (in sorted array) that doesn't
// conflict with the job[i]. If there is no compatible job,
// then it returns -1.
int latestNonConflict(Job arr[], int i)
```

```
{
    for (int j=i-1; j>=0; j--)
    {
        if (arr[j].finish <= arr[i-1].start)
            return j;
    }
    return -1;
}

// A recursive function that returns the maximum possible
// profit from given array of jobs.  The array of jobs must
// be sorted according to finish time.
int findMaxProfitRec(Job arr[], int n)
{
    // Base case
    if (n == 1) return arr[n-1].profit;

    // Find profit when current job is inclueded
    int inclProf = arr[n-1].profit;
    int i = latestNonConflict(arr, n);
    if (i != -1)
      inclProf += findMaxProfitRec(arr, i+1);

    // Find profit when current job is excluded
    int exclProf = findMaxProfitRec(arr, n-1);

    return max(inclProf,  exclProf);
}

// The main function that returns the maximum possible
// profit from given array of jobs
int findMaxProfit(Job arr[], int n)
{
    // Sort jobs according to finish time
    sort(arr, arr+n, myfunction);

    return findMaxProfitRec(arr, n);
}

// Driver program
int main()
{
    Job arr[] = {{3, 10, 20}, {1, 2, 50}, {6, 19, 100}, {2, 100, 200}};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "The optimal profit is " << findMaxProfit(arr, n);
    return 0;
}
```

Output:

```
The optimal profit is 250
```

The above solution may contain many overlapping subproblems. For example if lastNonConflicting() always returns previous job, then findMaxProfitRec(arr, n-1) is called twice and the time complexity

becomes O(n*2^n). As another example when lastNonConflicting() returns previous to previous job, there are two recursive calls, for n-2 and n-1. In this example case, recursion becomes same as Fibonacci Numbers.
So this problem has both properties of Dynamic Programming, Optimal Substructure and Overlapping Subproblems.
Like other Dynamic Programming Problems, we can solve this problem by making a table that stores solution of subproblems.

Below is C++ implementation based on Dynamic Programming.

```
// C++ program for weighted job scheduling using Dynamic Programming.
#include <iostream>
#include <algorithm>
using namespace std;

// A job has start time, finish time and profit.
struct Job
{
    int start, finish, profit;
};

// A utility function that is used for sorting events
// according to finish time
bool myfunction(Job s1, Job s2)
{
    return (s1.finish < s2.finish);
}

// Find the latest job (in sorted array) that doesn't
// conflict with the job[i]
int latestNonConflict(Job arr[], int i)
{
    for (int j=i-1; j>=0; j--)
    {
        if (arr[j].finish <= arr[i].start)
            return j;
    }
    return -1;
}

// The main function that returns the maximum possible
// profit from given array of jobs
int findMaxProfit(Job arr[], int n)
{
    // Sort jobs according to finish time
    sort(arr, arr+n, myfunction);

    // Create an array to store solutions of subproblems.  table[i]
    // stores the profit for jobs till arr[i] (including arr[i])
    int *table = new int[n];
    table[0] = arr[0].profit;

    // Fill entries in M[] using recursive property
```

```
    for (int i=1; i<n; i++)
    {
        // Find profit including the current job
        int inclProf = arr[i].profit;
        int l = latestNonConflict(arr, i);
        if (l != -1)
            inclProf += table[l];

        // Store maximum of including and excluding
        table[i] = max(inclProf, table[i-1]);
    }

    // Store result and free dynamic memory allocated for table[]
    int result = table[n-1];
    delete[] table;

    return result;
}

// Driver program
int main()
{
    Job arr[] = {{3, 10, 20}, {1, 2, 50}, {6, 19, 100}, {2, 100, 200}};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "The optimal profit is " << findMaxProfit(arr, n);
    return 0;
}
```

Output:

```
The optimal profit is 250
```

Time Complexity of the above Dynamic Programming Solution is $O(n^2)$. Note that the above solution can be optimized to O(nLogn) using Binary Search in latestNonConflict() instead of linear search. Thanks to Garvit for suggesting this optimization.

References:
http://courses.cs.washington.edu/courses/cse521/13wi/slides/06dp-sched.pdf

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

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort

- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Tags: Dynamic Programming

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

Prajapati • a month ago
why cann't we use LIS approach? is it not variation of LIS?
• Reply • Share ›

Mission Peace • 2 months ago
https://www.youtube.com/watch?... has an explanation
• Reply • Share ›

paradox • 2 months ago
The non-memoized version has a bug. When finding the max job in [0,i) that doesn't conflict with job[i], we want to compare finish time of job[j] to start time of job[i] and not job[i-1]
• Reply • Share ›

Rasmi Ranjan Nayak • 3 months ago
According to the explanation,
"""We can get the maximum profit by scheduling jobs 1 and 4.
Note that there is longer schedules possible Jobs 1, 2 and 3
but the profit with this schedule is 20+50+100 which is less than 250."""
But I feel we can get maximum profit by scheduling Job 3 & 4, 100+200 = 300 and which is more than 250.
Why did not you consider Job 3 and 4 in stead Job 1 & 4.

Please help me if I have misunderstood the question.
• Reply • Share ›

Karthikeyan Sreenivasan ➚ Rasmi Ranjan Nayak • 3 months ago
@Rasmi The Q is : "Find the maximum profit subset of jobs such that no two jobs in the subset overlap".

Consider the jobs running on a single processor. Jobs 3 & 4 although yielding a higher

value, cannot be scheduled (they overlap).

Hence 1 & 4.

• Reply • Share ›

Rasmi Ranjan Nayak Karthikeyan Sreenivasan • 3 months ago

Requesting you please explain little more.
Because I did not understand how 3 and 4 will overlap with each other

• Reply • Share ›

Karthikeyan Sreenivasan Rasmi Ranjan Nayak • 3 months ago

Hi,

Job 3: {6, 19, 100}
Job 4: {2, 100, 200}

The important assumption is that, only 1 job can be run at a time.

We are running this job in a single processor. Hence at most 1 job can run from start to finish.

The star time for J3 is 6 and J4 is 2, so assuming you schedule J4 first, but the finish time for J4 is 100, which means, once job 4 is started, the next job can be scheduled only when T=100. But start time of J3 is 6, so u cannot schedule J3 as J4 will be currently running when T=6, till T=100.

Conversely, if you schedule J3 first at 6, you lose J4 since it has to start at T=2.

For 1 & 4, it is explained above. For 1 , 2 & 3 you can see that the finish time of one job does not overlap with the start time of other.

Hope this helps.

• Reply • Share ›

Rasmi Ranjan Nayak Karthikeyan Sreenivasan • 3 months ago

Hi, Sreenivasan,
You are right. Thanks for the explanation. I think I have misunderstood/did not understand properly.
Thank You.... Thanks A Lot

• Reply • Share ›

Rajdeep Podder • 3 months ago

Create a graph with these schedules. Make two different path if they overlap. Then find the longest path in graph using edge relaxation based on topological order. It will run in O(n) time.

• Reply • Share ›

Ravi • 3 months ago

is there a bug in the DP implementation? -
Just checking the single latestNonConflicted() job will not give the optimal solution, what if the second latestNonConflicted () job gives maximum profit.

|----------------| |--------m---------| |----------n---------|
_____________ |----k----|

(underscores because i was not able to align k with m while illustrating)

lastestNonConflicted for n would be m , but what if k gives more profit instead of m ??

Am i understanding the algorithm right??

• Reply • Share ›

aaman ➚ Ravi • 16 days ago

I agree. they are just checking the latest non overlaping job. which can have a conflicting job before it with higher profit. exm
0-20 $2
25-30 $300 (total profit including this point = 300+2 = 302)
26-32 $ 20 (total including this = 20 + 2 = 22)

now the 4th job if starts at say 30 sec , should have profit as (302+ current profit) . according to algo, it will be (22+profit), which is wrong.

• Reply • Share ›

aaman ➚ aaman • 16 days ago

i meant 4 job start at say 35 sec not 30 sec.

• Reply • Share ›

Deb Roy ➚ Ravi • a month ago

I think you are right, the scenario that you have covered above is not addressed in the above DP solution.

• Reply • Share ›

Fernando Ferreira • 3 months ago

This is my implementation using Binary Search. If you find something wrong, please advise me.

http://ideone.com/IOb07L

1 • Reply • Share ›

Guest • 3 months ago

@GeeksforGeeks , this is my implementation using Binary Search. If you find something

wrong, please advise me.

http://ideone.com/IOb07L

• Reply • Share ›

peng li • 3 months ago

This problem is similar to "Buy Sell stock to get maximum profit at most k times". See my local, global array solution.
http://allenlipeng47.com/Perso...

I was lazy not using binary search.

• Reply • Share ›

peng li ➚ peng li • 3 months ago

To some extent, this also looks like 0-1 knapsack problem

• Reply • Share ›

garvit • 3 months ago

Cant we do Binary search to find the latest finish time just less than or equal to the starting time to reduce the complexity to O(nLogn) since the finsh array is already sorted.

3 • Reply • Share ›

GeeksforGeeks Mod ➚ garvit • 3 months ago

Thanks for suggesting this optimization. We have added this to the original post as a note in time complexity.

• Reply • Share ›

harshit • 3 months ago

can this be done DP ?

• Reply • Share ›

NA • 3 months ago

Hello
Can anyone explain how the complexity is 2^n in first case

• Reply • Share ›

NA ➚ NA • 3 months ago

sry complexity is (n*2^n)

• Reply • Share ›

Rainer Hoffmann • 3 months ago

Hello,
the recursive function seems to actually return 200. Could you please check?
Thanks

Rainer

• Reply • Share ›

GeeksforGeeks Mod ➚ Rainer Hoffmann • 3 months ago

Thanks for pointing this out. We have fixed the issue in recursive code now.

• Reply • Share ›

novice ➚ GeeksforGeeks • 3 months ago

I have a solution. I dont know if it will work for all cases. I am new to DP and tried to apply DP for this.

Please review:

class Sched

{

int start;

int end;

int cost;

Sched(int start,int end,int cost)

{

this.start=start;

see more

• 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

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