Dynamic Programming | Set 10 (0-1 Knapsack Problem)

Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In other words, given two integer arrays val[0..n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the complete item, or don't pick it (0-1 property).

A simple solution is to consider all subsets of items and calculate the total weight and value of all subsets. Consider the only subsets whose total weight is smaller than W. From all such subsets, pick the maximum value subset.

1) Optimal Substructure:

To consider all subsets of items, there can be two cases for every item: (1) the item is included in the optimal subset, (2) not included in the optimal set.
Therefore, the maximum value that can be obtained from n items is max of following two values.
1) Maximum value obtained by n-1 items and W weight (excluding nth item).
2) Value of nth item plus maximum value obtained by n-1 items and W minus weight of the nth item (including nth item).

If weight of nth item is greater than W, then the nth item cannot be included and case 1 is the only possibility.

2) Overlapping Subproblems
Following is recursive implementation that simply follows the recursive structure mentioned above.

```
/* A Naive recursive implementation of 0-1 Knapsack problem */
#include<stdio.h>

// A utility function that returns maximum of two integers
int max(int a, int b) { return (a > b)? a : b; }

// Returns the maximum value that can be put in a knapsack of capacity W
int knapSack(int W, int wt[], int val[], int n)
{
   // Base Case
   if (n == 0 || W == 0)
       return 0;

   // If weight of the nth item is more than Knapsack capacity W, then
   // this item cannot be included in the optimal solution
   if (wt[n-1] > W)
       return knapSack(W, wt, val, n-1);

   // Return the maximum of two cases: (1) nth item included (2) not included
   else return max( val[n-1] + knapSack(W-wt[n-1], wt, val, n-1),
                    knapSack(W, wt, val, n-1)
                  );
}

// Driver program to test above function
int main()
{
    int val[] = {60, 100, 120};
    int wt[] = {10, 20, 30};
    int  W = 50;
    int n = sizeof(val)/sizeof(val[0]);
    printf("%d", knapSack(W, wt, val, n));
    return 0;
}
```

It should be noted that the above function computes the same subproblems again and again. See the following recursion tree, K(1, 1) is being evaluated twice. Time complexity of this naive recursive solution is exponential (2^n).

```
In the following recursion tree, K() refers to knapSack().  The two
parameters indicated in the following recursion tree are n and W.
The recursion tree is for following sample inputs.
wt[] = {1, 1, 1}, W = 2, val[] = {10, 20, 30}

                        K(3, 2)         ---------> K(n, W)
                     /            \
                   /                \
               K(2,2)                  K(2,1)
             /       \                 /    \
           /           \             /        \
       K(1,2)      K(1,1)        K(1,1)     K(1,0)
       /  \          /   \         /   \
     /      \      /       \     /       \
K(0,2)  K(0,1)  K(0,1)  K(0,0)  K(0,1)   K(0,0)
Recursion tree for Knapsack capacity 2 units and 3 items of 1 unit weight.
```

Since suproblems are evaluated again, this problem has Overlapping Subprolems property. So the 0-1 Knapsack problem has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by constructing a temporary array K[][] in bottom up manner. Following is Dynamic Programming based implementation.

```
// A Dynamic Programming based solution for 0-1 Knapsack problem
#include<stdio.h>

// A utility function that returns maximum of two integers
int max(int a, int b) { return (a > b)? a : b; }

// Returns the maximum value that can be put in a knapsack of capacity W
int knapSack(int W, int wt[], int val[], int n)
{
   int i, w;
   int K[n+1][W+1];

   // Build table K[][] in bottom up manner
   for (i = 0; i <= n; i++)
   {
       for (w = 0; w <= W; w++)
       {
           if (i==0 || w==0)
               K[i][w] = 0;
           else if (wt[i-1] <= w)
                 K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);
           else
                 K[i][w] = K[i-1][w];
       }
   }

   return K[n][W];
}

int main()
{
    int val[] = {60, 100, 120};
```

```
    int wt[] = {10, 20, 30};
    int  W = 50;
    int n = sizeof(val)/sizeof(val[0]);
    printf("%d", knapSack(W, wt, val, n));
    return 0;
}
```

Time Complexity: O(nW) where n is the number of items and W is the capacity of knapsack.

References:
http://www.es.ele.tue.nl/education/5MC10/Solutions/knapsack.pdf
http://www.cse.unl.edu/~goddard/Courses/CSCE310J/Lectures/Lecture8-DynamicProgramming.pdf

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.

Baggy • 16 days ago
This solution is returning lowest value instead of highest value. Shouldn't the answer be 300 (10wt * 5 = 60*5 value)? But, it returns 220

Ashish Maheshwari Baggy • 14 days ago
your logic (which gives answer as 300) is applicable when items can be included multiple times..

but in the above code/algo, each item can be included only once,
hence we do K[i-1] without taking into consideration wether nth item was included or not.

• Reply • Share ›

Indrasen Singh • 20 days ago

I think the line K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);
should be K[i][w] = max(val[i-1] + K[i][w-wt[i-1]], K[i-1][w]);

• Reply • Share ›

Baggy ➚ Indrasen Singh • 16 days ago

yes. seems with this change, 300 (right solution) is seen

• Reply • Share ›

Andy Toh • a month ago

Here is my template meta solution to the Knapsack problem. It starts off with a quicksort, and then the knapsack algorithm itself.

http://ideone.com/K8WIWB

• Reply • Share ›

raya • 2 months ago

item 1 2 3 4 weight 2 6 4 8 value in rs. 12 16 30 40 knapsack capacity given as w+ 12 analyze the knapsackproblem usingmemory functions with the help of the values given below

• Reply • Share ›

raya • 2 months ago

item 1 2 3 4 weight 2 6 4 8 value in rs.knapsack capacity given as w+ 12 analyze the knapsackproblem usingmemory functions with the help of the values given below

• Reply • Share ›

Mission Peace • 2 months ago

https://www.youtube.com/watch?... Check out my video explaining this solution

• Reply • Share ›

Mohammed • 2 months ago

I think this line :
K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);

should be :
K[i][w] = max(val[i] + K[i-1][w-wt[i-1]], K[i-1][w]);
Am I correct ?

• Reply • Share ›

nagendra ➚ Mohammed • 2 months ago

its because the input is indexed with 0 and not 1,so the i th profit is actually at i-1 position

• Reply • Share ›

sagar ➚ Mohammed • 2 months ago

I agree

• Reply • Share ›

Cracker • 2 months ago

http://algods-cracker.blogspot...

• Reply • Share ›

Tejwinder • 3 months ago

I am using a lookup table instead of bottom up solution.

http://ideone.com/QI4pcj

how to calculate the complexity of lookup solution ??

• Reply • Share ›

Aditya Goel ➚ Tejwinder • 3 months ago

Complexity remains the same of both tabular and memoized method. For memoized version, one can find complexity by (No_of_distinct_subproblems)* (time_spent_per_subproblem_without_considering_recursion)

• Reply • Share ›

Tejwinder ➚ Aditya Goel • 3 months ago

Thanks~

can you share how to calculate :

No_of_distinct_subproblems

(shall we use recursion tree ??)

• Reply • Share ›

Aditya Goel ➚ Tejwinder • 3 months ago

Better if you watch these MIT video lectures for understanding this- Lecture #19-22 (Dynamic Programming)
http://ocw.mit.edu/courses/ele...

• Reply • Share ›

paddy • 3 months ago

When I construct a tree for the given example i.e. val = {60, 100, 120};wt = {10, 20, 30}; , W = 50; I did not find any overlapping subproblems. Am I missing something here, please explain

so, I did not find any overlapping subproblems. Am I missing something here, please explain.

• Reply • Share ›

max.red • 5 months ago

Thanks a lot!

• Reply • Share ›

seshu • 5 months ago

isn't the base condition in first code should be (n == 0 || W <= 0)

• Reply • Share ›

Anurag Singh seshu • 5 months ago

No. W will never be negative. Look at if condition.
We do "W-wt[n-1]" only when wt[n-1] <= W
In case of doubts, it's always good to take the code, put enough debug messages and run it on different possible inputs. That will help to understand whats happening in code.

• Reply • Share ›

Mission Impossible • 6 months ago

Write the algorithm 0/1 knapsack which has maximum number of items and sum of weights of items is not exceeding max weight

40+10=50 (not okay) (it has less items)

5+10+15+19=49(okay) (it has more items)

consider max weight=50

don't sort the items according to weights or values in algorithm
consider (profit=weight)

2 • Reply • Share ›

SHIVAM DIXIT • 7 months ago

Can you please provide a post on space optimised knapsack?? In that we don not have to take such a large 2-d array....Its sometimes not possible to take array of size arr[n+1][capacity+1] for dp...example for such a problem is http://www.spoj.com/problems/L...

• Reply • Share ›

Guest • 7 months ago

Can anyone clarify the final answer?

wt[] goes from index 0 to 2. (3 weights)
"i" goes from 0 to n. (W is 50 so the for loop goes from 0 to 50 for i)

But the else-if block checks for wt[i-1]?

• Reply • Share ›

Sahdev • 7 months ago

Thankyou so much geeksforgeeks , you ppl made these dynamic programming problems as easy as other paradigms. :)
great great efforts

7 ^ | v • Reply • Share ›

peng • 8 months ago

```
int val[] = {60, 100, 120, 130, 140};
int wt[] = {10, 20, 30, 40, 50};
int W = 100;
```

For the input above, the two solution will have different answers.
The optimized one seems to be wrong for this input.

^ | v • Reply • Share ›

Carlos ➦ peng • 8 months ago

try

```
int val[] = {0, 60, 100, 120, 130, 140};
int wt[] = {0, 10, 20, 30, 40, 50};
int W = 100;
```

with the optimized version

2 ^ | v • Reply • Share ›

Vaitheeswaran S • 9 months ago

//space and time efficient

```
#include<iostream>

using namespace std;

int max(int a, int b)

{

return (a>b? a : b);

}

int main()

{

int n, k, m, j, i, e, d;
```

cin>>n; // no of test cases

see more

• Reply • Share ›

Guest • 9 months ago

//space and time efficient

```
#include<iostream>

using namespace std;

int max(int a, int b)

{

return (a>b? a : b);

}

int main()

{

int n, k, m, j, i;

cin>>k; // capacity of knapsack
```

see more

1 • Reply • Share ›

Subramanian • 9 months ago

give the input for value as val[]={120,100,60},for wt[]={3,2,1} and W=5.
and check.I am getting a wrong answer

1 • Reply • Share ›

Anmol • 9 months ago

Can't we use this instead?
for w = 0 to W :
for i = 0 to n:

1 • Reply • Share ›

usualraj Anmol • 7 months ago

Even I have the same doubt.
I wrote a program that would check optimal solution for a given weight 'w' i.e.
for w = 0 to W,
for i = 0 to n
However, this program is giving wrong answer. Can someone clarify?

However, this program is giving wrong answer. Can someone clarify?

• Reply • Share ›

Anurag Singh usualraj • 7 months ago

It should work. In the posted solution, table is filled from left to right. Changing the for loop order will fill table from top to bottom. Result should be same in both cases.

• Reply • Share ›

Kush Pandey • 10 months ago

this problem can be solved without dynamic programming with time complexity O(Wn) and space complexity O(1)
Then why dynamic programming?

• Reply • Share ›

Somebody Kush Pandey • 18 days ago

Where did you hear that? It's only O(nW) when using dynamic programming... otherwise it's exponential.

http://en.wikipedia.org/wiki/K...

• Reply • Share ›

Shirsh Zibbu • a year ago

I cant completely understand the meaning of the following line in the code
if "K[i-1][w]" means total value of bag, without taking current item, then what does "K[i-1][w-wt[i-1]]" mean?
this is the line:
K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);

• Reply • Share ›

nrj Shirsh Zibbu • a year ago

it means the maximum value possible with the remaining capacity of knapsack i.e w-wt[i-1] which is the total capacity minus the weight of currently picked item.
Hope it helps, if u still hv doubt reply then.

2 • Reply • Share ›

Shirsh Zibbu nrj • a year ago

k thanks
got it

1 • Reply • Share ›

nrj • a year ago

here is the implementation of knapsack using 2 rows only

```
#include<stdio.h>
/**
*@Neeraj
*knapsack using 2 rows only space complexity if O(W)
*/
#define max(a,b) (a>b?a:b)
int knapsack(int W,int wt[],int val[],int n)
{
int i,j,k,f=0;
int l[2][W+1];

for(k=0,i=0;k<=n;k++,i++)
{
if(i==2)
i=0;
if(k!=0)
```

see more

3 • Reply • Share ›

nrj • a year ago

Can someone please explain in detail why we use 2D array in some DP problems while 1D array in other DP problems, like here we use 2D array but in LIS problem we used 1D array...pls help i am newbie to DP.
Thanks!

6 • Reply • Share ›

Ajinkya Kher MI ➚ nrj • 9 months ago

It depends on the situation really..
1.) For a fibonacci calculation, all you wanted to do was see if the result of a fibonacci number was already calculated, and if so, retrieve it in constant time, instead of having to down the recursion tree yet again. This clearly need a 1d array.
Note that here there was just one input entity: The # whose fib. was to be computed.

2.) This is very similar to Longest common substring, longest common subsequence problems.. Basically pattern matching of 2 input entities: In this case, the n input elements, and the overall capacity from 0 -> W.

• Reply • Share ›

Sachin • a year ago

Could someone please tweak this code so that we can also find the items included in the knapsack finally ..

1 • Reply • Share ›

nrj ➚ Sachin • a year ago

```
#include<stdio.h>
/**
*@Neeraj
*knapsack using 2 rows only space complexity if O(W)
*/
#define max(a,b) (a>b?a:b)

int picks[100][100]={0};

int knapsack(int W,int wt[],int val[],int n)
{
int i,j,k,f=0;
int l[2][W+1];

for(k=0,i=0;k<=n;k++,i++)
{
if(i==2)
i=0;
if(k!=0)
```

see more

• Reply • Share ›

Jordan • a year ago

I like the bottom-up memoized solution, but it seem that there is a small mistake:

K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);

should be

K[i][w] = max(val[i-1] + K[i][w-wt[i-1]], K[i-1][w]);

11 • Reply • Share ›

Karshit Jaiswal ➚ Jordan • a year ago

no!! Here we consider the optimal value for the previous item plus the value obtained by adding the new item.

• Reply • Share ›

mtik • a year ago

Thanks so much! My exam is tomorrow and i've found it very helpful :)

• Reply • Share ›

Code_Addict • a year ago

Java version of above problem (both recursive and iterative):
http://ideone.com/mFaz8X

4 • Reply • Share ›

Guest • 2 years ago

there is a slight mistake in the algorithm....it should be...

the maximum value that can be obtained from n items is max of following two values.1) Maximum value obtained by n-1 items and W weight (excluding nth item).
2) Value of nth item plus maximum value obtained by n items and W minus weight of the nth item (including nth item).

notice a small change made here by me...its must be n items and not n-1 items

9 • Reply • Share ›

GeeksforGeeks Mod ➚ Guest • a year ago

The recurrence relation given in the post looks correct. You can verify the same from wikipedia (http://en.wikipedia.org/wiki/K...

Note that there is only one item of every type. Let us know if we missed anything.

2 • Reply • Share ›

mtik ➚ Guest • a year ago

Thank you.

• Reply • Share ›

chandeep • 2 years ago

How to solve the following?
What is the Minimum Amount not possible using an infinite supply of coins (Unbounded knapsack)

You are given coins of

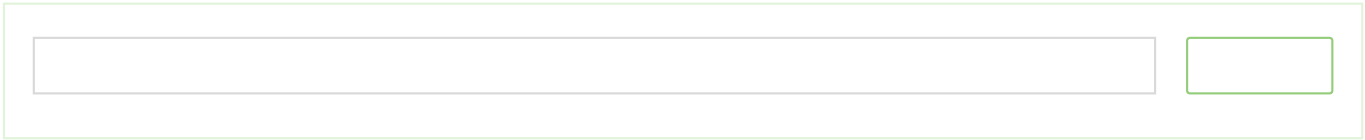
Denominations {v1, v2 , v3, v4 ....vn } of weight {w1, w2, w3 .....wn}

Now that you have a bag of capacity W .

Find the smallest Value V that is not possible to have in the bag.

• Reply • Share ›

Load more comments



- 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 · 1 hour ago

- Piyush

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

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

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

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour 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