CS F364: Design & Analysis of Algorithm

10 Dynamic Programming 0/1 Knapsack Problem

Dr. Kamlesh Tiwari
Associate Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Jan 29, 2025 ONLINE (Campus @ BITS-Pilani Jan-May 2025)

http://ktiwari.in/algo

Optimization Problems

Some problems have many acceptable solution

- Minimum Spanning Tree
- Activity Selection
- Knapsack problem
- Matrix Chain Multiplication

Then we would like to get the one which is more suitable

Dynamic Programming (DP)

DP is like divide-and-conquer, it is applied if the subproblems overlap

- DP solves each subproblem just once and then saves its result
- Typically applied to optimization problems (can have many possible solutions). And we wish to find best/worst solution
- **Elements of dynamic programming**
  1) Optimal substructure and 2) Overlapping subproblems.

DP algorithm needs following four steps

1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of optimal solution in bottom-up fashion
4. Construct an optimal solution from computed information

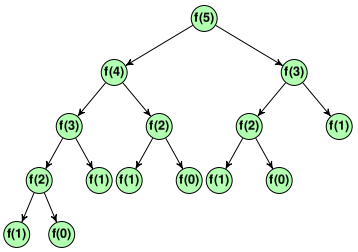
Caution: DP could be overkill

Which number next?

1, 1, 2, 3, 5, 8, 13, 21, ...?... $F(n) = F(n-1) + F(n-2)$

Algorithm 1: Fib (n)

```
1 If n ∈ {0,1} Then return 1
2 Else return Fib(n − 1) + Fib(n − 2)
```



- To find the value of $f(5)$ one need to compute

Value	Times
$f(4)$	1
$f(3)$	2
$f(2)$	3
$f(1)$	5
$f(0)$	3

Fibonacci Number

Solution of $F(n) = F(n-1) + F(n-2)$

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$$

where $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$

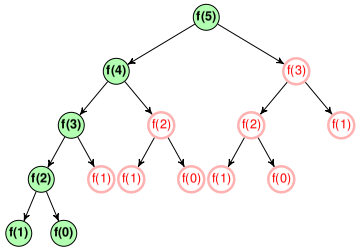
- Time needed $T(n) = T(n-1) + T(n-2) + 1$
  $T(0) = T(1) = 1$

$$T(n) = \theta(\phi^n)$$

Using Memory

Algorithm 2: Fib2 (n)

```
1 If (n ∈ {0,1}) Then { F[n] = 1; return 1}
2 If (F[n − 1] > 0) Then a = F[n − 1];
3 Else {a = Fib2(n − 1); F[n − 1] = a}
4 If (F[n − 2] > 0) Then b = F[n − 2];
5 Else {b = Fib2(n − 2); F[n − 2] = b}
6 return F[n] = a + b
```

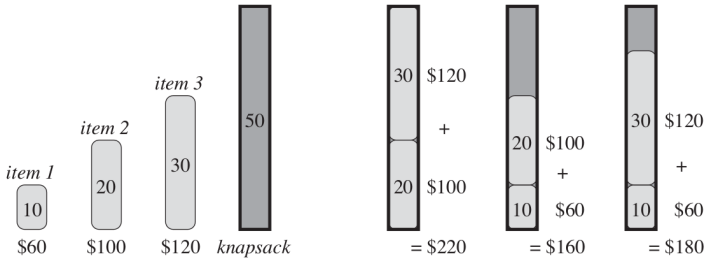


- Bottom-up approach
- Time complexity $O(n)$
- Called dynamic programming

0-1 Knapsack Problem

Let knapsack can have 50kg

3 items of wt 10, 20, 30 of price Rs 60, 100 and 120 respectively



Problem Setting

- Item I_1, $I_{2,}$, I_3, ...
- Weight w_1, w_2, w_3, ...
- Profit p_1, p_2, p_3, ...
- Knapsack with capacity W
- Selected? $x_i = 1$ if i^{th} item is selected

One have to maximize

$$\sum_{i=1}^{n} p_i \times x_i$$

subject to

$$\sum_{i=1}^{n} w_i \times x_i \leq W$$

Exponential number of possibilities arises for evaluation

Solution Sketch

$$M(i, w) = max(M(i-1, w), M(i-1, w-w_i) + p_i)$$

i	p_i	w_i	00	01	02	03	04	05	06	07	08	09	10
0													
1	3	2											
2	6	1											
3	4	4											
4	2	3											
5	5	4											
6	4	2											

Algorithm

Algorithm 3: 0/1-Knapsack (n, W)

```
1 Initialize M[ 0...n, 0...W] to zeros
2 for i from 1 to n do
3     for w from 0 to W do
4         if w < w_i then
5             M[i, w] = M[i − 1, w]
6         else
7             M[i, w] = max(M[i−1, w], p_i + M[i−1, w − w_i])
8 return M[n, W]
```

Complexity? $O(n \times W)$

Thank You!

Thank you very much for your attention! (Reference[1])

Queries ?

[1] Book - *Introduction to Algorithm*, By THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN