CS F364: Design & Analysis of Algorithm

10 Dynamic Programming 0/1 Knapsack Problem

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

Feb 08, 2021 ONLINE (Campus @ BITS-Pilani Jan-May 2021)

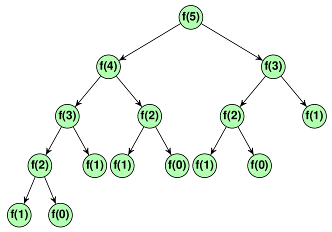
http://ktiwari.in/algo

Which number next?

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

Algorithm 1: Fib (n)

If $n \in \{0, 1\}$ **Then return** 1
Else **return** $Fib(n) + Fib(n-1)$



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

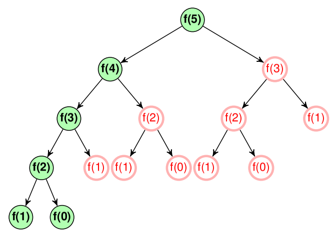
- Time needed $T(n) = T(n-1) + T(n-2) + 1$
  $T(0) = T(1) = 1$

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

Using Memory

Algorithm 2: Fib2 (n)

If $(n \in \{0, 1\})$ **Then** $\{ F[n] = 1;$ **return** $1\}$
If $(F[n-1] > 0)$ **Then** $a = F[n-1];$
Else $\{a = Fib2(n-1); F[n-1] = a\}$
If $(F[n-2] > 0)$ **Then** $b = F[n-2];$
Else $\{b = Fib2(n-2); F[n-2] = b\}$
return $a + b$

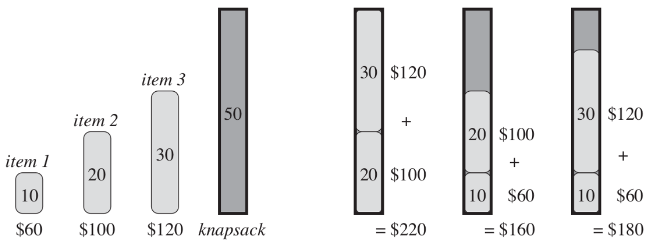


- Bottom-up approach
- Time complexity $O(n)$
- Called dynamic programming

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	9	3											
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)

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

Complexity? $O(n \times W)$

Thank You!

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

Queries ?

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