Leftover of Dynamic Programming

Amrit Kumar

Privatics team
INRIA

November 16, 2015

Bellman's equation

$$
\begin{aligned}
B(n,m) &= \max_{k\in\{0,\ldots,n\}} \left(b_m(k) + B(n-k, m-1)\right), \\
B(n,1) &= b_1(n), \\
B(0,m) &= 0,
\end{aligned}
$$

We also set $b_i(0) = 0$ for all i.

Naive recursion w/o memoization

```
Procedure maxprofit(n, k)
    if k == 1 then
        return b_1(n)
    end
    if n == 0 then
        return 0
    end
    q ← −∞
    for i ← 0 to n do
        q ← max(q, b_k(i) + maxprofit(n-i, k-1))
    end
    return q
```

Recursion w/ memoization

memo[0..n][1..k]={$-\infty$};

Procedure `DPmaxprofit`(n, k)

if $k == 1$ **then**
  return $b_1(n)$
end
if $n == 0$ **then**
  return 0
end
if $memo[n][k] == -\infty$ **then**
  $q \leftarrow -\infty$
  for $i \leftarrow 0$ **to** n **do**
    $q \leftarrow max\left(q, b_k(i) + \texttt{DPmaxprofit(n-i, k-1)}\right)$
  end
  memo[n][k]$\leftarrow q$
end
return $memo[n][k]$

Storing Optimal Choices

memo[0..n][1..k]={$-\infty$};

opt_choice[0..n][1..k]={0};

Procedure `DPmaxprofit`(n, k)

```
if k == 1 then
    return b_1(n)
end
if n == 0 then
    return 0
end
if memo[n][k] == -∞ then
    q ← -∞
    for i ← 0 to n do
        q ← max (q, b_k(i) + DPmaxprofit(n-i, k-1))
    end
    memo[n][k] ← q
    opt_choice[n][k] ← i_opt  // i for which max is obtained
end
return memo[n][k]
```

Building Optimal Solution

opt_sol[1..k]={0};

Procedure

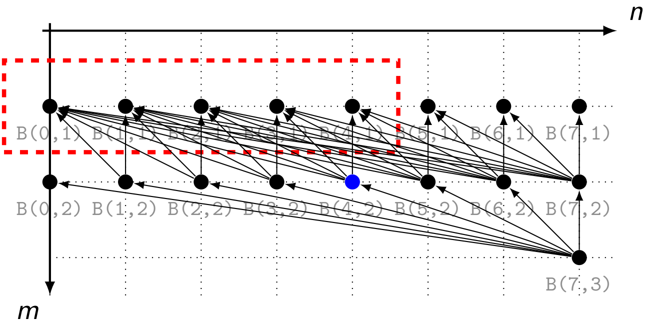
`DPmaxprofit_optsol`(*opt_choice*[0..n][1..k], n)

    for $i \leftarrow k$ **to** 1 **do**

        opt_sol[i]$\leftarrow$ opt_choice[n][i]

        $n \leftarrow n-$opt_choice[n][i]

    end



n
m
B(0,1) B(1,1) B(2,1) B(3,1) B(4,1) B(5,1) B(6,1) B(7,1)
B(0,2) B(1,2) B(2,2) B(3,2) B(4,2) B(5,2) B(6,2) B(7,2)
B(7,3)