CS 4800: Algorithms & Data

Lecture 8

February 3, 2017

Coin change

- Coin of denominations d_1, d_2, …, d_k
- Wants to make change for n cents using as few coins as possible

Memoization

- Initialize $Best[0] \leftarrow 0$
- $ComputeBest(v)$:
  - If $Best[v]$ is calculated, return $Best[v]$
  - Else
    - $Best[v] \leftarrow 1 + \min_{i=1\ldots k, d_i \leq v} ComputeBest(v - d_i)$
    - Return $Best[v]$

Bottom-up style

- $Best[0] \leftarrow 0$
- For v from 1 to n
  - $Best[v] \leftarrow 1 + \min_{i=1\ldots k, d_i \leq v} Best[v - d_i]$

Knapsack problem

- n items, each with weight w_i and value v_i
- Capacity W
- Pick items with maximum total value without exceeding capacity
- Comparison with log cutter?

First attempt

- Imitate our log cutter solution
- Best(w) : best value from sack size w
- Let the first item in the sack be item i with weight w_i and value v_i
- The rest of the solution should be the best way to fill up capacity $w-w_i$
- If not, we can sub in best solution for $w-w_i$ and get a better solution for w
- What's wrong?
- best($w-w_i$) might use item i and we cannot use i twice

Second attempt

- Our subproblems not only needs sack size w but also which items are available
- S(i, w): best value for sack size w using items i, i+1, …, n.

Formulate recursion

- Either pick item 1 or not
- $S(\{1\ldots n\},W) = \max(S(\{2\ldots n\},W), v_1+S(\{2\ldots n\},W-w_1))$

- $S(\{i\ldots n\},w) = \max(S(\{i+1\ldots n\},w), v_i+S(\{i+1\ldots n\},W1-w_i))$
- Order to evaluate?
- From large i to small i

Knapsack DP

- For j $\leftarrow 0\ to\ W$
  - S$[n+1, j] \leftarrow 0$ // base case: no item, no value
- For i $\leftarrow n\ downto\ 1$ // add one item at a time
  - For j $\leftarrow 0\ to\ W$
    - $S[i, j] \leftarrow S[i+1, j]$ // not use item i
    - If j $\geq w_i$ and $S[i+1, j] < v_i + S[i+1, j-w_i]$
      - $S[i, j] \leftarrow v_i + S[i+1, j-w_i]$ // use item i
- Return $S[1, W]$

Remember choices

- For j $\leftarrow 0\ to\ W$
  - S$[n+1, j] \leftarrow 0$ // base case: no item, no value
- For i $\leftarrow n\ downto\ 1$ // add one item at a time
  - For j $\leftarrow 0\ to\ W$
    - $S[i, j] \leftarrow S[i+1, j]$ // not use item i
    - $picked[i, j] \leftarrow False$
    - If $j \geq w_i$ and $S[i+1, j] < v_i + S[i+1, j-w_i]$
      - $S[i, j] \leftarrow v_i + S[i+1, j-w_i]$
      - $picked[i, j] \leftarrow True$
- Return $S[1, W]$

Retrace the solution

- cap = W
- sol = {}
- For i $\leftarrow 1\ to\ n$
  - If picked[i,cap] then
    - $sol \leftarrow sol \cup \{i\}$
    - $cap \leftarrow cap - w_i$

Blueprint of DP

- Model solution as a sequence of decisions
- Guess the first/last decision to reduce to smaller problems
- Relate subproblems via recurrence
- Either
  - Recursion and memoization
  - Determine evaluation order to build DP table bottom-up
- Solve original problem

Blueprint of DP in log cutter

Subproblems	Best(t): max $ from t-inch log for t=1…n
Guess	Thickness of last board
Recurrence	$Best(j) = \max_{i \leq j} p_i + Best(j - i)$
Order	Compute Best(t) for t from 1 to n
Solve orig. problem	Return Best(n)

3 items, capacity 6, $w_1=w_2=w_3=2$, $v_1=9$, $v_2=8$, $v_3=7$

i / Cap	{1,2,3}	{2,3}	{3}	none
0	0	0	0	0
1	0	0	0	0
2	Max(9,8)=9	Max(8,7)=8	7	0
3	9	8	7	0
4	Max(17,15)	15	7	0
5	17	15	7	0
6	Max(24,15)	15	7	0

Answer = S[{1,2,3}, 6] = 24