Algorithms: Design and Analysis, Part II

Dynamic Programming

An Algorithm for the Knapsack Problem

Recurrence from Last Time

Notation: let $V_{i,x}$ = value of the best solution that:

① uses only the first i items

② has total size $\leq x$

Upshot of last video: for $i \in \{1, 2, \ldots, n\}$ and any x,

$$V_{i,x} = \max \begin{cases} V_{(i-1),x} & \text{Case 1, item } i \text{ excluded} \\ v_i + V_{(i-1),x-w_i} & \text{case 2, item } i \text{ included} \end{cases}$$

our recurrence

Edge case: if $w_i > x$, must have $V_{i,x} = V_{(i-1),x}$.

Tim Roughgarden

The Subproblems

recall W and the w_i's are integral

Step 2: identify the subproblems.
- all possible prefixes of items $\{1, 2, \ldots, i\}$
- all possible (integral) residual capacities $x \in \{0, 1, 2, \ldots, W\}$

Step 3: use recurrence from Step 1 to systematically solve all subproblems.

Let A = 2-D array.

Initialize $A[0, x] = 0$ for $x = 0, 1, 2, \ldots, W$.

For $i = 1, 2, 3, \ldots, n$:

  For $x = 0, 1, 2, 3, \ldots, W$:

previously computed, available for $O(1)$-time lookup

ignore this case if $w_i > x$

$$A[i, x] := \max\{A[i-1, x],\ A[i-1, x - w_i] + v_i\}$$

Return $A[n, W]$

Running Time

Question: what is the running time of this algorithm?

(A) $\Theta(n^2)$

(B) $\Theta(nW)$

(C) $\Theta(n^2W)$

(D) $\Theta(2^n)$

$\Theta(nW)$ subproblems, solve each in $\Theta(1)$ time

Correctness: straightforward induction [use step 1 argument to justify inductive step].