Dynamic Programming (DP)

Taught in CS 6363 (Algorithms)

[Read by yourself from Cormen et al:

(i) Rod cutting problem (0-∞ Knapsack)

(ii) Activity Selection problem

(iii) Matrix chain Multiplication

(iv) Optimal bst's (v) Longest common subsequence problem]

Theory: Sometimes DAC algorithms (Divide and Conquer) have overlapping subproblems.

Ex: $\text{fib}(n) : \left\{ \begin{array}{l} \text{if } n=0 \text{ or } n=1 \text{ then return } n \\ \text{else return } \text{fib}(n-1) + \text{fib}(n-2) \end{array} \right.$

In those cases, we can implement the algorithm bottom-up, storing solutions to subproblems in some storage, avoiding recursion by looking up solutions from storage, when needed.

store fib(n) in $F[n]$.

$$\left.\begin{array}{l} F[0] \leftarrow 0 \quad F[1] \leftarrow 1 \\ \text{for } i \leftarrow 2 \text{ to } n \text{ do} \\ \qquad F[i] \leftarrow F[i-1] + F[i-2]. \end{array}\right\} O(n)$$

---

1. 0-1 Knapsack:

Input: $A[1..n] > 0$, target sum (Knapsack size) $= K$

Q: Is there a subset of $A[1..n]$, whose sum is k?

Variations:

2. Find a subset of $A[1..n]$, whose sum is as close to k as possible, without going over.

3. How many subsets of $A[1..n]$ have sum $= k$?

4. Weighted: Each $A[i]$ has size s_i and value v_i.
Find a subset with maximum total value, whose total size $\leq k$.
~~max Σ A[i] : A[i] ∈ S : Σ~~

Find a subset S of A:

$$\sum_{a_i \in S} v_i \text{ is a maximum, } \sum_{a_i \in S} w_i \leq k.$$

5. Set Partition: Q: Can $A[1..n]$ be split into 2 subsets A_1 and A_2 such that $\sum_{A[i]\in A_1} A[i] = \sum_{A[i]\in A_2} A[i]$?

6. Bisection Partition: same as set Partition..

$$+ \quad |A_1| = |A_2|$$

7. How many solutions are there to set Partition?

0,1 Knapsack: $A = \{1, 4, 9\}$ $t = \cdot$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
✓	✓			✓	✓				✓	✓			✓	✓

$$KS(i,t) = \begin{cases} \text{True} & \text{if there is a subset of } A[1..i] \text{ whose sum is } t, \\ \text{False} & \text{otherwise} \end{cases}$$

Base: $KS(0,t) = \begin{cases} \text{True} & \text{if } t = 0 \\ \text{False} & \text{if } t > 0 \end{cases}$

Recurrence: $KS(i,t) = \begin{cases} KS(i-1,t) & \leftarrow \text{if } A[i] \text{ is not needed} \\ \text{or} & \\ KS(i-1, t-A[i]) & \text{if } A[i] \text{ is needed} \end{cases}$

Answer: $KS(n, k)$.

Dynamic Program for KS: // Store $KS(i,t)$ in $KS[i,t]$

$KS[0..n, 0..k]$ // Solve increasing i, increasing t

//Base: $KS[0,0] \leftarrow$ True for $t \leftarrow 1$ to k do $KS[0,t] \leftarrow$ False

for $i \leftarrow 1$ to n do

RT = $O(n \cdot k)$. for $t \leftarrow 0$ to k do

if $t \geq A[i]$ then $KS[i,t] \leftarrow KS[i-1,t]$ or $KS(i-1, t-A[i])$

else $KS[i,t] \leftarrow KS[i-1,t]$

Knapsack – Counting/enumeration version

Input: $A[1..n] > 0$, target sum $= k$

Q: How many subsets have sum $= k$?

Define $Count(i, t) = $ # of subsets of $A[1..i]$, whose sum is t.

Base: $Count(0,t) = \{ 1$ if $t=0$, 0 if $t \neq 0$ $(\neq)$ $t>0$.

step: $Count(i,t) = Count(i-1,t) + \underbrace{Count(i-1, t-A[i])}_{\text{use this term only if } t \geq A[i]}$

DP: store $count(i,t)$ in $C[i,t]$

$C[0..n, 0..k]$, increasing i, increasing t.

// Base: $C[0,0] \leftarrow 1$ for $t \leftarrow 1$ to k do $C[0,t] \leftarrow 0$

for $i \leftarrow 1$ to n do
  for $t \leftarrow 0$ to k do
    if $t \geq A[i]$ then $C[i,t] \leftarrow C[i-1,t] + C[i-1, t-A[i]]$
    else $C[i,t] \leftarrow C[i-1,t]$.

// space can be reduced to $O(k)$ by keeping only $i-1$ & i.
Answer: $C[n,k]$.
RT $= O(n \cdot k)$.

Example: $A = 1, 2, 7, 6, 1, 5$ $k = 8$

	0	1	2	3	4	5	6	7	8
i = 0	1	0	0	0	0	0	0	0	0
1 = 1	1	1	0	0	0	0	0	0	0
2 = 2	1	1	1	1	0	0	0	0	0
3 = 7	1	1	1	1	0	0	0	1	1
4 = 6	1	1	1	1	0	0	1	2	2
5 = 1	1	2	2	2	1	0	1	3	4
6 = 5	1	2	2	2	1	1	3	5	6

subsets =

$a_1 + a_3 = 1+7 = 8$

$a_3 + a_5 = 7+1 = 8$

$a_1 + a_2 + a_6 = 1+2+5 = 8$

$a_2 + a_5 + a_6 = 2+1+5 = 8$

$a_2 + a_4 = 2+6 = 8$

$a_1 + a_4 + a_5 = 1+6+1 = 8$