Weighted Independent Set on A Path.

Input: $o-o-o-\cdots-o-o$

$v_1\ v_2 \quad \cdots \quad v_{n-1}\ v_n$

$w_1\ w_2 \quad \cdots \quad w_{n-1}\ w_n$

Output: an independent set S with maximum weight.

一个总权重最大之独立集.

$\Longrightarrow$ 没有两个点通过一条边相连.

case 1: $v_n \notin S^*$ 最优解不选择 v_n

S^* = opt for G_{n-1} 最优解就是前 $n-1$ 个点之最优解.

case 2: $v_n \in S^*$ $\cdots$ 选择 v_n, $\Rightarrow$ 不能选 v_{n-1}

S^* = opt for $G_{n-2} \cup \{v_n\}$ 最优解就是前 $n-2$ 个点之最优解.

Subproblems:

for $i \in [0, n]$, define $c[i]$ = total weight of opt for G_i

从前 i 个点选之最大独立集.

$\therefore c[n] = \max\{c[n-1], c[n-2] + w_n\}$

Recurrences:

$$\begin{cases} c[i] = \max\{c[i-1], c[i-2] + w_i\} \\ c[1] = w_1 \quad \text{base case.} \\ c[0] = 0 \end{cases}$$

Computing

1. recursion

```
recur(i)
    if i==0 or i==1
```

return base case

else if $i \geq 2$

return $\max\{recur(i-1), recur(i-2)+W_i\}$

$T(n) = T(n-1) + T(n-2) + O(1) \Rightarrow T(n) = O(c^n)$

实际在这个过程中有很 $\overset{\max}{\triangle}$ 多重复的运算，导致浪费时间

空间复杂度.

2. recursion with memoization 记录递归中间值.

global $c[0....n]$

$c[0]=0$, $c[1]=w_1$, $c(i)=-1$ for $i>1$

recur(i):

if $c(i) != -1$:

return $c(i)$

else:

$c(i) = \max\{recur(i-1), recur(i-2)+W_i\}$

return $c(i)$

$T(n) = O(N)$

3. Iteration

$c[0]=0$

$c[1]=w_1$

for $i=2$ to n

$c[i] = \max\{c[i-1], c[i-2]+W_i\}$

$T(n) = O(N)$

Reconstructing opt solution

C[0], C[1], ..., C[n]

```
if C[n] == C[n-1]      // case 1, vn ∉ S*
else                   // case 2, vn ∈ S*
```

不断重复：

```
S* = ϕ
i = n
while i ≥ 2:
    if C[i] == C[i-1]:
        i = i-1
    else
        S* = S* ∪ {vi}
        i = i-2
if i == 1:
    S* = S* ∪ {vi}
return S*
```

Dynamic Programming.

1. defining subproblem
2. finding recurrence
3. computing the optimal value for (sub) problems.
4. reconstructing the optimal solution.

Knapsack Problem 背包问题

Input: n items with $w_1, w_2 \cdots w_n$

and values $v_1, v_2, \cdots v_n$

capacity C

Output: 不超过 C 的情况 value 尽可能大

a subset of items with maximum $\sum_{i \in S} v_i$

set $\sum_{i \in S} w_i \leq C$

case 1: $n \notin S^*$

S^* = opt for first $n-1$ items with total weights $\leq C$

case 2: $n \in S^*$

$S^* = \{n\}$ + opt for first $n-1$ items

with total weights $\leq C - w_n$

Subproblems:

for $i \in [0, n]$, for $c \in [0, C]$

define $V[i][c]$ be the maximum total value of a subset

of first i items with total weight at most c.

$V[n][C] = \max\{V[n-1][C], v_n + V[n-1][C - w_n]\}$

for any $i \in [1, n]$, any $c \in [0, C]$

$$\begin{cases} V[i][c] = \max\{V[i-1][c], v_n + V[n-1][c - w_n]\} \\ V[0][c] = 0 \text{ for } c \in [0, C] \\ V[i][c] = -\infty \text{ for } c < 0 \end{cases}$$

Computing $V[i][c]$

$V[0][c] = 0$ for $c \in [0, C]$

for $i = 1$ to n

  for $c = 0$ to C

    if $w_i > c$

      $V[i][c] = V[i-1][c]$

    else

      $V[i][c] = \max\{V[i-1][c], V[i-1][c-w_i] + v_i\}$

return $V[n][C]$.

time: $O(nC)$ space: $O(nC)$

Reconstructing opt sol.

$c = C$

$S = \phi$

for $i = n$ to 1

  if $c \geq w_i$ and $V[i][c] = V[i-1][c-w_i] + v_i$

    $S = S \cup \{i\}$

    $c = c - w_i$

return S.

time: $O(N)$ space: $O(nC)$

Remark

time: $O(nC)$ → $\log_2 C$ bits to present C

pseudo polynomial time 伪多项式时间算法

space: 假设B找到值value: