

Knapsack

You are robbing a store.
 Store has n items, each with a
 weight w_i pounds
 & value v_i dollars

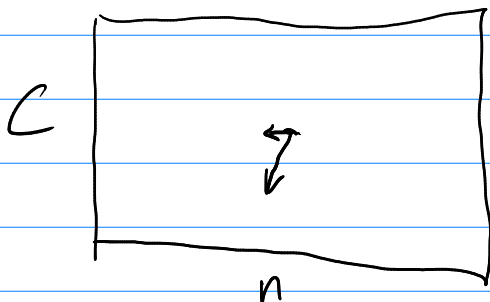
You can carry up to C pounds.
 What is the maximum total value you can take?

$f(k, w) :=$ maximum value using the
 first k items & $\leq w$
 total weight.

$$f(0, w) = 0 \quad \text{if } w \geq 0$$

$$f(k, w) = \max \begin{cases} f(k-1, w) & \leftarrow \text{don't take item } k \\ f(k-1, w-w_i) + v_i & \text{(if } w \geq w_i) \\ & \leftarrow \text{do take item } k \end{cases}$$

$O(n \cdot C)$ space & time



Knapsack Variants

Above is "0-1" knapsack.

- infinite multiplicity: can take many copies of each item.

$$f(k, w) = \max \left(\begin{array}{l} f(k-1, w) \leftarrow \text{don't take item } k \\ f(k, w - w_i) + V_i \quad (\text{if } w \geq w_i) \end{array} \right)$$

↑ can still reuse it ↪ do take item k

$f(w)$ = opt. for weight w w/ all n items:

$$f(w) = \max(0, \max_{i \in [n]} f(w - w_i) + V_i)$$

$O(C)$ space, $O(nC)$ time.

- High multiplicity:

each item usable $\leq k_i$ times

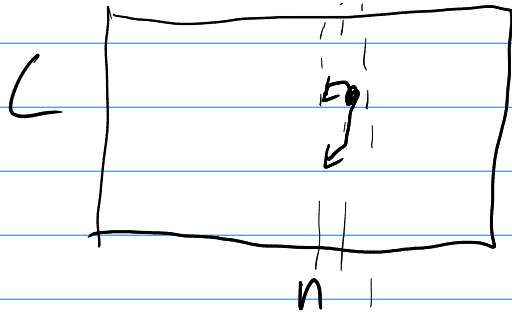
Easy: $O(C \cdot \sum k_i)$

Straightforward: $O(C \cdot \sum_{i=1}^n \log k_i)$

Tricky: $O(nC)$

- Sliding window:

For regular knapsack, get $O(C)$ space & $O(nC)$ time.



Each column only depends on previous column
 \Rightarrow only need to track two columns

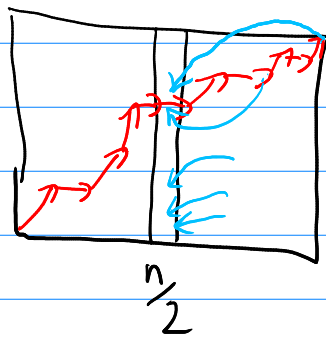
$$f(n \bmod 2, w)$$

[implementation trick: one column, scan down]

$\Rightarrow O(nC)$ time, $O(C)$ space

Issue: gives solution value but not solution
[would want the back pointers, which take nC space...]

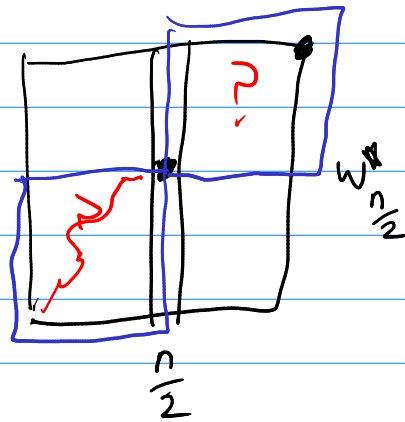
Trick:



— Solution path
— back pointer to location of path on column $n/2$

Instead of full nC back pointers, only store pointer to where the path was at column $n/2$.

This can be kept in sliding window $\Rightarrow O(C)$ space.



This finds one Point
 on optimal path in
 $O(nC)$ time, $O(C)$ space.
 Call it $w_{n/2}^*$.

Then

$$\text{Path}(\{1, \dots, n\}, C) = \text{Path}(\{1, \dots, \frac{n}{2}\}, w_{\frac{n}{2}}^*) + (\frac{n}{2}, w_{\frac{n}{2}}^*) \\
 + \text{Path}(\{\frac{n}{2}+1, \dots, n\}, C - w_{\frac{n}{2}}^*)$$

$$T(n, C) = nC + T(\frac{n}{2}, w_{\frac{n}{2}}^*) + T(\frac{n}{2}, C - w_{\frac{n}{2}}^*)$$

Total area remaining is $\frac{nC}{2}$
 $\Rightarrow O(nC)$ time.

Note Knapsack is not polynomial time

because C can be very large.

(Think: 64-bit integers $\Rightarrow n \cdot 2^{64}$)

Only fast if w_i small

(or: v_i small & w_i large, by

$f(k, v) = \text{min weight for given value}$

instead of $F(k, w) = \text{Max Value for given weight}$

)