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?

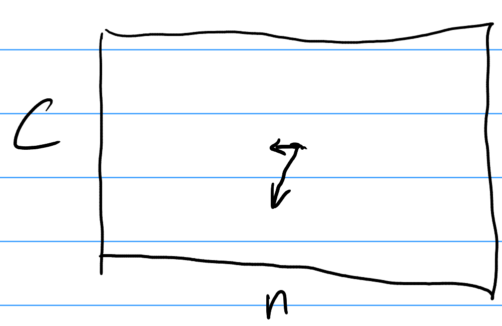
$f(k, w) :=$ maximum value using the first k items & $\leq w$ total weight.

$f(0, w) = 0$ if $w \geq 0$

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

↖ do take item k

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



Knapsack Variants

Above is "0-1" knapsack.

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

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

can still reuse it

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

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

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

- High multiplicity:

each item usable $\leq k_i$ times

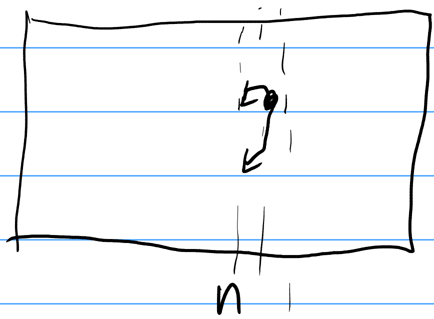
Easy: $O(C \cdot \sum k_i)$

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

Tricky: $O(nC)$

- Sliding window:

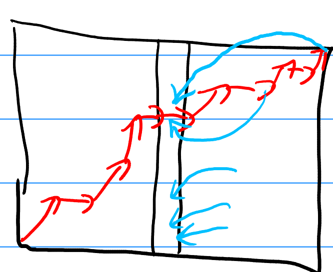
For regular knapsack, get $O(C)$ space & $O(nC)$ time

C

n

Each column only depends on previous column
⇒ only need to track two columns
$f(k \bmod 2, w)$
[implementation trick: one column, scan <u>down</u>]

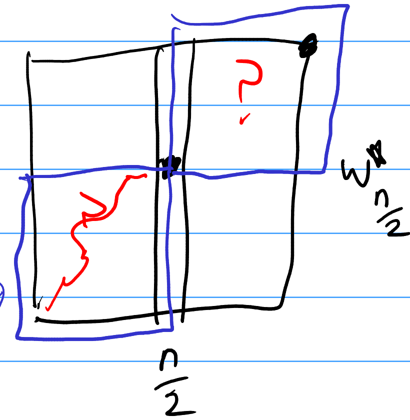
⇒ $O(nC)$ time, $O(C)$ space

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

Trick:

$n/2$

— 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 ⇒ $O(C)$ space.



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

Then

$$\text{Path}(\{1,\dots,n\}, C) = \text{Path}(\{1,\dots,\tfrac{n}{2}\}, w^*_{n/2}) + (\tfrac{n}{2}, w^*_{n/2})$$
$$+ \text{Path}(\{\tfrac{n}{2}+1,\dots,n\}, C - w^*_{n/2}).$$

$$T(n,C) = nC + T(\tfrac{n}{2}, w^*_{n/2}) + T(\tfrac{n}{2}, C - w^*_{n/2})$$

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

__Note__ Knapsack is <u>not polynomial time</u>

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
$f(k,v)$ = min weight for given value
instead of $F(k,w)$ = max value for given weight
)