- Sat 3/6 Midterm
- Wed 3/10 Lecture Jay Aslam
  Teams Only (@ WVF 020)
- cheating on HW

PB

Greedy

- Divide (split) SUBPB
  Decide
  Break
- Solve SubPb
- $\boxed{Sol}$ = combine (Subpb SOL)

Dynamic Programming.

- Look at all possible SUBPB
  dont know how to break it
- Solve all possible SUBPB
  (even ones we dont need)
- given SOL(subpb) decide the split ⟹ which SUBPB we need
- $\boxed{SOL}$ = comb (selected subpb SOL)

Brute Force

- Try all possibilities
- Keep track of obj
- Return best SOL (OPT SOL)

Act. Sel

all subsets of non-overlapping activities

$|\mathcal{P}(A)| = 2^n$

SOL

DP writing parts (required)

1. Chord OPT SOL = SPLIT Funct (sub-pb OPT SOL)
   - thinking exercise for you, rather than formal

2A. C[input] = value <u>recurrence</u> of OBJ.
   "Rec value of OPT SOL"

2B. Subproblem-dependency table/graph
   (drawing, usually a table ⇒ visual of 2A

3. Compute C[ ] table (usually all inputs/table
   usually bottom-up
   <u>sometimes</u> rec+cache top down)

4. Trace the solution/choices

5. Run Time

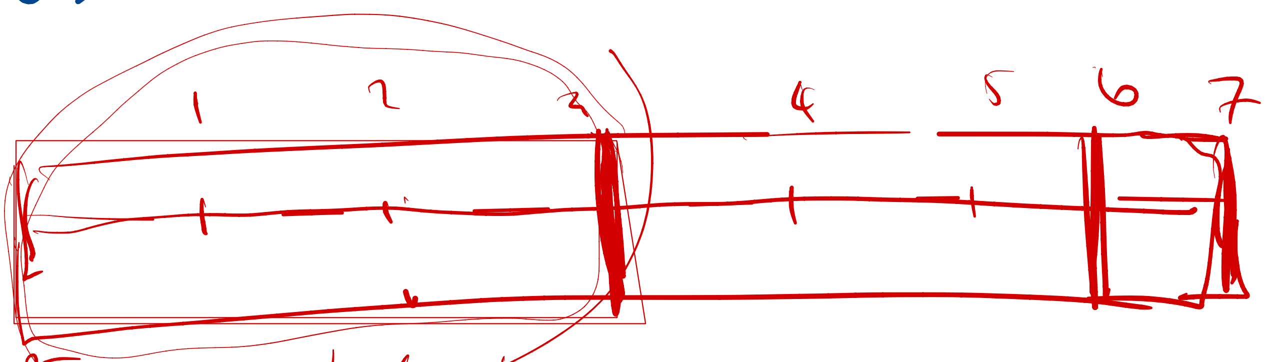
DP1 Rod-cutting n length rod.

table of prices

length	1	2	—	—	n
price	p_1	p_2			p_n

p ∀ length

Task: cut the rod to max total value



Greedy choice ≠ OPT SOL

L	1	2	3 (OPT sol)	4	5	6
V	1	2	4.5	6.4	8	
Q	1	1	1.5	1.6		
Tot			9	6.4+2		

① OPT sol charact / split

ex. $l_1=3$, $l_2=3$, $l_3=1$

⇒ OPT sol comp. for each piece of rod

$n=3$ ⇒ OPT SOL $=(l=3)$

$n=4$ ⇒ OPT SOL $=(3,1)$

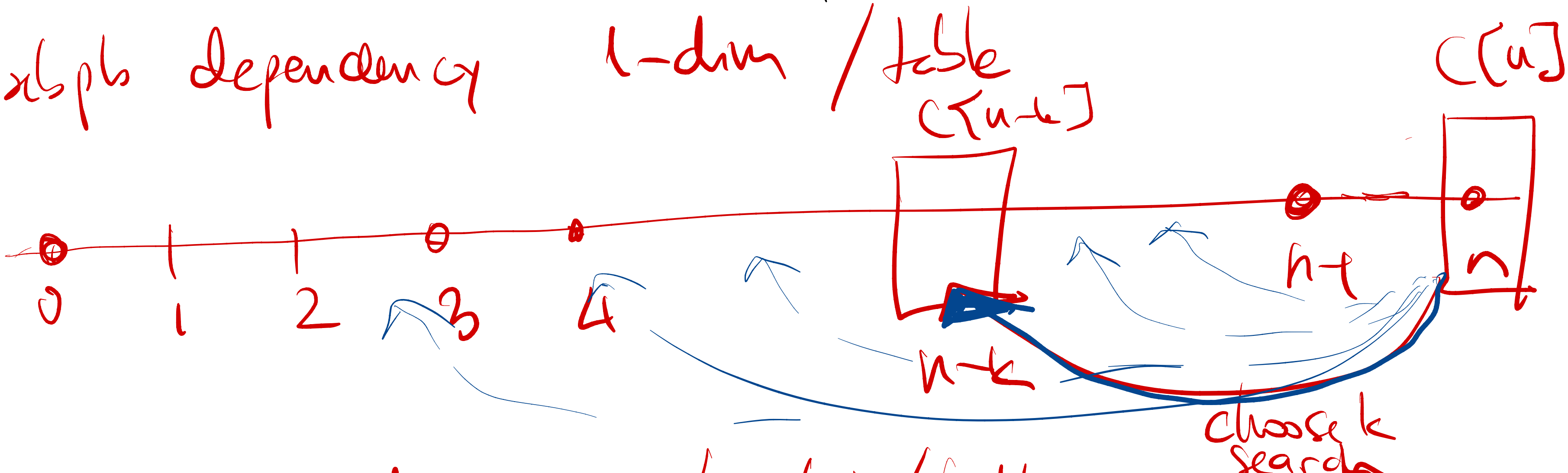
(2A) $C[n, \underbrace{p_1 | p_2, \ldots, p_n}_{\text{Global}}]$ = max total value

K = first cut (unknown)

$$C[n] = \max_{\underset{\text{(search)}}{k}} \{ \underbrace{p_k}_{\text{value at cut } k} + \underbrace{C[n-k]}_{\text{sub pb}} \}$$

• solve $C[n-k]$ all subproblems first

(2B) subpb dependency 1-dim / table



1-dim array/matrix/table

③ Fill / compute table C[] bottom up.

$C[0] = 0$

for $i = 1$: n

$C[i] = \max_{1 \le k \le n} \{ p_k + C[n-k] \}$ $\Theta(n)$ // the value

$S[i] = \arg\max_k (p_k + C[n-k])$ // the k

// I now have C[n]

④ Trace Solution

→ track it from C[] itself ⇒ procedure

→ or explicit S[input] = choice/decision ⇒ nothing

PrintSolution (n)
if n ≤ 0 exit
print S[n] = k
PrintSolution (n − k)
...

DP 2 Coin change

$d_1, d_2, \ldots, d_n$ denominations

∞ coins

Task: min # of coins

n = # cents to make

① Charact OPT SOL

k = # coins

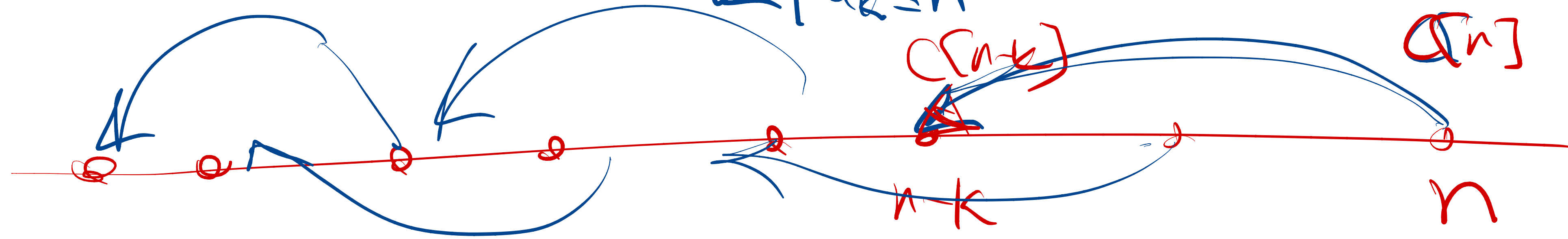
4 coins	3 coins
$d_1 \mid d_3 \mid d_1 \mid d_2$	$d_4 \mid d_5 \mid d_2$
$\underbrace{b \text{ cents}}_{\text{optimal}}$	$\underbrace{n-b \text{ cents}}_{\text{optimal}}$

② $c[n]$ = # of coins (input: n)

$c[n]$ = search for first coin d_k

$$c[n] = \min_{k \mid d_k \le n} \left\{ 1 + c[n-d_k] \right\}$$

2B



③ Fill the C[ ] table left → Right

$C[0]=0$

for $i = 1 : n$

$$C[i] = \min_{\substack{1 \le k \le n \\ d_k \le n}} \{ 1 + C[n - d_k] \}$$

$$S[i] = \arg\min_k \{ \dots - \} \Rightarrow k \text{ or } d_k$$

search for k

④ Trace Solution

$C[100] = 11$ need to find k / d_k

so $\boxed{C[100 - d_k] = 10}$

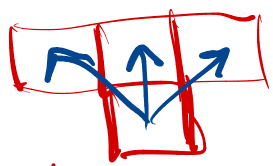
OPT sol subpb: 1, 2, , 3, 4, 5, 6

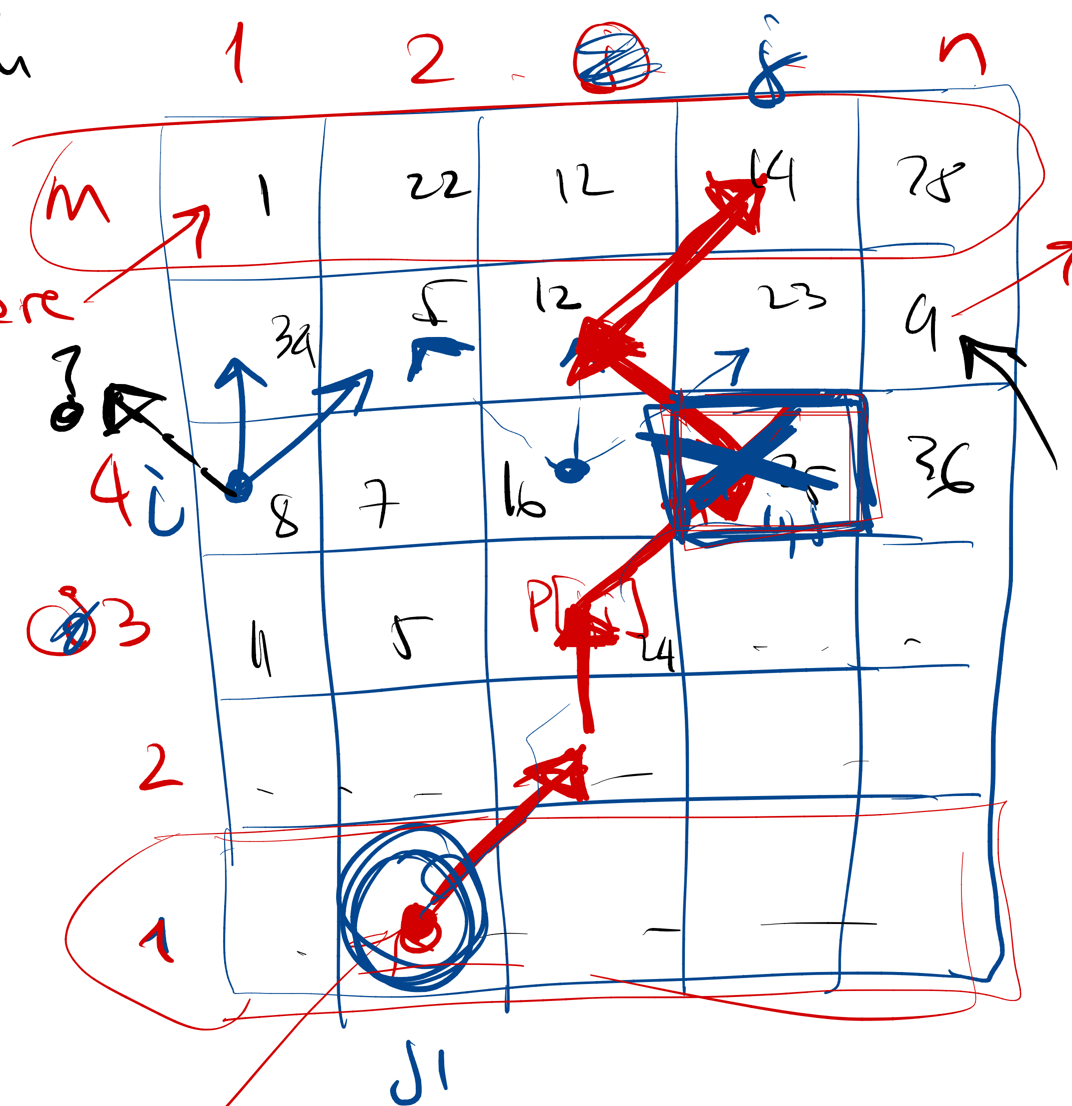
(DP3) Check board best path
min

$P[i,j]$ = penalty of stepping here
row ↓ column ↘

- start anywhere on row = 1
- finish anywher on row = m

3 moves are up one row





Task: ~~min~~ path total penalty.

(1) Character OPTSOL ⟹ new task optimal path from cell $(1, j_1)$ to cell (i, j)

new task

any path from anywhere on first row → all (i,j)

(2) $C[i,j]$ = penalty of best path to $\boxed{i,j}$

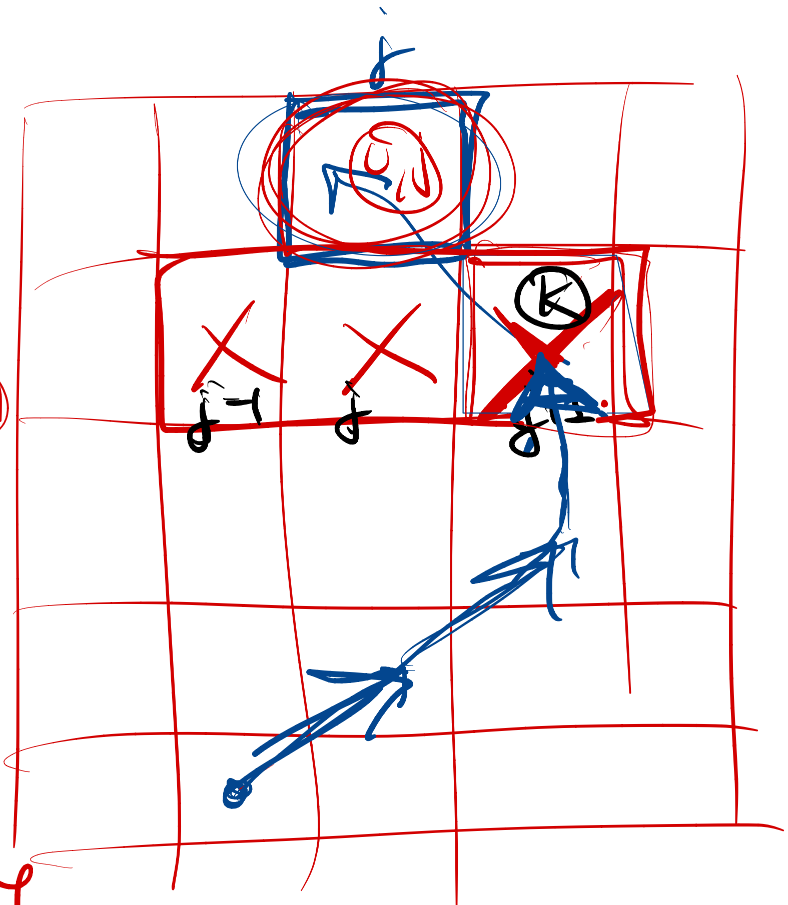
search for the last move ↗ ↑ ↖

row $i-1$, colum $j-1, j, j+1$

$j, j-1, j+1$ → k

$= P[i,j] + C[i-1, k]$

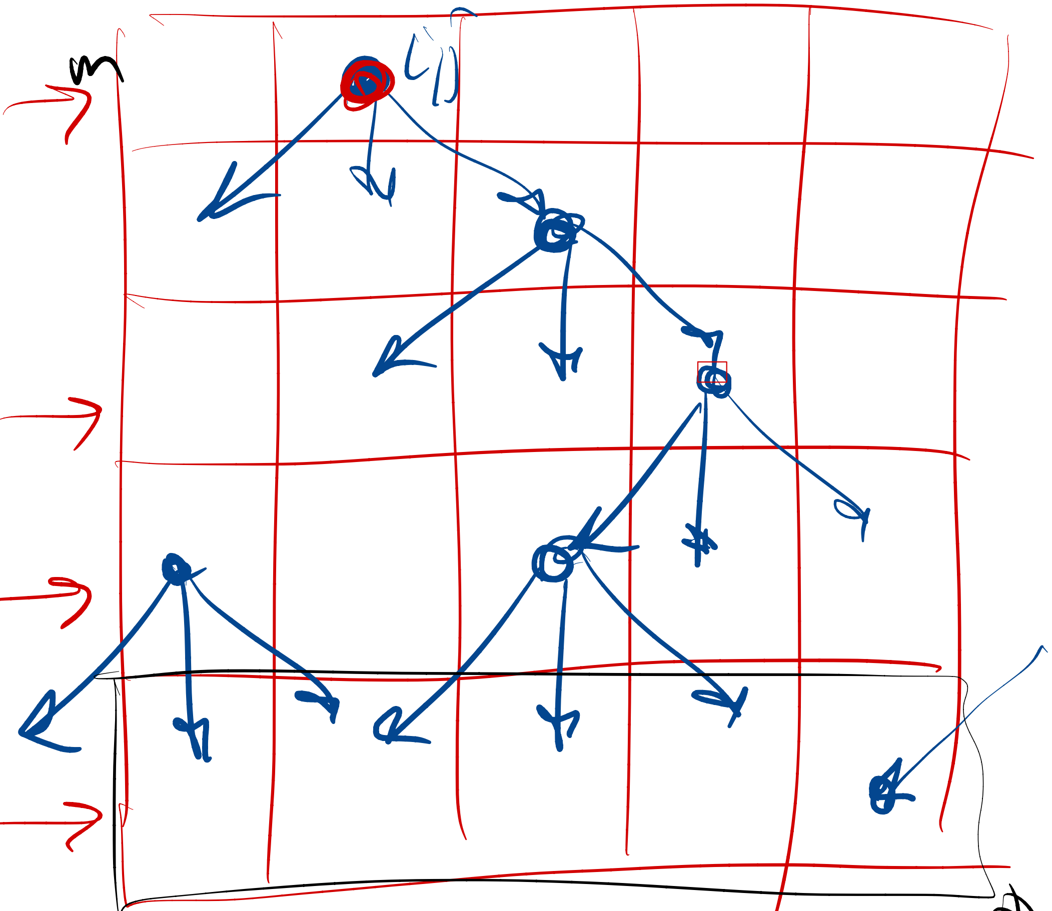
$$P[i,j] + \min \begin{cases} C[i-1, j-1] \\ C[i-1, j] \\ C[i-1, j+1] \end{cases}$$



(2b) SUBPB dependency table

at row k, must have all prev rows 1 → k-1

m →



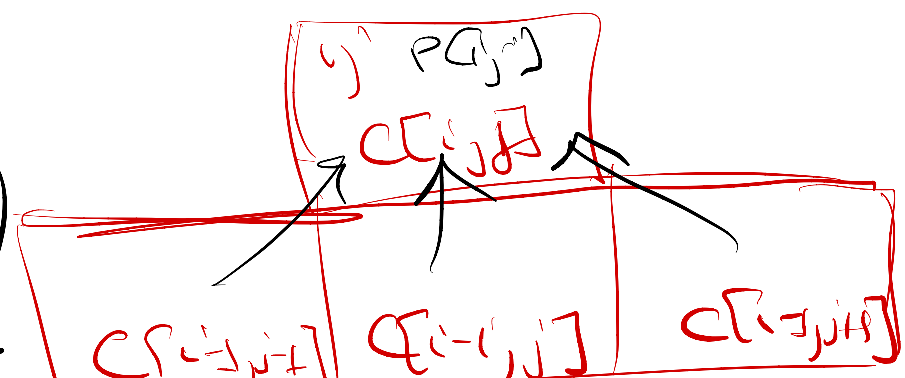
(3) bottomup

C[first row] = P[first row]

```
for r = 2 : m
    for c = 1 : n
        // solve C[r,c]
        C[r,c] = p[r,c] + min{ C[r-1, c-1], C[r-1, c], C[r-1, c+1] }
S[ ] = ? store the j underit
```

(4) Solution (i,j)

Prn P[i,j], (i,j)



j_{new} = ? C[i-1, j_{new}] + P[i,j] = C[i,j]

orig pb: - find cell $i=m$, j with min $c[i,j]$
last row
- use that cell in solved pb.

Kinda DP

$\binom{n}{k}$ = # of subsets of size k out of n

= # ways to pick k items out of n

$C[n,k]$ = # of ways . . . = $\binom{n}{k}$

1. indexed

1	2	3	. . n-1 objects	n

choose k out of n

2 options

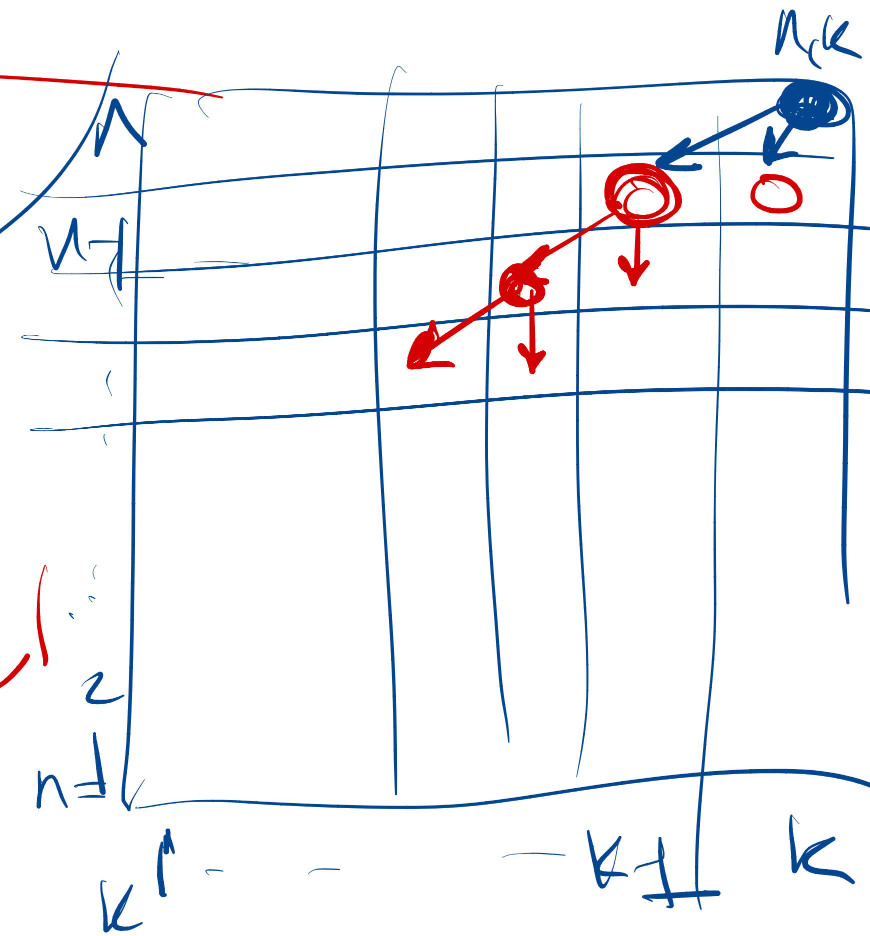
→ choose item n $C[n-1,k-1] = \binom{n-1}{k-1}$

→ do not choose item n $C[n-1,k]$

2.

$$\boxed{C[n,k] = C[n-1,k-1] + C[n-1,k]}$$

n	
1	1
2	1 2 1
3	1 3 3 1
4	1 4 6 4 1
5	1 5 10 10 5 1
6	1 6 15 20 15 6 1



n

Discrete Knapsack

	1	2	3	...		—	n
values	v_1	v_2	v_3	—		—	v_n
weights	w_1	w_2	w_3	—		—	w_n

Knapsack max weigh W

Task: select the max-value subset of items subject to total weight $\leq W$

$$C[W, \underset{\{1,2,\dots n\}}{\text{item\_set}}]$$

$$\text{choose } k = \underset{\text{item}}{v_k} + C[W-w_k, \text{item\_set} \setminus \{k\}]$$

not good