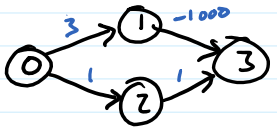
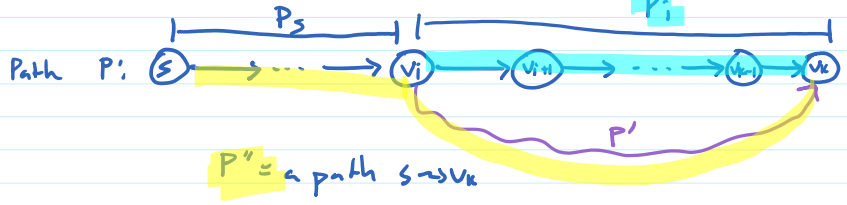
Negative Weights

Negative weights → ~~Dijkstra~~



If $s, v_1, v_2, \ldots, v_k$ is a shortest path $s \rightsquigarrow v_k$ then

for each v_i, $v_i, v_{i+1}, \ldots, v_k$ is a shortest path $v_i \rightsquigarrow v_k$ (P_i)



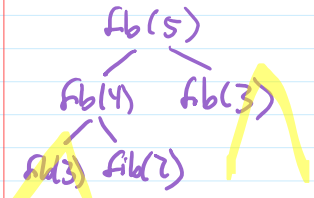
~~$l(P_s)$~~ $+ l(P_i) = l(P) \leq l(P'') =$ ~~$l(P_s)$~~ $+ l(P')$

$l(P_i) \leq l(P')$

P_i as short as shortest path P', so P_i is itself shortest

we want $d_s(n-1, v)$ (HWK)

Let $d_s(i, v)$ = total weight of min-weight path from $s \rightsquigarrow v$ with $\leq i$ edges



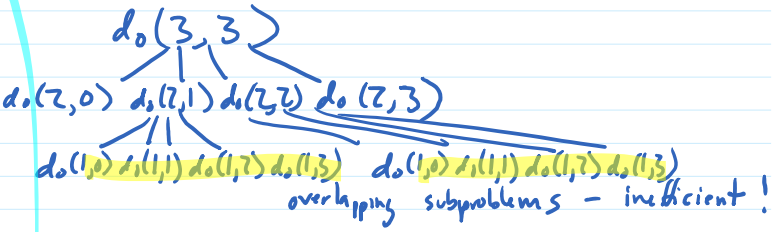
$$= \begin{cases} 0 & \text{if } v = s \\ \infty & \text{if } i = 0 \text{ and } v \neq s \\ \min\left(d_s(i-1, v), \min_{u \in V} d_s(i-1, u) + l(u, v)\right) & \end{cases}$$

∞ if no edge not there

SHORTEST-PATHS(s, i, v) Memoized

```
if v=s return 0
else if i=0 return ∞
else if (i,v) in Memo return Memo[i,v]

else
    dist ←
    Memo[i,v] ← dist
    return dist
```

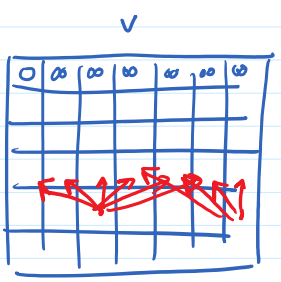


$O(n)$ to compute each entry in Memo
$O(n^2)$ entries in memo
$O(n^3)$ total

SHORTEST-PATHS(n, s) Dynamic Programming

```
M ← n×n array
for v=0 to n-1  M[0,v] ← ∞
M[0,s] ← 0

for i=1 to n-1
    for v=0 to n-1
        M[i,v] =  [use recursive formula]
    M[i-1] ← NIL  (delete entire row)
```



for v = 0 to n-1
  M[i,v] = [use recursive formula]
M[i-1] ← NIL (delete entire row)

$O(n^3)$ overall

```
SHORTEST-PATHS(n, s)

    M ← n×n array
    for v = 0 to n-1   M[0,v] ← ∞
    M[0,s] ← 0

    for i = 1 to n-1                                              O(n)
        for v = 0 to n-1
            M[i,v] = min( M[i-1,v], min_{u=0...n-1} M[i-1,u] + l(u,v) )   ] O(m)
```

rework to iterate over all edges

$O(n \cdot m)$ overall

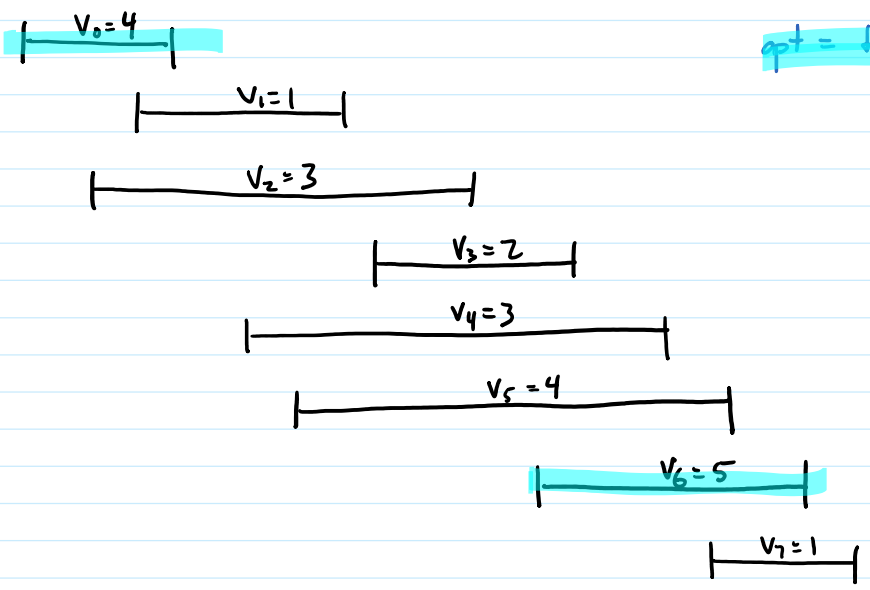
```
SHORTEST-PATHS(n, s)

    M ← n×n array
    for v = 0 to n-1   M[0,v] ← ∞
    M[0,s] ← 0

    for i = 1 to n-1
        for v = 0 to n-1
            M[i,v] = min( M[i-1,v], min_{u=0...n-1} M[i-1,u] + l(u,v) )
```

Weighted Interval Selection

index of last that finishes before i starts



opt = total value of 9

i	P(i)
0	-1
1	-1
2	-1
3	1
4	0
5	0
6	2
7	4

want OPT(8)

OPT(j) = value of optimal selection using activities $< j$

$$= \begin{cases} 0 & \text{if } j = 0 \\ \max\left(\underbrace{OPT(P(j-1)+1) + value_{j-1}}_{\text{use interval } j-1}, \underbrace{opt(j-1)}_{\text{don't use it}} \right) \end{cases}$$

COMPUTE-OPT(n, v, P)

M ← n-elt array
M[0] ← 0
for j = 1 to n
  M[j] ← max(v[j-1] + M[P[j-1]+1], M[j-1])
  USE[j] ← M[j] ≠ M[j-1]

O(n)
O(1)
O(n) overall

SELECT(OPT, j)

Seam Carving

https://www.youtube.com/watch?v=6NcIJXTlugc

M[i,j] = cost of min-cost seam from bp to row i, col j

= {

Energy

1	1	1	4	1	2
3	6	5	1	0	3
2	4	3	2	2	0
0	2	1	6	5	3
1	1	0	2	1	0

1	1	1	4	1	2

Subset Sum

Given set of positive integers S and sum k, find

Example: $S = \{1, 4, 9, 13, 16, 20\}$ $k = 28$

Brute Force:

Substructure:

$OPT(i, w) =$

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	2	0	1	1	1	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	.	.	.			
9	3	0	1	1	1	4	5	5	5	5	9	10	10	10	13	14	14	14	14	-	-	-	-	.						
13	4	0	1	1	1	4	5	5	5	5	9	10	10	10	13	14	14	14	17	18	18	18	18	22	23	23	23	26	27	27
16	5	0	1	1	1	4	5	5	5	5	5	10	10	10	13	14	14	14	17	18	16	20	21	21	21	21	25	26	27	27
20	6	0	1	1	1	4	5	5	5	5	5	10	10	10	13	14	14	14	17	18	18	20	21	21	21	24	25	26	27	27

$S = \{1, 4, 9, 13, 16, 20\}$ $k = 23$

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

1
4
9
13
16
20

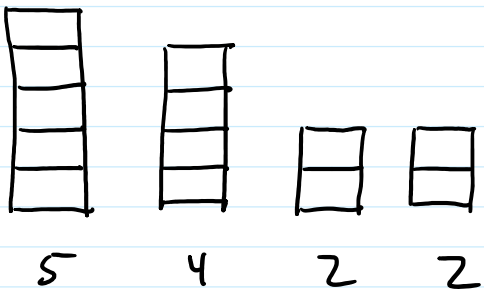
k-Combination of $\{1, \ldots, n\}$ is

$$\binom{n}{k} =$$

$$= \begin{cases} \end{cases}$$

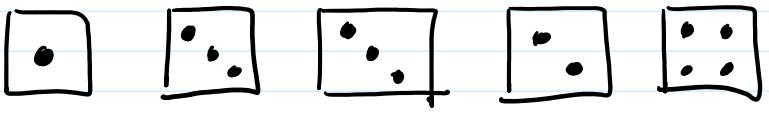
```
CHOOSE(n, k)
    if k<0 or k>n return 0
    C ← n×n array
    C[0,0] = 1
```

return C[n,k]



Count(n,h) = # nonincreasing sequences of length n, height h

$= \{$



How many sequences of 5 die rolls sum to 13?

Count(n,k) = # sequences of n rolls that sum to k

$= \{$