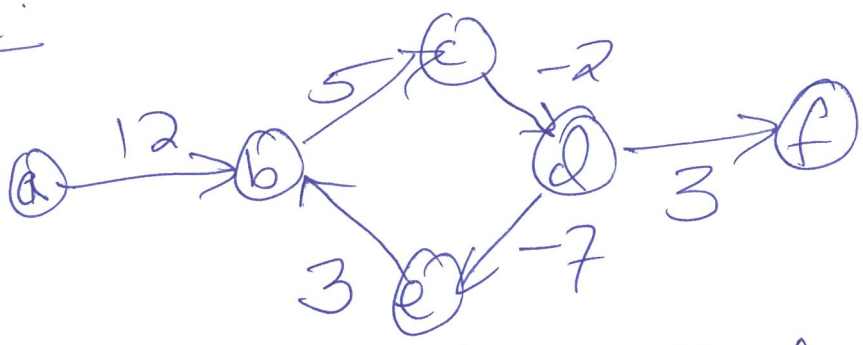
Dijkstra's algorithm:

Shortest path from s to all other vertices
But assumed positive edge lengths.

What if there are negative weight edges?
(or negative lengths)

Is the problem still well-defined?

Example:



What's the shortest path from a to f?

The cycle $b \to c \to d \to e \to b$
has total weight -1
So it's called a negative weight cycle.

Our goal: Given a graph with edge weights, determine if there is a negative weight cycle. If there are none then determine for all pairs of vertices the shortest path length between them.

For $G=(V,E)$, let $V=\{1,2,\ldots,n\}$ so the vertices are numbered $1,2,\ldots,n$.

<u>Goal:</u> find $n \times n$ matrix D where

$D(i,j)=$ length of the shortest path from i to j.

<u>Dijkstra's algorithm high-level idea:</u>

$dist(s)=0$

for all $w \neq s$, $dist(w)=\infty$

$R=\emptyset$

while $R \neq V$

- Let w be vertex not in R with min $dist(w)$
- Add w to R
- Update $dist(z)$ for all $(w,z) \in E$

<u>Thus:</u> $dist(z)=$ length of shortest path from s to z only using vertices in R as intermediate vertices.

Back to the general problem,

let's assume no negative weight cycles for now,

& let's compute D.

We'll use Dynamic Programming.

<u>Dijstra's:</u>

$dist(z)$ = length of shortest $s \to z$ path only using R as intermediate vertices

Here: use prefix of input so $\{1, \ldots, k\}$ instead of R

For $0 \le k \le n$, $1 \le i \le n$, $1 \le j \le n$,

let $C_k(i,j)$ = length of shortest path from i to j only using vertices in $\{1, \ldots, k\}$ as intermediate vertices.

<u>Our goal:</u> compute $C_n = D$.

<u>Base case:</u> $k=0$

$$C_0(i,j) = \begin{cases} w(i,j) = \text{weight of edge } i \to j & \text{if } \overrightarrow{ij} \in E \\ \infty & \text{if } \overrightarrow{ij} \notin E \end{cases}$$

For $k>0$,

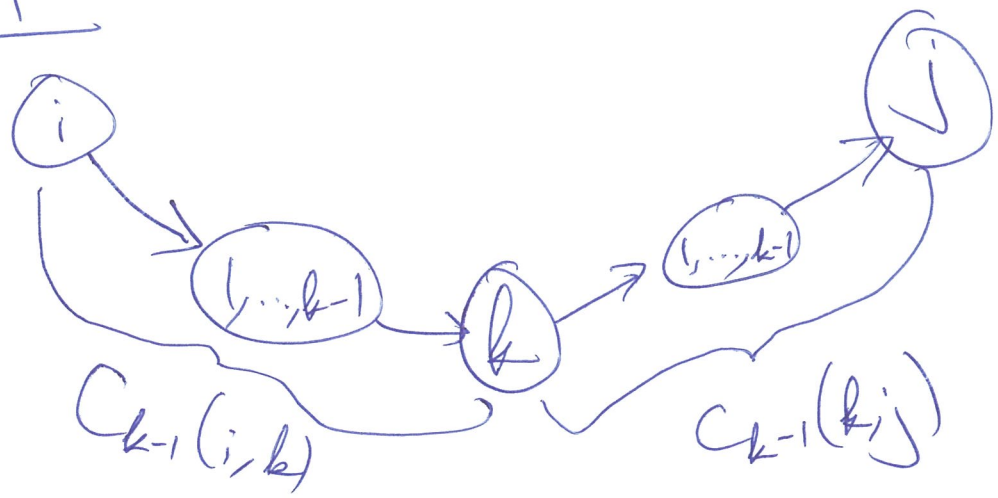
either the shortest $i \rightsquigarrow j$ path using $\{1,\dots,k\}$ as intermediate:

1) uses k as an intermediate vertex

or 2) doesn't use k.

<u>Case 2:</u> $C_k(i,j) = C_{k-1}(i,j)$

<u>Case 1:</u>



$$C_k(i,j) = C_{k-1}(i,k) + C_{k-1}(k,j)$$

Therefore, for $k \geq 1$:

$$C_k(i,j) = \min\{C_{k-1}(i,j),\ C_{k-1}(i,k) + C_{k-1}(k,j)\}$$

<u>Floyd-Warshall(G,w)</u>

input: directed $G=(V,E)$ with $V=\{1,\dots,n\}$
& for $\overrightarrow{ij} \in E$ $w(i,j)$ = weight of edge $i \to j$

output: matrix D

```
for i=1 → n
    for j=1 → n
        if i=j
            if ij ∈ E then C(i,j,0) = w(i,j)
                      else C(i,j,0) = ∞
        if i=j then C(i,i,0) = 0
for k=1 → n
    for i=1 → n
        for j=1 → n
            C(i,j,k) = min{C(i,j,k-1), C(i,k,k-1)
                           + C(k,j,k-1)}
Return(C(·,·,n))
```

Running time: $O(n^3)$

What about negative weight cycles?

Suppose vertex i is on a negative weight cycle C.

Then there is a path from i to i of negative weight.

Hence, $C(i,i,n) < 0$

So look at diagonal entries,

if there is an i where $C(i,i,n) < 0$

then there is a negative weight cycle which includes vertex i

else if all diagonal entries $= 0$

then no negative weight cycles.

& $C(i,j,n) =$ length of shortest $i \to j$ path

$= D(i,j)$.