CMSC 451 Oct. 19

Dynamic Programming (IV)

All-Pair Shortest Path.

Given weighted digraph $G = (V, E)$, w/ weight w.

Single-Source Shortest Path (SSSP) (Sep. 14 lecture)

for some source vertex s, find shortest path $s \to u$, $\forall u$.

$n = |V|$ $m = |E|$

Dijkstra	$O(m + n \log n)$	non-negative weights only
Bellman-Ford	$O(nm)$	no neg. weight cycle.

All-Pair Shortest Path. (APSP)

for any $u, v \in V$, compute shortest path $u \to v$.

Repeat Dijkstra for any s $\to O(n \cdot (m + n \log n))$ / $O(n^2 m)$: Repeat B-F.

Floyd-Warshall (1960s)	$O(n^3)$	negative weight allowed
Claude (last week)	$O(n^{2.9995})$	

Input / Output Format

Dijkstra / B-F, adjacency list.

Floyd-Warshall adjacency matrix.

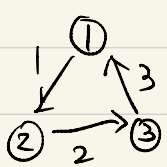
APSP result $d[u, v]$, $\forall u, v$. $O(n^2)$

no point to save space using adj. list.

Weighted Adjacency matrix. W

$w[u, v] = +\infty \Rightarrow$ no edge $u \to v$

$= w \Rightarrow \exists$ an edge $u \to v$, w/ weight w.



w	1	2	3
1	0	+∞	3
2	1	0	+∞
3	+∞	2	0

Output Format: $d[u, v]$, distance of shortest path $u \to v$.

to recover the actual path,

$h[u, v]$. "hook" between $u \to v$, can be any midpoint on s. path $u \to v$.

$next[u, v]$. next step from u, on the s.p. $u \to v$.

DP Formulation

How to reduce shortest paths into subproblems?

problems. $d[u, v]$

- By # edges in the path?

  $d^{(\ell)}(u, v)$ = shortest path $u \to v$ w/ at most ℓ edges.

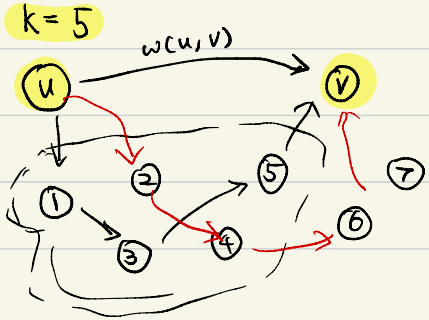
Leads to a slower algorithm!

- Floyd & Warshall: restrict the set of vertices the path can go through.

  $d^{(k)}(u, v)$ = cost of shortest path $u \to v$. s.t. only vertices $1, \cdots, k$ can be used as intermediate stops.

  $d^{(n)}(u, v) = d(u, v)$

Example: $k = 5$



Base Case: $d^{(0)}(i,j) = w(i,j) \quad \forall\, i,j \in V$

$k \geq 1$: consider $i,j \in V$. $d^{(k)}(i,j)$ = shortest path $i \to j$ s.t. only $\{1,\dots,k\}$ is allowed as int. stops.

Case #1: shortest path does not pass through k.

$\Rightarrow$ only $\{1,\dots,k-1\}$ can be used

$\Rightarrow$ cost is $d^{(k-1)}(i,j)$

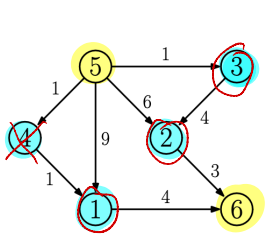
Case #2: Shortest path pass through k.

since no negative cost cycle, k is visited once

$\Rightarrow$ path is $i \to k \to j$ ← still only $\{1,\dots,k\}$ allowed.

each part is a path with $\{1,\dots,k-1\}$ as intermediates

$\Rightarrow$ cost is $d^{(k-1)}(i,k) + d^{(k-1)}(k,j)$



$d^{(0)}_{5,6} = \infty$ (no path)

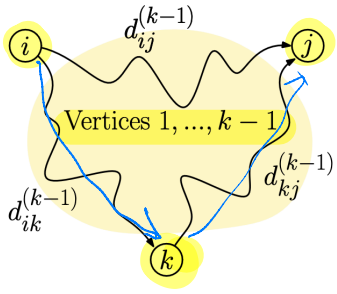
$d^{(1)}_{5,6} = 13$ $\langle 5,1,6 \rangle$

$d^{(2)}_{5,6} = 9$ $\langle 5,2,6 \rangle$

$d^{(3)}_{5,6} = 8$ $\langle 5,3,2,6 \rangle$

$d^{(4)}_{5,6} = 6$ $\langle 5,4,1,6 \rangle$

$d^{(5)}_{5,6} = d^{(6)}_{5,6} = 6$ (no change)



$$d^{(k)}(i,j) = \begin{cases} w(i,j) & k = 0 \\ \min\left(d^{(k-1)}(i,j),\ d^{(k-1)}(i,k) + d^{(k-1)}(k,j) \right) & k \geq 1 \end{cases}$$

Bottom-up:

Floyd-Warshall(w)

```
for i ← 1 to n
    for j ← 1 to n
        d[i,j] ← w[i,j]                    } d ← d^(0)

for k ← 1 to n.                            before: d = d^(k-1)
    for i ← 1 to n.
        for j ← 1 to n.                    } d ← d^(k)
            d^(k)[i,j] ← min(d^(k-1)[i,j], d^(k-1)[i,k] + d^(k-1)[k,j])

return d[-,-]
```

Running time: $O(n^3)$

Space complexity : $O(n^2)$

Why? $d^{(k)}[i,j]$ only needs $d^{(k-1)}[_,_]$.

Therefore, only 2 copies of the d-array is required.

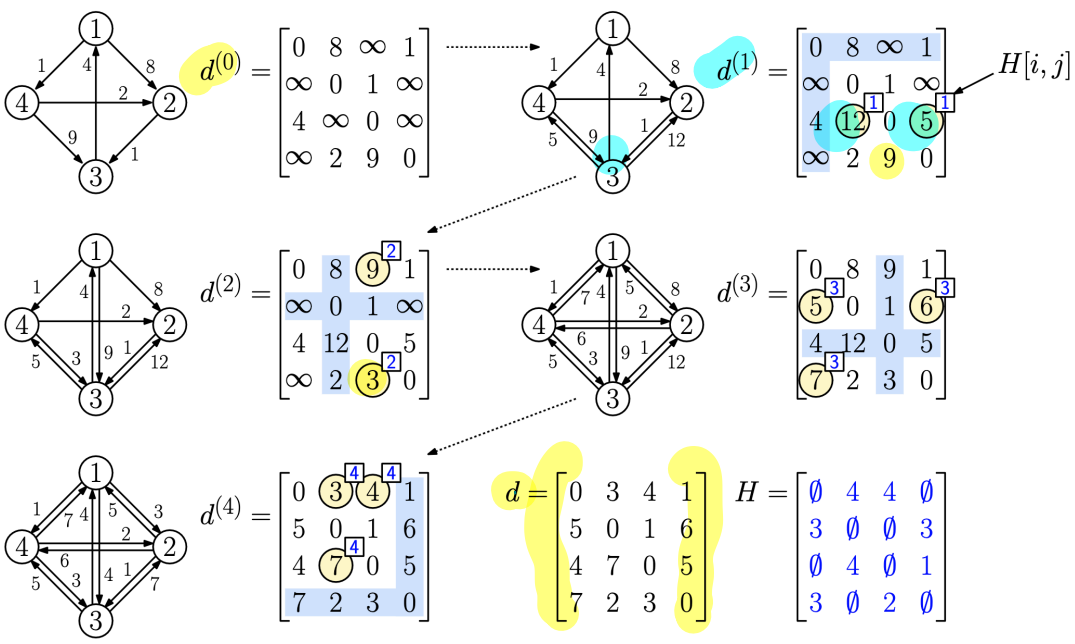
$d^{(k)}(i,j)$ needs $d^{(k-1)}(i,j)$

Actually only 1 is needed!

$d^{(k-1)}(i,k)$

$d^{(k-1)}(k,j)$

not updated in round k



Extract the shortest path:

```
Floyd-Warshall(w)
    for i ← 1 to n
        for j ← 1 to n
            d[i,j] ← w[i,j]
            h[i,j] ← ∅
    for k ← 1 to n.
        for i ← 1 to n.
            for j ← 1 to n.
                newCost ← d[i,k] + d[k,j]
                if newCost ≤ d[i,j]
                    d[i,j] ← newCost
                    h[i,j] ← k,
    return d[ ], h[ ].
```

if "≤", then you will get infinite recursion on get-path.

```
get-path(i,j)
    if h(i,j) = ∅, return (i,j)
    else mid ← h[i,j]
        return get-path(i, mid) ⊕
               get-path(mid, j)
```

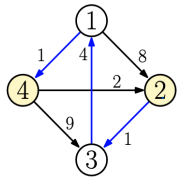
⊕: list concatenation

$$H = \begin{bmatrix} \emptyset & 4 & 4 & \emptyset \\ 3 & \emptyset & \emptyset & 3 \\ \emptyset & 4 & \emptyset & 1 \\ 3 & \emptyset & 2 & \emptyset \end{bmatrix}$$

get-path$(2,4)$: get-path$(2,3)$ $\oplus$ get-path$(3,4)$

$(2,3) \oplus$ (get-path$(3,1)$ $\oplus$ get-path$(1,4)$)

$(2,3) \oplus (3,1) \oplus (1,4)$



Question: Why get-path can always terminate?

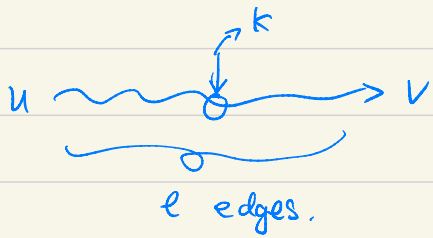
Another way:

```
Floyd-Warshall(w)
    for i ← 1 to n
        for j ← 1 to n
            d[i,j] ← w[i,j]
            next[i,j] ← j
    for k ← 1 to n.
        for i ← 1 to n.
            for j ← 1 to n.
                newCost ← d[i,k] + d[k,j]
                if newCost < d[i,j]
                    d[i,j] ← newCost
                    next[i,j] ← next[i,k]
    return d[ ], next[ ]
```

get-path(i,j): $v \leftarrow i$,
while $v \neq j$: $v = next[v,j]$, print(v).

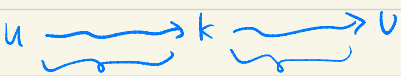
- By # edges in the path?

$d^{(\ell)}(u, v)$ = shortest path $u \to v$ w/ at most ℓ edges.



mid-point can be any $k \in V$, $(k \neq u, v)$

Case #k, mid-point is k,



$d^{(\ell/2)}(u, k) + d^{(\ell/2)}(k, v)$

$$d^{(\ell)}(i,j) = \begin{cases} w(i,j) & \ell = 1 \\ \min_{k \in V} \left(d^{(\ell/2)}(i,k) + d^{(\ell/2)}(k,j) \right), & \ell \geq 2 \end{cases}$$

$d^{(1)} \to d^{(2)} \to d^{(4)} \to d^{(8)} \to \cdots \to d^{(n)}$ total $O(\log n)$ rounds.

each round: $d^{(\ell)} \to d^{(2\ell)}$ $O(n^3)$ running time.

Overall time $O(n^3 \log n)$

```
for i ← 1 to n
  for j ← 1 to n
    for k ← 1 to n
```