CMSC351-2023-11-10

Friday, November 10, 2023 8:16 AM

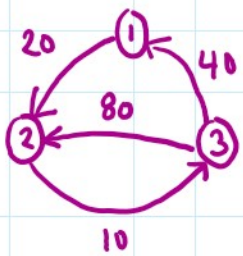
FLOYD'S ALGORITHM

REMINDER!
EXAM NEXT FRIDAY!

① Intro: Floyd's Algorithm finds shortest paths between all pairs of vertices in a directed, weighted, simple, and connected graph.

note: Non-positive weights are allowed!

② A Dynamic Example! Consider:



Adj matrix (1-indexed!)

$$d = \begin{bmatrix} 0 & 20 & \infty \\ \infty & 0 & 10 \\ 40 & 80 & 0 \end{bmatrix}$$

d[i,j] = weight of edge from i → j
or ∞ if no edge
or 0 if i=j

another way to think of d:
d[i,j] = length of the shortest path from i → j allowing no intermediate vertices.

Now: Let's allow 1 to be an intermediate vertex.
now the SP from 3 → 2 changes to 60 b/c we go by way of 1.
More specifically d[3,1] + d[1,2] < d[3,2]
We'll update d. Note - nothing else changes!

Now:

$$d = \begin{bmatrix} 0 & 20 & \infty \\ \infty & 0 & 10 \\ 40 & 60 & 0 \end{bmatrix}$$

think: this d yields lengths of SP allowing 1 as an int. vertex!
"Pass by 1 matrix"

Now let's also allow 2 to be an int. vertex!
now the SP from 1 → 3 appears! We had d[1,3] = ∞ b/c no path!
Spec. d[1,2] + d[2,3] < d[1,3]
We'll update d. Nothing else changes!

Now

$$d = \begin{bmatrix} 0 & 20 & 30 \\ \infty & 0 & 10 \end{bmatrix}$$

Now
$$d = \begin{bmatrix} 0 & 20 & 30 \\ \infty & 0 & 10 \\ 40 & 60 & 0 \end{bmatrix}$$

think: this d yields lengths of SP allowing 1, 2 as int. vert.
"Pass by (1 and) 2 matrix"

Now let's also allow 3 to be an int. vertex!
now the SP from 2→1 appears! we had $d[2,1] = \infty$ b/c no path!
Spec. $d[2,3] + d[3,1] < d[2,1]$
we'll update d. Nothing else changes!

Now
$$d = \begin{bmatrix} 0 & 20 & 30 \\ 50 & 0 & 10 \\ 40 & 60 & 0 \end{bmatrix}$$

think: this d yields lengths of SP allowing 1, 2, 3 as int. vert
"Pass by (1 and 2 and) 3 matrix"

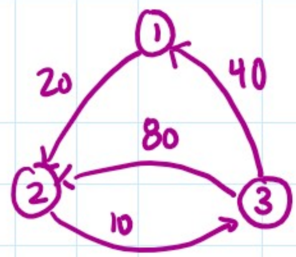
BUT we're now allowing all vertizes so this d yields lengths of all SP!

NOTE: Our update ineq. calculations come from observations about d as it is being updated.
That's why it's dynamic programming!

③ Finding the Path

note: d only gives us lengths, not paths! :(
we'll construct a new matrix which will also get updated.
First we'll define matrix p such that:

$p[u,v] = u$ for every edge $u \to v$
$p[v,v] = v$ for every vertex v
otherwise null.



for our graph we have
$$p = \begin{bmatrix} 1 & 1 & \text{null} \\ \text{null} & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

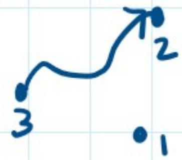
$[3 \quad 3 \quad 3]$

think: $p[i,j]$ = predecessor of j in the SP from $i \to j$
allowing no intermediate vertices.

eg. $p[1,2]=1$ b/c the SP from 1 to 2 w/ no int. vert. is just $1 \to 2$
so 2's pred. is 1.

eg. $p[1,1]=1$ b/c the SP from 1 to 1 w/ no int. vert. is just 1
so 1's pred is 1. (a bit goofy!)

Next: if we allow 1 as an int vertex
the SP from 3 to 2 changes b/c it goes via 1.
abstractly we had:



we need to change
the predecessor of 2!

now:

what we do is set:

$p[3,2] = p[1,2]$

b/c our shortest path from 3 to 2
now goes via 1,
so 2's pred from 3 ought to be
changed to 2's pred. from 1.

now we have

$$p = \begin{bmatrix} 1 & 1 & null \\ null & 2 & 2 \\ 3 & 1 & 3 \end{bmatrix}$$

next: let's allow 1 and 2 as int. vert!
a path 1 to 3 appears! It's $1 \to 2 \to 3$
we had $p[1,3]$ = null b/c no path.
now we set $p[1,3] = p[2,3]$ b/c we go to 2 then to 3.

now

$$p = \begin{bmatrix} 1 & 1 & 2 \\ null & 2 & 2 \\ 3 & 1 & 3 \end{bmatrix}$$

next let's allow 1, 2, 3 as int vert!
a path 2 to 1 appears! It's $2 \to 3 \to 1$
we had $p[2,1]$ = null b/c no path.

a path 2 to 1 appears: it's $2 \to 3 \to 1$

we had $p[2,1]$ = null b/c no path.

now we set $p[2,1] = p[3,1]$ b/c we go to 2 then to 3.

thus

$$p = \begin{bmatrix} 1 & 1 & 2 \\ 3 & 2 & 2 \\ 3 & 1 & 3 \end{bmatrix}$$ ← final p!

monday: how we use p to get actual paths!

④ Pseudocode! look @ notes!

first two for loops initialize P.

They are $\Theta(V^2)$ and $\Theta(V)$ respectively.

The huge loop: k is "pass by".

ie. k=1 gets us d when 1 is allowed.

k=2 1,2 are allowed

⋮

the inner loops update d, p if appropriate.

Those loops are $\Theta(V^3)$

total: $\Theta(V^3)$. Seems slow!