CMSC351-2023-11-06

Monday, November 6, 2023 8:14 AM

REMINDER! EXAM 3 NEXT FRIDAY!

SHORTEST PATH ALGORITHM : NON-TARGET VERSION

- If we make some simple modifications to the SPA pseudocode...

```
function shortestpath(G,s,t)
    dist = distance array of size V full of inf
    pred = predecessor array of size V full of NULL
    Q = empty queue
    dist[s] = 0
    Q.push(s)
    while Q is nonempty
        x = Q.pop
        for each infinity vertex y adjacent to x
            dist[y] = dist[x] + 1
            pred[y] = x
            if y == t
                return(pred)        } delete
            end
            Q.push(y)
        end
    end
end                                 } return pred
```

Practical Result

pred will contain all predecessor info nec. to construct shortest paths from s to all other vertices.

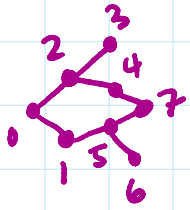
Time Complexity

Still $\theta(V+E)$.

BREADTH-FIRST TRAVERSE!

① **Idea:** Given a graph and a starting vertex s
we wish to visit all vertices but visit those closest to s first.

ex



An ex of a BFT starting at 0 would be:

0, 1, 2, 3, 4, 5, 6, 7

0 | 1 | 2 | 3

edges away

② **Note:** The SPA essentially does this process.
We simply need to make a few changes:

- remove dist list, replace by a visited list of booleans which store if we've visited a vertex or not. Don't visit twice!
- throw out pred list (don't need it!)
- introduce a list vorder which stores the order

- introduce a list vorder which stores the order in which we visit (traverse) the vertices.

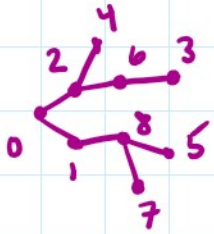
③ Pseudocode:

```
function bft(G,s)
    QUEUE = [s]
    VISITED = list of FALSE of length V
    VISITED[s] = TRUE
    VORDER = [s]
    while QUEUE is not empty
        x = QUEUE.dequeue
        for all y adjacent to x
            if VISITED[y] == FALSE
                QUEUE.enqueue(y)
                VISITED[y] = TRUE
                VORDER.append(y)
            end
        end
    end
    return(VORDER)
end
```

This is essentially the same structure as SPA. Thus the time complexity is still $\Theta(V+E)$. (worst-case!)

DEPTH-FIRST TRAVERSE

① Intro: The idea is now, from s, to go as far as possible, then when we can go no further, we back up to a junction where we can go deep again.

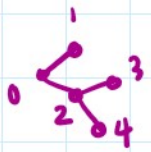
ex 

an ex. of a DFT starting at 0 would be:

0, 1, 8, 5, 7, 2, 4, 6, 3

back up! back way up! back up

the natural way to do this is a stack.

ex 

start at 0

S = [0] to start

pop 0, push 1,2: S = [1,2]

pop 2, push 3,4: S = [1,3,4]

pop 4, no push: S = [1,3]

pop 3, no push: S = [1]

pop 4, no push: S = [1,3]
pop 3, no push: S = [1]
pop 1, done!
↳ the pop order is a DFT!

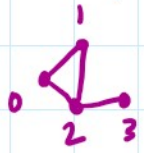
However most graphs are not trees.
What impact does this have?
We'll say: Keep track of whether we've visited a vertex and if so, don't visit it again!
Q: Should we not have things on the stack multiple times?
For now, let's not care...

② Pseudocode

```
VORDER = []
VISITED = list of length V full of FALSE
STACK   = [s]
while STACK is not empty:
    x = STACK.pop()
    if VISITED[x] == FALSE:
        VISITED[x] = TRUE
        VORDER.append(x)
    end if
    for all nodes y adjacent to x:
        if VISITED[y] == FALSE:
            STACK.append(y)
        end if
    end for
end while
```

This is perfectly good!

Ex



Start @ 0

start S = [0]
pop 0: set VISITED[0] = T
VORDER = [0]
append 1,2: S = [1,2]
pop 2: set VISITED[2] = T
VORDER = [0,2]
append 1,3: S = [1,1,3]
pop 3: set VISITED[3] = T
VORDER = [0,2,3]
append nothing: S = [1,1]
pop 1: set VISITED[1] = T
VORDER = [0,2,3,1]
append nothing: S = [1]

append nothing: S=[1]

pop 1: been visited
nothing happens. S=[].

③ Pseudocode Time Complexity?

Some sources say $\Theta(V+E)$. They lie! ∴ Why?

ex Sps a graph has V vertices, all conn. to one another.
Start at some vertex. pop it. V-1 get pushed (all non-visited!)
after next pop we push V-2 vertices on (all non-visited!)
Keep going. # of vert. pushed onto the stack:

$$1+(V-1)+(V-2)+\dots+(1)$$

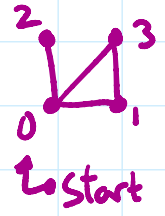
$$= 1+\sum_{i=1}^{V-1} i = 1+\frac{V(V-1)}{2} = \Theta(V^2)$$

So in the code the body of while, ignoring for loop, iterates $\Theta(V^2)$ times!
The body of the for loop, overall, iterates 2E times (as SPA, BFT).
Thus $\Theta(V^2+E)$.

④ Fix! we need to ensure things don't go on the stack more than once!
we still won't put them on the stack if they've been visited,
but in addition here are two ideas:
(a) Store if something has been on the stack. if so, don't push again.
(b) If it is already on the stack, remove earlier occurrence!

(a) works ∴ but fails!

ex



S=[0]
pop 0, it's now visited, VORDER=[0]
push 3,2,1 so S=[3,2,1]
pop 1, it's now visited. VORDER=[0,1]
push nothing. so S=[3,2]
pop 2. it's now visited. VORDER=[0,1,2]

push nothing. so S = [3,2]

pop 2, it's now visited. VORDER: [0,1,2]

push nothing, so S = [3]

pop 3. it's now visited. VORDER = [0,1,2,3]

oh crap! not a DFT! we should do 3 after 1!

okay, how do we do (b)?