07 BFS/Dijkstra

Tuesday, January 28, 2025 13:49

- BFS ←
- Dijkstra's Algorithm
- MST Prim's Algo

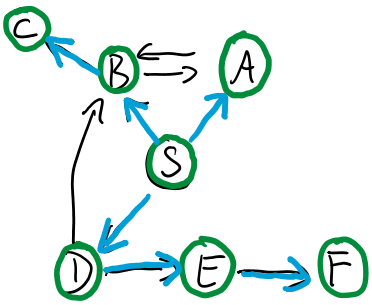
```
Search(v)
    queue.add(v)
    T = ∅
    while queue not empty
        v,e = queue.pop()
        if visited[v]
            continue
        T.add(e)
        visited[v] = true
        for neighbour u of v
            queue.add(u, (v,u))
```

LIFO queue → DFS
(last in first out, stack)

FIFO queue → BFS
(first in first out)

breadth first search

later:
priority queue → many different algorithms



BFS (FIFO)

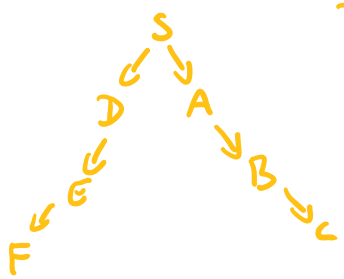
queue | dist 1 | dist 2 | dist 3

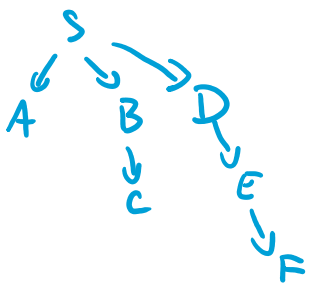
$ | A | B | D | B | A | C | B | C | F
SA SB SD AB BA BC DB DE EF

dist 0

BFS tree

DFS tree



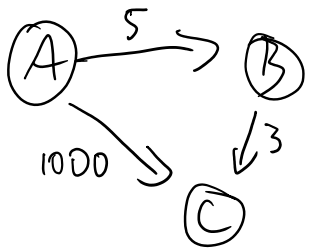


Observation: For all $d \in \mathbb{N}$
BFS visits vertices of distance d before visiting vertices of distance $d+1$

BFS tree is also called shortest path tree rooted at s

rooted at s
because it consists
of the shortest paths
from s to all other
vertices.

BFS gives shortest paths
where length is measured by # of steps.



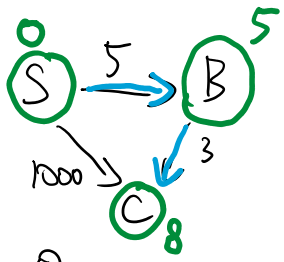
In weighted graphs, each edge has a weight
and the length of a path is the sum
of edge weights.

In weighted graphs BFS fails to find the shortest path.

To fix this issue, we want to keep the queue sorted by distance.

```
Dijkstra(v)
    queue.add(v, ∅, 0)
    while queue is not empty
        v, p, d = queue.pop()   // remove entry with smallest priority
        if visited[v]
            continue
        visited[v] = true
        distance[v] = d
        for neighbor u of v
            queue.add(u, v, distance[v] + W_vu)
```

priority



Queue

vertex	parent	distance
~~S~~		~~0~~
~~B~~	~~S~~	~~5~~
C	S	1000
~~C~~	~~B~~	~~8~~

$O(|E| \cdot \log |E|)$

↑ # insertions/pops

↖ $\log|E|$ = time per queue operations

$O(|E| + |V| \log |V|)$ via Fibonacci heaps

$O(|E| + |V| \log |V|)$ via Fibonacci heaps

2:52

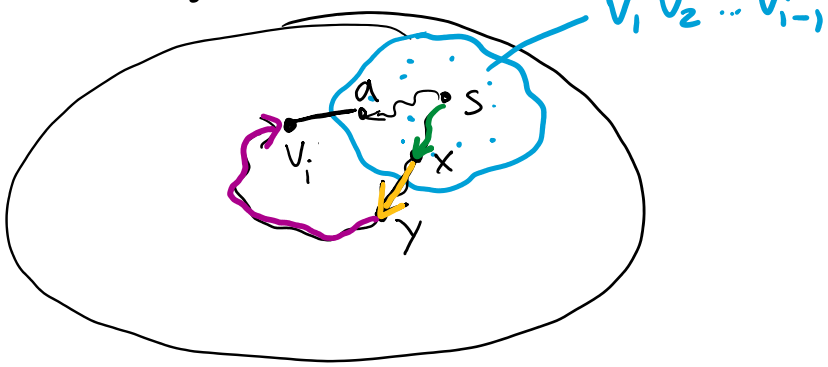
Thm: distance[v] is correct (length of shortest path from start vertex to v) $\forall v \in V$ if all edge weights are not negative.

Proof by contradiction. Assume there is some vertex v distance[v] is wrong

Let $v_1\ v_2\ v_3 \dots v_n$ be the vertices in order they are visited by Dijkstra

Let v_i be the first vertex where distance$[v_i]$ is wrong

distance$[v_j]$ for $j<i$ is correct.



distance$[v_i]$ = distance$[a] + w_{av_i}$

> distance from s to v_i

= distance$[x] + w_{xy}$ + distance from y to v_i

≥ 0

$\geq$ distance$[x] + w_{xy}$

$\Rightarrow$ distance$[a] + w_{av_i} >$ distance$[x] + w_{xy}$

$\Rightarrow$ y should have been removed from the queue before v_i

but by assumption v_i was removed before y since y is not part of $v_1\ v_2 \dots v_{i-1}$

$\Rightarrow$ contradiction.

Prim

~~Dijkstra~~ (v)

queue.add$(v, \emptyset, 0)$

$T = \emptyset$ t by

while queue is n emp

| v, p, d = queue.pop() // remove entry with smallest priority

| if visited[v]

```
    v, p, _ = pq.pop()
    if visited[v]
        continue
    visited[v] = true
    T.add((p, v))
    for neighbor u of v
        queue.add(u, v, ~~distance[v] +~~ w_vu)
return T
```

priority

T is a minimum weight spanning tree.

of all edges in the tree

Def: a spanning tree where the sum of edge weights is as small as possible.

$O(|E| \log |E|)$