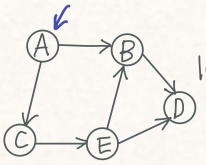
Breadth First Search

1. Algorithm Design

```
Def BFS (G, s)
    for all vertices u in V
        dist[u] = ∞        # Initialize dist to ∞
    dist[s] = 0            # dist from source is 0.
    Q = {s}                # Queue to keep track of nodes
    while Q is not empty :
        u = eject(Q)       <—— # Assertion 1*
        for all neighbors, v, of u
            if dist[v] == ∞
                inject(Q, v)
                dist(v) = dist(u) + 1
```

Source



Dist	A	B	C	D	E	Queue {}	ejected	injected
init:	0	∞	∞	∞	∞	A	A	B, C
1st while:	0	~~∞~~ 1	~~∞~~ 1	∞	∞	~~A~~ B, C	B	D
	0	1	1	∞	∞	~~B~~, C	C	E
2nd :	0	1	1	2	∞	~~C~~, D		
	0	1	1	2	2	~~D~~, ~~E~~	D, E	

2. Use Cases

To find shortest paths, DFS is not helpful

↳ path length can differ depending on which path to explore first.

⇒ Explore "shallow" nodes before "deeper" ones.
Keep track of nodes to explore w/ FIFO queue.

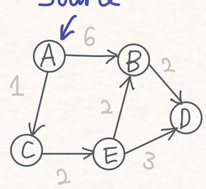
* Dijkstra's Shortest Path w/ Edge Weight

Given edge length $\ell(u,v)$,
Keep track of nodes to explore using "Priority Queue"

↳ instead of dist(u) = dist(v) + 1,
use "Relaxation"

```
Def DSP (G, s)
    for all vertices u in V
        dist[u] = ∞            # Initialize dist to ∞
    dist[s] = 0                # dist from source is 0.
    H = MakeHeap(V)            # Min Heap to keep track of nodes
    while H is not empty :  ........... O(n)
        u = DeleteMin(H)    ........... O(log n)
        for all neighbors, v, of u
            if dist[v] > dist[u] + ℓ[u,v]  ....... O(e)
                dist[v] = dist[u] + ℓ[u,v]
                DecreaseKey(H, v)  ........... O(log n)
```

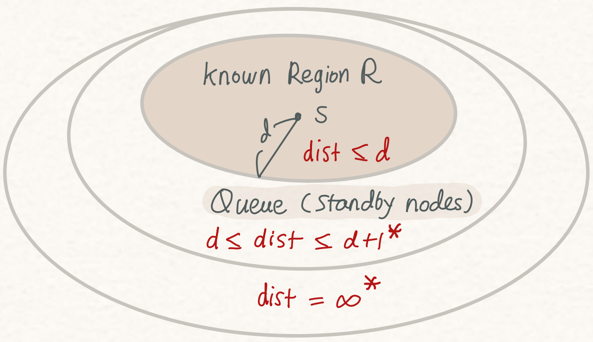
Source



Dist	A	B	C	D	E	Heap {}	Removed	Added	Dist
init:	0	∞	∞	∞	∞	A	A	C	$\ell(A,C) = 1$
1st while:	0	6	1	∞	∞	C	C	E	$\ell(C,E) = 2$
	0	6	1	∞	3	E	E	B	$\ell(E,B) = 2$
	0	5	1	∞	3	B	B	D	$\ell(E,D) = 3$
	0	5	1	6	3	D			

⇒ Time Complexity : $O(n \log n + e \log n)$

3. Properties of BFS



At initialization, dist = ∞ except for dist(source) = 0

With the (standby) queue, closest nodes from s are pulled out.

Once the distance is updated, it is final & correct value.

⇒ Time Complexity : $O(n + e)$