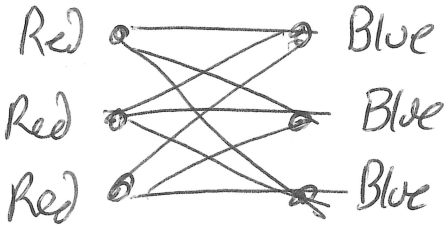
COP 3503 2/29/2024

0. Katti's Note
1. Extra Notes To Add (Typed)
   - Perm ↔ Rank
   - Fast I/O in Java
2. Two Color (Katti's #4: Breaking Bad)
3. Dijkstra's Alg.
4. I have to leave @ end of class!!!

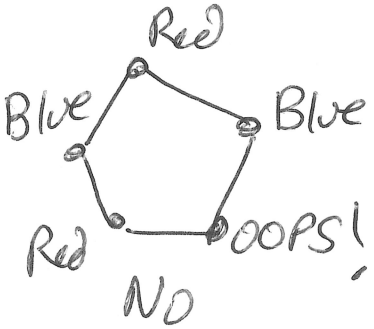
Two Color Problem

Given undirected, unweighted graph
Can you assign each vertex one of two colors such that no two vertices connected by an edge share the same color?

Solve in $O(V+E)$ time adapt DFS or BFS.

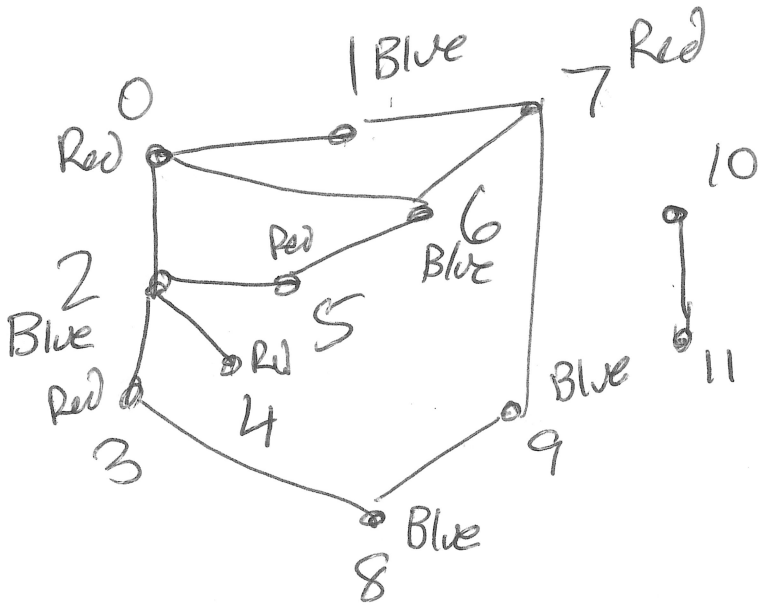


yes



NO

If a graph can be 2 colored we call it bipartite.



As you run DFS/BFS, always color a node opposite of the previous node you come from. While you loop through all the edges from the newly colored node, if anything is visited already double check that its color is opposite to the newly colored node if not, graph isn't 2 colorable

BFS(0)

Q: ~~1,2,6~~ → ~~,7,3,4,5~~ , 9, 8

0 Red check
0,5,7 Red ✓

Note
Check 0 Red ✓

When we dequeue look at its neighbors & is Blue return false

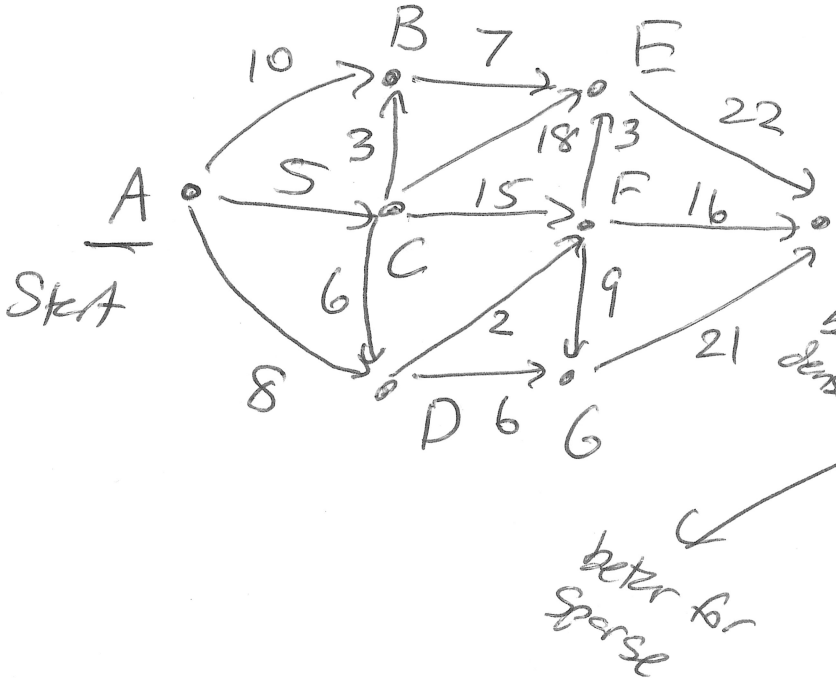
```
BFS
for (int x : g[cur]){
  if (dist[x] == -1) {

  ~~~

  else
    // Double check color[x] != color[cur]
}
```

Single Source Shortest Path in Directed Weighted Graph

No neg edge weights



Dijkstra's Alg.

$O(V^2)$ inefficient

better dense

$O(E \lg V)$ efficient

If $E \sim V^2$, then it's just better to do this

better for sparse

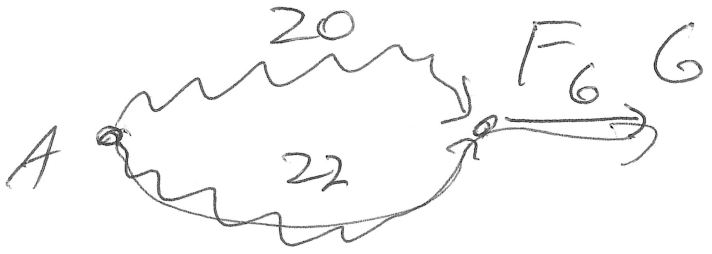
1. Create a set $S = \emptyset$, set of vertices ~~from~~ for which we have the correct shortest path from source vertex
2. Dist ~~estim~~ estimates: dist[source] = 0
   dist[rest] = ∞
3. Priority Queue of dist estimates:
   {source, 0}, {B, ∞}, {C, ∞}
4. While (S != V) { → set of vertices
   estimate cur = pq.poll(); // smallest item in PQ.
   if (cur.v in S) continue;
   S.add(cur.v); // what's in dist is correct.
   for (Edge e: g[cur.v]) {
   if(cur.w + e.w < dist[e.v]) {
   dist[e.v] = cur.w + e.w;

If traveling this edge improves our estimate

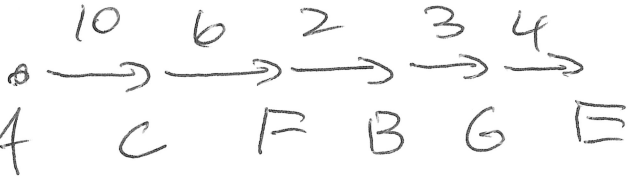
```
        pq.add({e.v, dist[e.v]})
    }
}
```

Intuitive Proof

All shortest distances build off of shortest distance.



Claim shortest $AG = 28$ but exist $AF = 20$ and edge $FG = 6$ contradiction

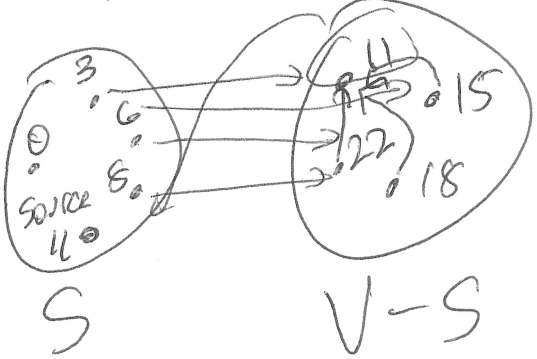


all $AF = 16$ $CF = 6$
$AB = 18$ $CB = 8$
$AG = 21$ $CG = 11$
$CE = 15$
etc.

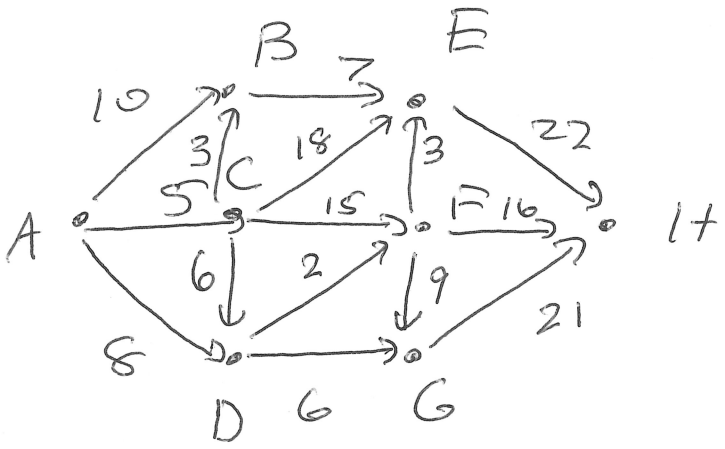
All subpaths are also shortest paths!

Proof assume contrary that at some point in time Dijkstra's adds a vertex into S but its distance estimate isn't the shortest

Add in II



Only edges not considered are from vertices in V-S



S	A	B	C	D	E	F	G	H
A	0	10	5	8	∞	∞	∞	∞
C	0	8	5	8	23	20	∞	∞
B	0	8	5	8	15	20	∞	∞
D	0	8	5	8	15	10	14	∞
F	0	8	5	8	13	10	14	26
E	0	8	5	8	13	10	14	26
G	0	8	5	8	13	10	14	26

H & end w/o loop edges