22.2 BREADTH FIRST SEARCH.

GIVEN $G = (V, E)$ (DIRECTED OR UNDIRECTED)
AND A DISTINGUISHED SOURCE VERTEX $s \in V$,
BFS DOES SEVERAL THINGS SIMULTANEOUSLY:

(1) SYSTEMATICALLY 'DISCOVERS' EVERY VERTEX IN G REACHABLE FROM s.

(2) COMPUTES $\delta(s, u)$ FOR EVERY $u \in V$.

(3) CREATES A BFS TREE, WHICH IS A SUBTREE OF G WHOSE VERTEX SET CONSISTS OF THOSE VERTICES REACHABLE FROM s. THE UNIQUE PATH IN THIS TREE FROM s TO u IS A SHORTEST s-u PATH.

TO ACCOMPLISH THESE TASKS BFS REQUIRES THAT VERTICES POSSES THREE ATTRIBUTES (i.e. FIELDS).

- Color[u] (white, grey, black) KEEPS TRACK OF DISCOVERED VS. UNDISCOVERED VERTICES.
- $d[u]$ STORES $\delta(s, u)$.
- $P[u]$ STORES u's PARENT IN THE BFS TREE. NOTE $P[s]$ is nil SINCE s HAS NO PARENT.

BFS ASSUMES G IS REPRESENTED BY AN ARRAY OF ADJACENCY LISTS Adj[...]. IT ALSO USES AN AUXILIARY FIFO QUEUE Q.

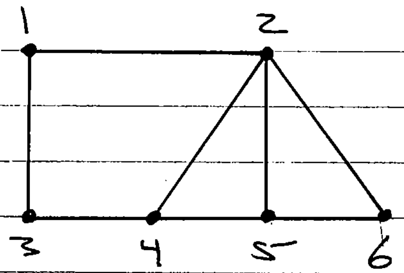
BFS (G, s)

```
1.)  for All u ∈ V - {s}
2.)       color[u] ← white
3.)       d[u] ← ∞
4.)       P[u] ← nil
5.)  Color[s] ← grey
6.)  d[s] ← 0
7.)  P[s] ← nil
8.)  Q ← ∅
9.)  Enqueue(Q, s)
10.) while Q ≠ ∅
11.)      u ← Dequeue(Q)
12.)      for All v ∈ Adj[u]
13.)           if Color[v] = white
14.)                color[v] ← grey
15.)                d[v] ← d[u] + 1
16.)                P[v] ← u
17.)                Enqueue(Q, v)
18.)      Color[u] ← black
```

- lines 1-7 initialize each vertex.
- lines 8-9 initialize FIFO Queue Q
- lines 10-18 explore G
  - white vertices are undiscovered
  - grey vertices are on the frontier
  - black vertices are complete.

Ex.

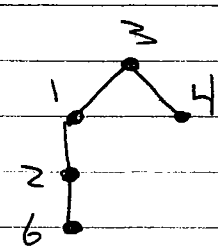
G



CALL BFS(G, 3)

	color	d	p
1 : 2 3	~~w~~ ~~g~~ b	~~∞~~ 1	~~n~~ 3
2 : 1 4 5 6	~~w~~ ~~g~~ b	~~∞~~ 2	~~n~~ 1
3 : 1 4	~~g~~ b	0	n
4 : 2 3 5	~~w~~ ~~g~~ b	~~∞~~ 1	~~n~~ 3
5 : 2 4 6	~~w~~ ~~g~~ b	~~∞~~ 2	~~n~~ 4
6 : 2 5	~~w~~ ~~g~~ b	~~∞~~ 3	~~n~~ 2

Q: ~~3~~ ~~1~~ ~~4~~ ~~2~~ ~~5~~ 6 BFS TREE:



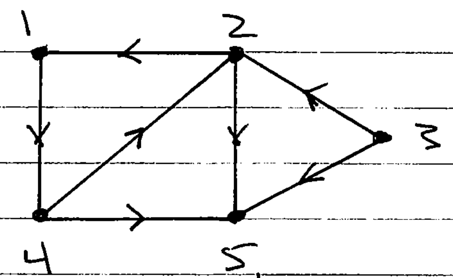
NOTE ALL VERTICES REACHABLE FROM S PASS THROUGH Q, WHICH ALWAYS CONTAINS THE grey VERTICES.

IF G IS UNDIRECTED AND DISCONNECTED, ALL VERTICES IN THE CONNECTED COMPONENT CONTAINING S will BE COLORED black, AND ALL OTHER VERTICES will BE COLORED white.

BFS AS WRITTEN IS VALID ON DIRECTED GRAPHS. ALL VERTICES REACHABLE FROM S will BE COLORED black, OTHERS white.

Ex.

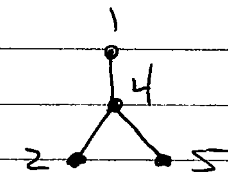
G



call BFS(G, 1)

	Color	d	p
1 : 4	~~g~~ b	0	n
2 : 1 5	~~w~~ ~~g~~ b	~~∞~~ 2	~~n~~ 4
3 : 2 5	w	∞	n
4 : 2 5	~~w~~ ~~g~~ b	~~∞~~ 1	~~n~~ 1
5 :	~~w~~ ~~g~~ b	~~∞~~ 2	~~n~~ 4

Q: ~~1~~ ~~4~~ ~~2~~ ~~5~~ BFS TREE:



THE BFS TREE (PREDECESSOR SUBGRAPH) is DEFINED FORMALLY BY $T = (V_p, E_p)$ WHERE

$$V_p = \{ \; \in V \mid p[u] \neq nil \} \cup \{s\}$$

$$E_p = \{ (p[u], u) \in E \mid u \in V_p - \{s\} \}$$

NOTE THAT THE BFS TREE CAN DEPEND ON THE ORDER IN WHICH ADJACENCY LISTS ARE PROCESSED.

EXERCISE ALTER THE ORDER OF THE ADJACENCY LISTS IN THE LAST TWO EXAMPLES AND RE-RUN BFS.

EXERCISE: RE-RUN THE PREVIOUS EXAMPLES WITH DIFFERENT SOURCES.

EXERCISE RUN BFS ON OTHER EXAMPLES: DIRECTED, UNDIRECTED, CONNECTED, STRONGLY CONNECTED, ETC..

ONCE BFS IS RUN, SHORTEST PATHS FROM s TO ANY OTHER $u \in V$ CAN BE PRINTED BY RUNNING THE FOLLOWING ALGORITHM.

PrintPath(G, s, u)

```
1.) if u = s
2.)     Print s
3.) else if P[u] = nil
4.)     Print 'no Path FROM' s 'to' u 'EXISTS'
5.) else
6.)     PrintPath(G, s, P[u])
7.)     Print u
```

READ: P. 534-538, THM 22.5 & PRECEDING LEMMAS FOR A PROOF OF CORRECTNESS OF BFS.

RUN TIME OF BFS

SINCE NO VERTEX IS EVER whitened THE TEST ON line 13 GUARANTEES THAT EACH VERTEX PASSES THROUGH THE QUEUE AT MOST ONCE. EACH QUEUE OPERATION (Enqueue, Dequeue) TAKES TIME $\Theta(1)$. THUS QUEUE OPERATIONS TAKE TIME $O(|V|)$ IN TOTAL.

SINCE EACH VERTEX IS AT THE HEAD OF THE QUEUE AT MOST ONCE, EACH ADJACENCY list IS SCANNED AT MOST ONCE. THE SUM OF THE LENGTHS OF All ADJACENCY lists IS THE SUM OF THE VERTEX DEGREES (IF G IS UNDIRECTED) WHICH IS $2|E|$ BY THE HANDSHAKE lEMMA. THUS $O(|E|)$ TIME IS SPENT SCANNING ADJACENCY lists.

INITIALIZATION TAKES TIME $\Theta(|V|)$.

THUS THE RUN TIME OF BFS IS $O(|V|+|E|)$, i.e. linear IN THE SIZE OF THE ADJACENCY list REPRESENTATION.