Idea

- **Given**: connected graph $G = (V, E)$, startnode $s \in V$
  Idea: Starting from s, search in all directions uniformly
  - visit vertices u with distance $d_G(s, u) = 1$, then with $d_G(s, u) = 2$, and so on.
  - traverse a *BFS-tree* T with root s: branches of T consist of shortest paths to s.
  - to build T, assign the following values to each vertex u:

$pre(u)$	predecessor of u in the BFS-Tree T
$state(u)$	new (unvisited), labelled (visited), saturated (all neighbours visited)

  - store visited unsaturated vertices in a **queue** Q
  - initially: Q empty, $pre(u)$ not set, $state(u)$ = new

Pseudo-Code

- ```
  BFS(G, s)     /* G given as adjacency list F */
  for all u ∈ V
      state(u)=new
  state(s)=labelled
  num(s)=1; i = 2; pre(s)=nil
  PUT(Q, s)
  while Q ≠ 0
      GET(Q, u)
      for all v ∈ F[u]     /* testing all neighbors of u */
          if state(v)==new
              state(v)=labelled; num(v)=i
              pre(v)=u; PUT(Q, v); i = i + 1
      state(u)=saturated
  ```

Properties

- time complexity
  ```

  - Each vertex is inserted into Q exactly once:
    $\Theta(1)$ time per vertex $\Rightarrow \Theta(n)$ for all vertices
  - After removal of a vertex u from Q, the algorithm goes through the adjacency-list of u:
    $\Theta(\text{degree}(u))$ time for u $\Rightarrow$ How much for all vertices?

    Every edge contributes to exactly two lists
    $\Rightarrow \sum_{u \in V} \text{degree}(u) = 2m$ $\Rightarrow \Theta(m)$ for all vertices

    $\Rightarrow$ The whole algorithm: $\Theta(n+m)$ time in total
- space complexity
  - $\Theta(n)$ for Q, +graph $\Rightarrow \Theta(n+m)$ space in total

Distance in Unweighted Graph

- All the shortest paths from some vertex u to s are coded in the BFS-tree T via the pre-pointers.
- distances to s can be easily computed during BFS:
  - $d_G(s,s) = 0$
  - $d_G(s,u) = d_G(s, pre(u)) + 1$

- $\Rightarrow$ Running BFS for n times (once for each vertex), one can compute the distances between any $u, v \in V$

  $\Rightarrow$ The *distance-matrix* of G can be computed in $\Theta(n \cdot m)$ time and $\Theta(n^2)$ space if G is connected.

Recognizing Bipartite Graphs

- [[Graphs]] are bipartite if 2-colorable
- combine this with [[Breadth-First Search]]

Algorithm:

- Choose arbitrary vertex s and color it blue
- Traverse G in BFS-Order, starting from s
- For each vertex u that is removed from the queue Q:
  - u is colored, w.l.o.g. say red
  - check for all colored neighbors of u that they are blue. If no: return false
  - color all uncolored neighbors of u in blue
- After processing all vertices: return true.

[[Shortest Path Algorithms]]