# * Breadth First Search

- can run on directed/undirected graph and we ignore any weights.
- we will assume an underline adjancay list representation.
- we start from a vertex, and explore them in the order of the number of links they are away ex: i.e all vertices that are 1 link away, then 2 etc until all reachable are finished.

## * Simple Implementation:

```
BFS (startVertex)
    ResetGraph()
    startVertex.discovered = true
    Q = new Queue()
    Q.enqueue(startVertex)
    while(not Q.isEmpty)
        u = Q.dequeue()
        for each v s.t u → v
            if(not v.discovered)
                v.discovered = True
                Q.enqueue(v)
```

```
ResetGraph()
    for v ∈ V
        v.discovered = False
```

- what it does is marks all the reachable vertex true.
- for same graph we can get a different BFS tree depending on what node you pick as a start vertex.

* Detailed Implementation:

```
BFS (startVertex)
    ResetGraph()
    startVertex.discovered = True
    startVertex.dist = 0
    Q = new Queue()
    Q.enqueue(startVertex)
    while(not Q.isEmpty)
        u = Q.dequeue
        for each v s.t u→v
            if(not v.discovered)
                v.discovered = True
                v.dist = u.dist + 1
                v.π = u
                Q.enqueue(v)
```

```
ResetGraph()
    for v ∈ V
        v.discovered = False
        v.dist = ∞
        v.π = nil
```

* BFS gives the shortest path

```
FindPath(startVertex, v)
    if(v == startVertex)
        return new List().add(v)
    if(v.π == nil) return nil.

    return FindPath(startVertex, v.π).add(v)
```

union of all the paths gives the BFS tree.
the more the connected the graph is, more bushy its BFS trees tends to look

* Another approach to discovered/not discovered is common vertex colors → white for undiscovered, grey → discovered but not yet finished (in the queue), black → after its finished.

* Analysis

- Reset graph takes $\Theta(|V|)$.
- Let $V', E'$ be the vertices and edges reachable from the start vertex.
  - $\Theta(|V'|)$ → vertices discovered, enqueued and dequed
  - $\Theta(|E'|)$ → edges explored by dequeued vertices
- Total runtime $\Theta(|V|+|E'|)$ or $O(|V|+|E|)$

* What does it achieve?

- marks as discovered
- calculates min dist from startVertex
- calculates path/parent.

for all reachable vertices. (discoverable)

* Correctness Argument :

- Set of vertices in Q never has more than 2 distinct distances, one apart. values enter and exit Q in non-decreasing distance order. (Induction on Q contents)

- All reachable vertices will be discovered, enqueued, marked with correct minimum link distances and valid predecessor nodes (except the start vertex), which are never changed again
(Induction on edge count from start node)