Breadth-First Search

Graphs

- $G = (V, E)$ where V is a set of vertices and E is a set of edges (v_1, v_2) where $v_1, v_2 \in V$
- typically, $n = |V|$ and $m = |E|$
- typically, elements of V are written as natural numbers

Types of graphs

- undirected
  - edges are unordered, i.e. $(v_1, v_2) = (v_2, v_1)$
- directed
  - edges are ordered, i.e. $(v_1, v_2) = (v_2, v_1)$

Representing graphs

Adjacency lists

- use list of size n, each slot i is the head of a linked list containing nodes that the vertex i is adjacent to
- works similarly for both directed and undirected graphs
- works for non-simple graphs
- size is $\Theta(n + m)$, small for sparse graphs
  - note that for simple graphs, $m \leq n^2$
- slow ($\mathcal{O}(m)$) searching

Adjacency matrices

- use matrix of size $n \times n$, where slot i, j stores a 1 if there is an edge from vertex i to vertex j
- works similarly for both directed and undirected graphs

- only works for simple graphs (though can store at most one self edge per vertex in slot i, i)
- size is $\Theta(n^2)$, necessarily big
- fast ($\mathcal{O}(1)$) searching

Graph search

- graph search is systematic exploration of a graph
- can reveal structural properties of a the graph
  - connectedness - is there a path between every two vertices?
- recording explored vertices:
  - colour v:
    - white if
    - grey if discovered but not yet explored
    - black if
  - set parent $p[u] = v$ if u was discovered while exploring v
    - records path from $stov$
  - store $d[v] = \ell$ where ℓ is the length of the discovery path from s to v
    - if $p[u] = v$, then $d[u] = d[v] + 1$

Breadth First Search

- starting from a node, explore neighours of one depth before visiting next depth

```
def BFS(G, s):
    colour[s] = "grey"
    d[s] = 0
    p[s] = NIL
    for each v in V-{s}:
        colour[v] = "white"
        d[v] = infinity
        p[v] = NIL
    Q = EmptyQueue()
    "..."
```

Time complexity

BFS is $\mathcal{O}(|V| + |E|)$ since each node needs to be discovered, and for each node every edge needs to be checked

Discovery path

Let $\delta(s, v)$ be the minimum distance between s and v

Lemma 1. If u is added to the queue Q before v, is then $d[u] \leq d[v]$

Suppose for contradiction that u, v is the first pair of vertices where v comes after u and $d[u] > d[v]$.

Theorem. After `BFS(G, s)`, for every $v \in V$, $d[v] = \delta(s, v)$. Thus, BFS finds shortest paths.

Suppose there exists $x \in V$ so that $d[x] \neq \delta(s, x)$ (clearly $x \neq s$).

- let v be the closest node from s such that $d[v] \neq \delta(s, v)$
- by lemma 0, $d[v]