BREDTH-FIRST SEARCH

Graph search:

"explore" a graph

Recall:

graph $G = (V, E)$

V = set of vertices

E = set of edges

undirected $\rightarrow e = \{v, w\}$ unordered pairs

directed $e = (v, w)$ ordered pairs

Applications:

- web crawling
- social networking
- network broadcast
- garbage collection
- model checking
- checking mathematical conj.
- solving puzzles games

Pocket Cube: 2×2×2

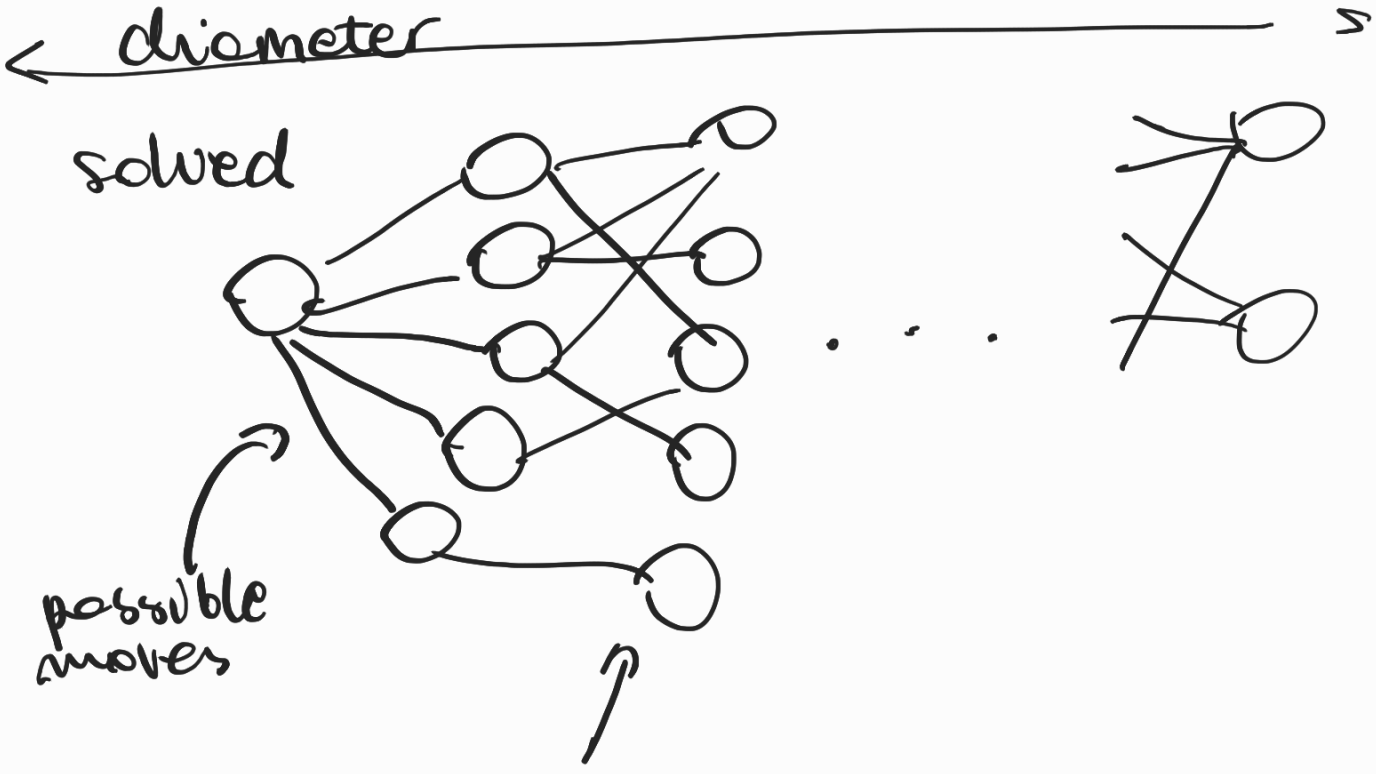
- **configuration graph**
  - vertex for each possible state of cube

$$\#\text{ vertices} = 8! \cdot 3^8 = 264{,}539{,}520$$

possibilities

8 small squares

- edge for each possible move



2x2x2 : 11

3x3x3 : 20

$n \times n \times n : \Theta\left(\frac{n^2}{\log n}\right)$

Graph representation:

Adjecency list:

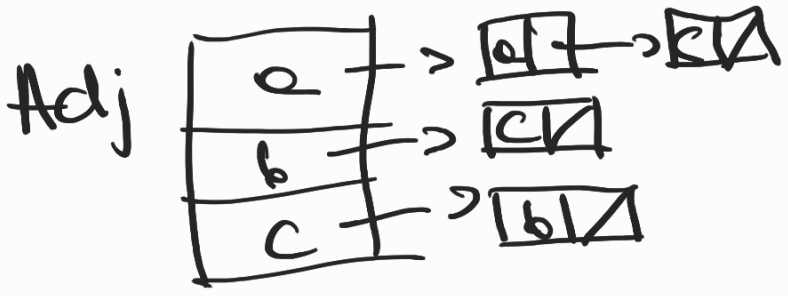
array Adj of $|V|$ linked lists

- for each vertex $u \in V$

Adj[u] store u's neighbors

Adj[v] store v's neighbours
↳ {v ∈ V | (u,v) ∈ E}

Adj[b] = {a, c}
Adj[a] = {c}
Adj[c] = {b}



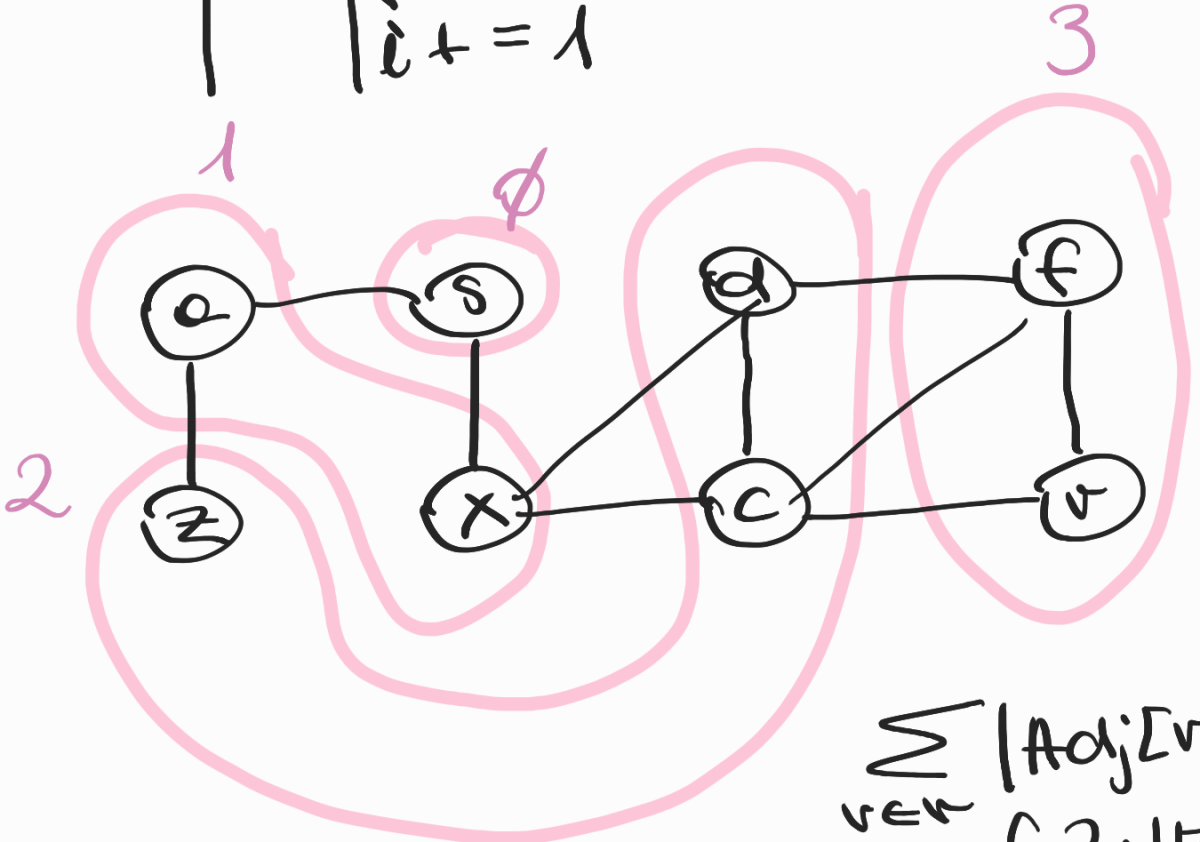
$\Theta(V+E)$

Object-oriented:
v.neighbour = Adj[v]

Implicit representation:
- Adj(u) is a function
- v.neighbours() is a method

BREADTH-FIRST SEARCH
- visit all nodes reachable from given $s \in V$
- $O(V+E)$ time
- look at nodes reachable in 0 moves, 1 move, 2 moves...
- careful to avoid duplicates

```
BFS(s, Adj):
    level = {s: 0}
    parent = {s: None}
    i = 1
    frontier = [s]
    while frontier:
        next = []
        for u in frontier:
            for v in Adj[u]:
                if v not in level:
                    level[v] = i
                    parent[v] = u
                    next.append(v)
        frontier = next
        i += 1
```



$$\sum_{v \in V} |Adj[v]| =$$

$= \begin{cases} 2 \cdot |E| & \text{undirected} \\ |E| & \text{directed} \end{cases}$

Shortest paths

$v \leftarrow parent[v]$

$\leftarrow parent[parent[v]]$

$\leftarrow \ldots$

$\leftarrow s$

is a shortest path from s to v