1 Breadth-First Search - Recap and alternative implementation (BFS) (CLRS §22.2)

BFS(s, Adj)

```
level = { s:0 }
parent = { s:None }
i = 1
frontier = [s]
while frontier not empty:
    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
```

Runtime:

- For any vertex $u \in V$, each vertex v in Adj$[u]$ enters `next`, then `frontier` *only once* (since v is added to `level` right before it is added to `next`).
  - For each vertex $u \in V$, Adj$[u]$ is looped through exactly one time
  - $\sum_{u \in V} |Adj[u]| = \Theta(E)$
- Total runtime: $\Theta(V + E)$

Remarks.

- For all v, the path (`v, parent[v], parent[parent[v]], ... , s`) in reverse is a shortest path from s to v of length `level[v]`
- The set of vertices we explored (all those reachable from s) together with the edges (`parent[v], v`) for each v give a **BFS tree**

Example

Can't have edges from level i to level j for $j \geq i + 2$

2 Depth-First Search (DFS) (CLRS §22.3)

Idea:

Example

```
u.color =
for v in
    if v.color ==
        v.parent
        DFS-VISIT
u.color =
time
u.f
```

We usually use DFS to learn something about the graph (as opposed to a vertex as in BFS). We therefore usually use:

```
for u in
    u.color =
    u.parent =
for u in
    if u.color ==
        DFS-VISIT
```

Runtime.

2.1 Edge Classification

Definition

- **Tree edge**

- **Forward edge**

- **Backward edge**

- **Cross edge**

How can we detect what type of edge (u, v) is?

- Tree edge:
- Forward edge:
- Backward edge:
- Cross edge:

To distinguish forward from cross edge,

- In forward edge

- In cross edge

Example

Example

Theorem 22.10 In a DFS of an undirected graph G, every edge is either

In other words,

Proof.

Corollary. An undirected graph is acyclic if and only if

Proof.

What about directed graphs?

Theorem The vertex v is a descendant of u in the DFS forest if and only if

Proof idea.

Lemma 22.11 A directed graph is acyclic if and only if

Proof.

3 Topological Sort (CLRS §22.4)

Tasks to be performed where some tasks must be completed before others. E.g. prerequisites for courses. Input is

Example

Definition

Algorithm.
Input:

1. Run

2. Output

Runtime.

Theorem. The given algorithm outputs

Proof.

4 Strongly Connected Components (SCC) (CLRS §22.5)

Example

Definition

Claim. The component graph is a DAG.

Proof.

Consider now DFS on the graph G. We can pretend that this DFS takes place in G^{SCC}, in the following sense: we say that

- a component is discovered when
- a component is finished when
- a component starts

Key lemma: The "pretend" DFS on G^{SCC} induced by the DFS on G is a valid DFS exploration of G^{SCC} .
Proof idea.

Observation. The last vertex to finish

We can now identify the entire left-most component by

SCC(G)

1. Call DFS(G) to compute
2. Call DFS(G^T) where in the main outer loop consider
3. Output

Runtime.

Example