Introduction to graph algorithms

Graph \( G = (V, E) \)

\[ V = \text{Set of vertices} = \# \text{vertices} \]
\[ E = \text{Set of edges} = \# \text{edges} \]

\[ E \leq \binom{V}{2} \quad \text{(undirected)} \]
\[ \leq V(V-1) \quad \text{(directed)} \]

Basic Question: Reachability

Given \( S, T \) find if \( S \rightarrow T \) path exists
(maybe: \& return path)

\[ \text{visited} = \mathbb{E}^3 \]

\begin{algorithm}
\text{def } \text{DFS}(v):
\begin{algorithmic}
  \State \text{if } v \text{ in visited: return False}
  \State visited.add(v)
  \If {v == t} \Return True
  \For {w in v.adj:}
  \If {DFS(w): return True}
  \EndFor
  \EndIf
\EndIf
\end{algorithmic}
\end{algorithm}
def BFS(s, t):
    Q = queue([s])
    visited = set()
    while Q:
        v = Q.pop_front()
        if v in visited: continue
        visited.add(v)
        if v == t:
            return True
        for w in v.adj:
            Q.push_back(w)
    return False

BFS: queue
DFS: stack

min. spanning tree: heap/priority queue (Prim's Algorithm)
shortest paths: priority queue on different weights (Dijkstra's Alg)

Claim: whatever the pop() used, the whatever first search returns True \iff \exists \text{ path}. 
add parent pointers to alg:

```python
def BFS(s, t):
    Q = queue([s, None])
    Parent = {}
    while Q:
        v, p = Q. pop_front()
        if v in Parent: continue
        Parent[v] = p
        if v == t: return True
        for w in v. adj:
            Q. push_back((w, v))
    return False
```

Consider the chain:

\[ t \rightarrow Parent[t] \rightarrow Parent[Parent[t]] \rightarrow \ldots \]

Except for \( (s, \text{None}) \), every \((v, \text{Parent}(v))\) pair has a corresponding \(\text{Parent}[v] \rightarrow v\) edge in \(E\).

Thus the chain either ends at \(s\) or in a loop.

Every step of this chain moves to a vertex visited earlier in the execution, so it must terminate; hence it (backwards) would be an \(s \rightarrow t\) path.
Pf ($\subseteq$) Suppose returns false, but \exists path.

Then \exists u visited, v not visited, u v exists.

u \neq t, when u visited, (v, u) added to Q

\implies v would be visited before Q empty, \implies \subseteq.

BFS: Return path w/ fewest steps
Exercises

1) Road networks: weights = max height of trucks taking road 
   \( l = \min \) height of bridges

   Q1: tallest truck that can go \( S \rightarrow T \)?

   Q2: can go between any pair of locations?

2) Floodfill: MS Paint