Lecture 20: Shortest paths, min. spanning trees

- Dijkstra's shortest path algorithm
- Greedy algorithms
- Prim's algo. for min. spanning trees

Announcements:
- PS 5 out soon
for all verts reachable from start are visited */
DFS (start)
worklist = new Set<Vert>
worklist.add (start)
visited = φ
while worklist ≠ φ
    for each edge v→w
        if w not in worklist
            add w to worklist
            mark v visited
end

BFS (start)
worklist = new Set<Vert>
worklist.add (start)
visited = φ
while worklist ≠ φ
    while worklist ≠ φ
        for each edge w→v
            if v not in visited
                visited.add (v)

Notation:

u, v are vertices
u→v there is an edge from u to v.
\( u \leadsto v \) there is a path of 0 or more edges from u to v.
\( u \leadsto u' \leadsto u'' \leadsto \ldots \leadsto v \)
da+b+c+...+dn
precondition: all edge weights are ≥ 0

Data structure:
- Min-heap
- Lookup structure (like in Dijkstra)
Dijkstra's:

**Invariant:**
- Visited list has all \( v \) with minimal path
- Started \( v \) in entire graph

**Worklist:** each \( v \) has minimal path
- starts in \( u \rightarrow v \)
- visited

\[ \text{Worklist: } (\text{start}, 0, \frac{1}{2} \text{ (heap)} \]
\[ \text{visited} = \emptyset \]

\[ \text{while worklist } \neq \emptyset : \]
- \( O(1) \) happens once.

\[ \text{total time } \leq O(\log V) \]

\[ \text{total time: } O(\log n \cdot |E|) \]
- happens for each edge

\[ |E| \leq |V|^2 \]

\[ a = \text{size of worklist } \leq |V| \]

\[ O((|V| + |E|) \log |V|) \] is \( O(|V|^2 \log |V|) \).

\[ |E| \leq |V|^2 \]

\[ |E| \times |V| \quad O(|V| \log |V|) \leq \]
A path $u \rightarrow v$ is a sequence of edges $u \rightarrow u' \rightarrow u'' \rightarrow \cdots \rightarrow v$

A cycle is a non-empty path from $v \rightarrow v$.

$G$ is cyclic if it contains any cycles, acyclic otherwise.

$DAG$: directed acyclic graph.

(in undirected graphs)

A tree $T$ is a graph satisfying 3 properties:

1. $T$ is acyclic
2. $T$ is connected
   (a collection of trees is a forest)
3. $\#\text{ vert of } T = \#\text{ edges} + 1$

1 and 2 imply 3
2 and 3 imply 1
1 and 3 imply 2