Lecture 20: Shortest paths, min. spanning trees

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

Announcements:

- P5 out soon

postcondition:
/** all verts reachable from start are visited */

DFS (start)

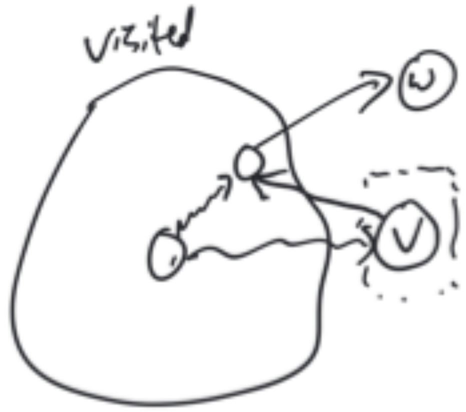
init { worklist = new Set(Verts)
worklist.add(start)
visited = ∅

term { while worklist ≠ ∅
remove v from wklist
mark v visited
for each edge v → u
if u not in wklist,
add u to wklist.

BFS (start)
worklist = new Set(verts)
worklist.add(start)
visited = ∅

while worklist ≠ ∅

same



everything v in visited set reachable by path start ⇝ v only cont. visited verts.

worklist contain all unvisited vertices v with a path start ⇝ v using only visited vertices ✓

Notation:

u, v are vertices

$u \xrightarrow{1} v$ there is an edge from u to v.

$u \overset{d}{\rightsquigarrow} v$ there is a path of 0 or more edges from u to v.

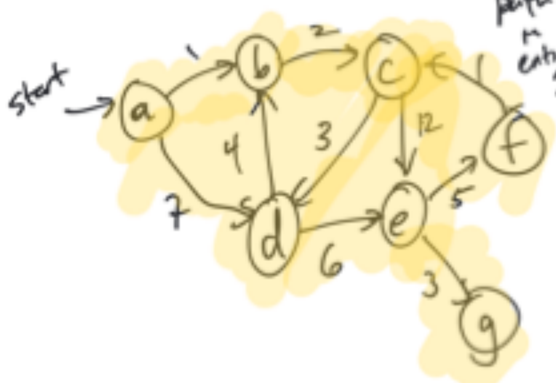
$u \xrightarrow{d_1} u' \xrightarrow{d_2} u'' \xrightarrow{d_3} \cdots \rightarrow v$

$d = d_1 + d_2 + \cdots + d_n$

additional BFS invariant:

- everything visited has a path length ≤ the shortest path to worklist
  - fewest # edges.
- worklist sorted by path length

precondition: all edge weights are ≥ 0



minimal path in entire graph } every v has a path start ⇝ v (only through visited)

Visited

vertex	distance from src	prev
a	0	start
b	1	start ⇝ $\boxed{a}$ → b
c	3	start ⇝ $\boxed{b}$ → c
d	6	start ⇝ $\boxed{c}$ → d
e	12	start ⇝ $\boxed{d}$ → e
~~g~~	15	start ⇝ $\boxed{e}$ → g
f	17	start ⇝ e ⇝ f

every v has a path start ⇝ u → v in visited set } minimal path only passing through visited.

worklist

vertex	distance from src.
~~a~~	~~Ø~~ start ⇝ a
~~b~~	~~0+1 = 1~~ start ⇝ a → b
~~d~~	~~0+7 = 7~~ start ⇝ a → d
~~c~~	~~1 + 2 = 3~~ start ⇝ b → c
~~e~~	~~3 + 12 = 15~~ start ⇝ c → e
~~d~~	~~3 + 3 = 6~~ start ⇝ c → d
~~e~~	~~6 + 6 = 12~~ start ⇝ d → e
~~f~~	~~12 + 5 = 17~~ start ⇝ e → f
~~g~~	~~12 + 3 = 15~~ start ⇝ e → g

start ⇝ a $\xrightarrow{7}$ d = 7

start ⇝ c $\xrightarrow{3}$ d = 6

start ⇝ d $\xrightarrow{6}$ e = 12

remove smallest from wklist!

a → b → c → d → e → g

worklist:

- add element with "score"
- check if empty
- remove: return lowest "score"
- change "score"

Data structure:

- Min-heap.
- Lookup structure (like in P4)

Dijkstra's:

invariant:

Visited list has all v with minimal path start to v in entire graph

worklist: each v has minimal path start $\rightsquigarrow u \to v$ with start $\rightsquigarrow u$ visited.

Dijkstra:

$$\left.\begin{array}{l}\text{worklist} = \{(\overset{\text{node}}{\text{start}}, \overset{\text{distance}}{0})\} \text{ (heap)} \\ \text{visited} = \emptyset\end{array}\right\} O(1) \text{ happens once.}$$

while worklist $\neq \emptyset$:

total time is $O(|V| \log |V|)$

- remove min elt v from wklist
- visit v.
- for each $v \to u$:

total time: $O(\log n \cdot |E|)$ which is $O((\log |V|) \cdot |E|)$

happen for each edge

- if u in worklist:
  - update distance to u. $\}O(\log n)$ where n is size of wklist
- else
  - add u to wklist. $\}O(\log n)$

happens once per vertex

n = size of wklist $\leq |V|$

$\boxed{O((|V|+|E|) \log |V|)}$ is $O(|V|^2 \log |V|)$.

gauranteed: $|E| \leq |V|^2$

common: $|E| \approx |V|$ $\qquad O(|V| \log |V|) \Leftarrow$

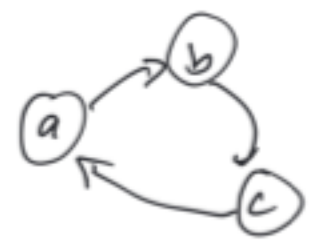
A path $u \rightsquigarrow v$ is a sequence of edges

$u \to u' \to u'' \cdots \to v$

A cycle is a non-empty path from

$v \rightsquigarrow v$.

G is <u>cyclic</u> if it contains any cycles, <u>acyclic</u> otherwise



$a \to b \to c \to a$

$a \rightsquigarrow a$

DAG: directed acyclic graph.

(in undirected graphs)

a <u>tree</u> T is a graph satisfying 3 properties:

1. T is acyclic
2. T is connected

   (a collection of trees is a forest)

3. # <u>verts</u> of T = #<u>edges</u> +1

1 and 2 imply 3

2 and 3 imply 1

1 and 3 imply 2