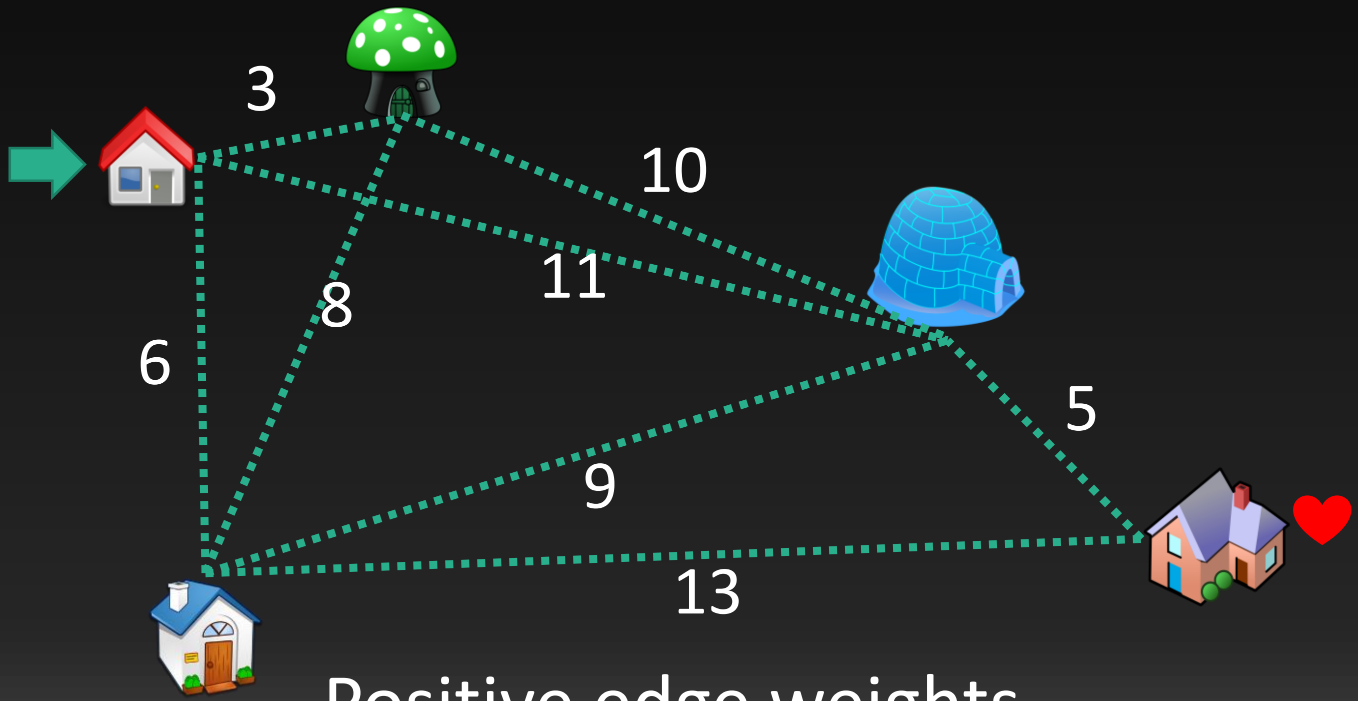
CS 4800: Algorithms & Data

Lecture 17

March 21, 2017

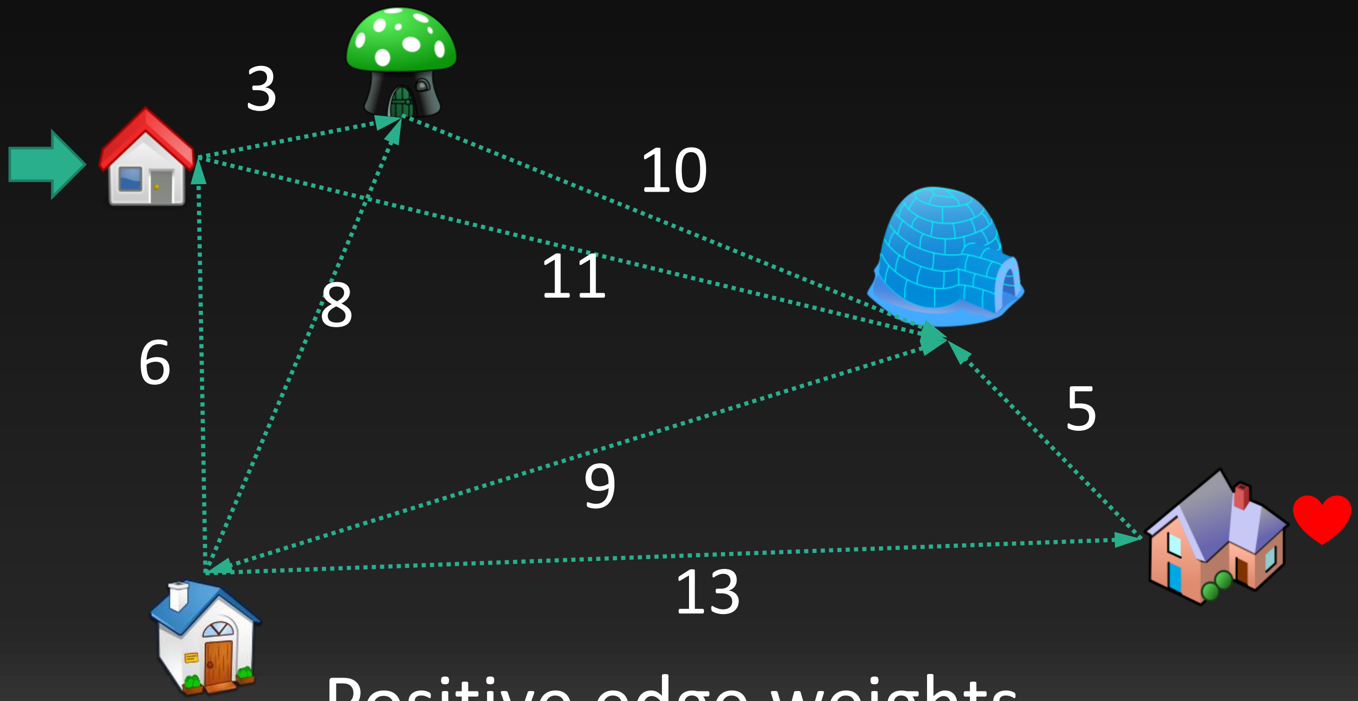
Shortest paths

What is the fastest way to get from A to B?



Positive edge weights

Directed graphs



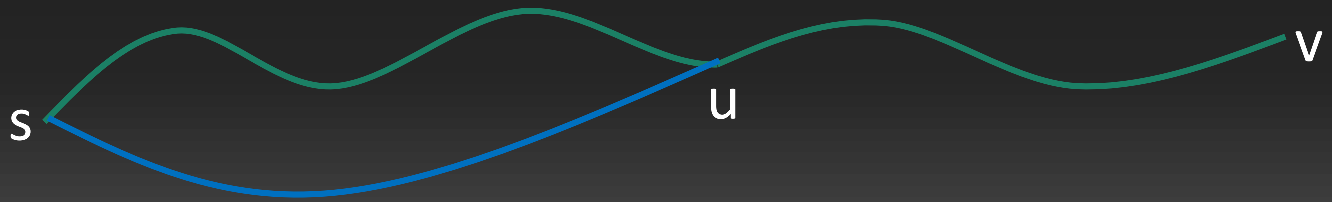
Positive edge weights

Dynamic programming

- Source vertex s
- d(v): length of shortest **tentative** path from s to v
- d*(v): length of shortest path from s to v
- pred(v): predecessor of v in shortest **tentative** path from s to v

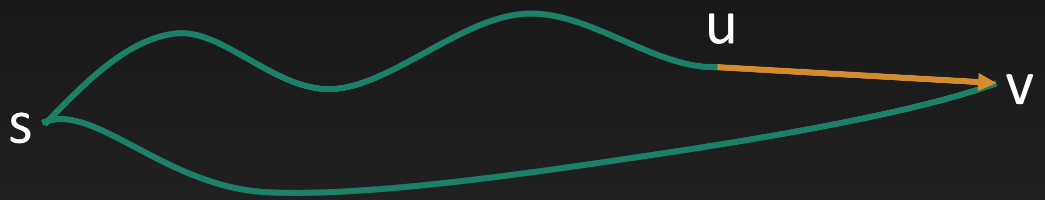
Optimal substructure

- Consider shortest path P from s to v
- Let u be a vertex on P
- The subpath of P from s to u must be shortest path from s to u
- If there is a shorter path from s to u then there is a shorter path from s to v than P



Relation among shortest distances

- Consider arbitrary edge (u,v)
- $d^*(v) \leq d^*(u) + w(u,v)$
- The path $s \to u \to v$ is a feasible path from s to v



- Edge (u,v) is **tense** if $d(v) > d(u) + w(u,v)$
- When a tense edge is found, can improve $d(v)$

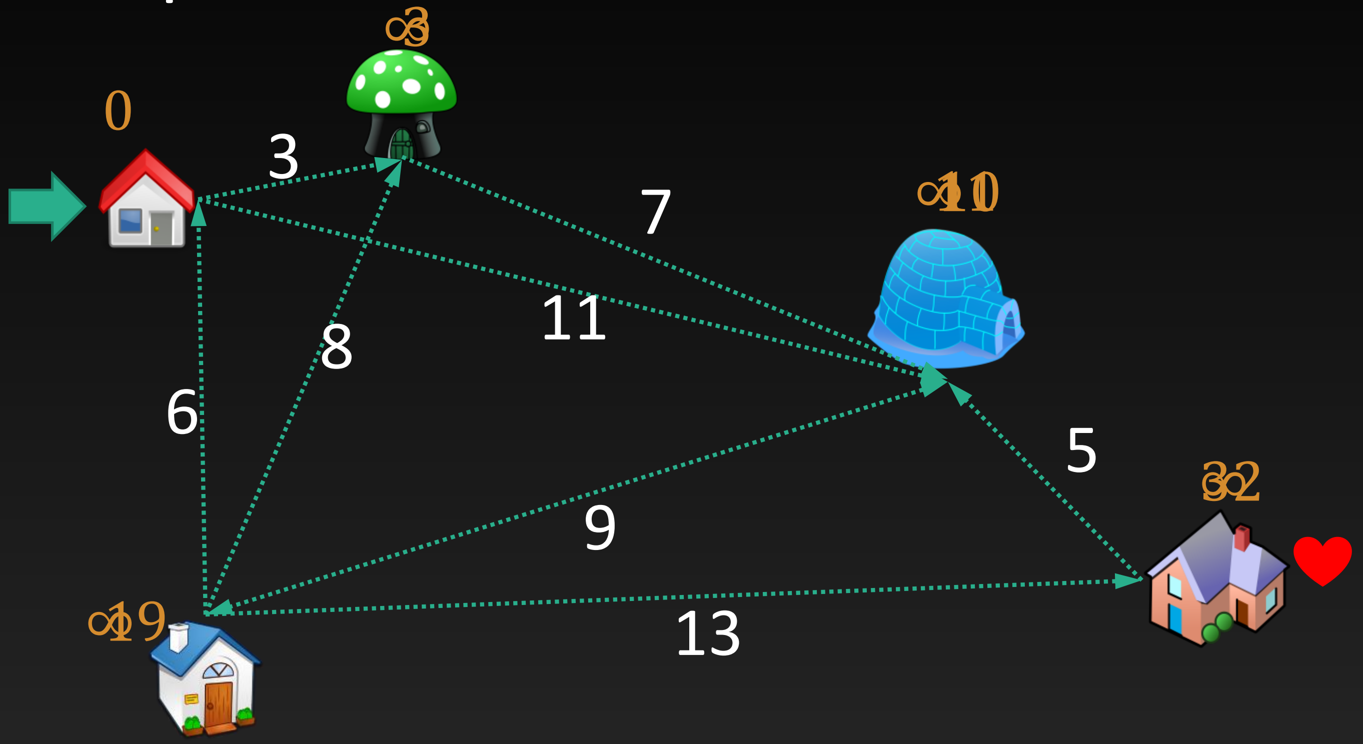
Generic shortest path algorithm

```
• Initialize $d(s) = 0$ and $d(v) = \infty$ for all $v \neq s$
• $Q \leftarrow \{s\}$
• While $Q \neq \emptyset$
    • Remove some u from Q
    • For all edges $u \rightarrow v$
        • If $d(v) > d(u) + w(u, v)$
            • $d(v) \leftarrow d(u) + w(u, v)$
            • $pred(v) \leftarrow u$
            • If $v \notin Q$, put $v$ in $Q$. Otherwise, DecreaseKey($v$).
```

Dijkstra's algorithm

- Initialize $d(s) = 0$ and $d(v) = \infty$ for all $v \neq s$
- $Q \leftarrow \{s\}$
- While $Q \neq \emptyset$
    - **Remove u with minimum d(u) from Q**
    - For all edges $u \rightarrow v$
        - If $d(v) > d(u) + w(u, v)$
            - $d(v) \leftarrow d(u) + w(u, v)$
            - $pred(v) \leftarrow u$
            - If $v \notin Q$, put v in Q. Otherwise, DecreaseKey(v).

Example



Q:

Correctness of Dijkstra’s

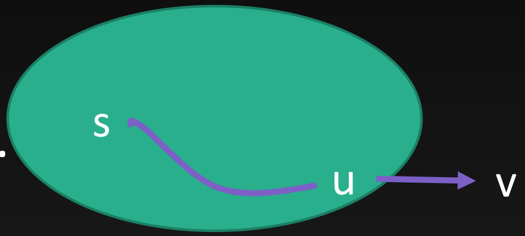
Theorem. Let S be set of nodes removed from Q. For all v in S, we have d(v)=d*(v) when v is removed from Q.

Proof. Induction over number of iterations.

First node to be removed is s and $d(s) = d^*(s) = 0$.

Assume claim is true for first k nodes.

Let v be the k+1st about to be removed. Let u = pred(v).



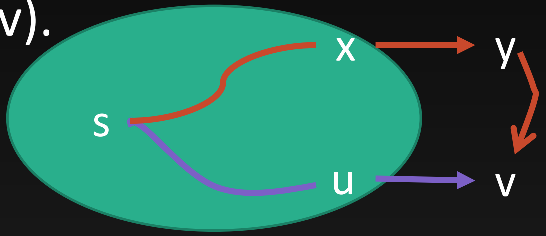
Correctness of Dijkstra's

Let v be the k+1st about to be removed. Let u = pred(v).

u is removed from Q before (when we set pred(v) = u), so d(u) = d*(u).

Consider any other path P from s to v not via edge (u,v).

P must leave S at some point via edge (x,y).



v is about to be removed, not y, so $d(v) \leq d(y)$.

x is removed from Q so $d(x) = d^*(x)$

$d(y) \leq d(x) + w(x,y) \leq$ distance from s to y on P

Thus, $d(v) \leq Length(P)$.

Therefore, $d(v) = d^*(v)$.

Running time

```
• Initialize $d(s) = 0$ and $d(v) = \infty$ for all $v \neq s$
• $Q \leftarrow \{s\}$
• While $Q \neq \emptyset$
V times • **Remove u with minimum d(u) from Q** ← O(log V)
            • For all edges $u \rightarrow v$
                • If $d(v) > d(u) + w(u,v)$
                    • $d(v) \leftarrow d(u) + w(u,v)$
                    • $pred(v) \leftarrow u$
E times             • if $v \notin Q$, insert $v$ into $Q$. Otherwise DecreaseKey(v) ← O(log V)
```

O((V+E)log V) time

Breadth-first search

- All edge weights are 1
- Distance = #edges on the path

Breadth-first search

```
Initialize $d(s) = 0$ and $d(v) = \infty$ for all $v \neq s$
$Q \leftarrow (s,)$ ← Q is a queue: first in first out
While $Q \neq \emptyset$
    Remove first $u$ in $Q$
    For all edges $u \rightarrow v$
        If $d(v) > d(u) + 1$
            $d(v) \leftarrow d(u) + 1$
            $pred(v) \leftarrow u$
            Put $v$ last in $Q$
```