17.3.6
How to compute the *i*th closest vertex?

Finding the ith closest node

1. X contains the $i-1$ closest nodes to s
2. Want to find the ith closest node from $V-X$.

1. For each $u \in V-X$ let $P(s,u,X)$ be a shortest path from s to u using only nodes in X as intermediate vertices.
2. Let $d'(s,u)$ be the length of $P(s,u,X)$

Observations: for each $u \in V-X$,

1. $\mathrm{dist}(s,u) \le d'(s,u)$ since we are constraining the paths
2. $d'(s,u) = \min_{t \in X}(\mathrm{dist}(s,t) + \ell(t,u))$ - Why?

Lemma (d' has the right value for ith vertex)

If v is the ith closest node to s, then $d'(s,v) = \mathrm{dist}(s,v)$.

Finding the ith closest node

1. X contains the $i-1$ closest nodes to s
2. Want to find the ith closest node from $V - X$.

1. For each $u \in V - X$ let $P(s, u, X)$ be a shortest path from s to u using only nodes in X as intermediate vertices.
2. Let $d'(s, u)$ be the length of $P(s, u, X)$

Observations: for each $u \in V - X$,

1. $\mathrm{dist}(s, u) \leq d'(s, u)$ since we are constraining the paths
2. $d'(s, u) = \min_{t \in X}(\mathrm{dist}(s, t) + \ell(t, u))$ - Why?

Lemma (d' has the right value for ith vertex)

If v is the ith closest node to s, then $d'(s, v) = \mathrm{dist}(s, v)$.

Finding the ith closest node

1. X contains the $i-1$ closest nodes to s
2. Want to find the ith closest node from $V - X$.

1. For each $u \in V - X$ let $P(s, u, X)$ be a shortest path from s to u using only nodes in X as intermediate vertices.
2. Let $d'(s, u)$ be the length of $P(s, u, X)$

Observations: for each $u \in V - X$,

1. $\mathrm{dist}(s, u) \leq d'(s, u)$ since we are constraining the paths
2. $d'(s, u) = \min_{t \in X}(\mathrm{dist}(s, t) + \ell(t, u))$ - Why?

Lemma (d' has the right value for ith vertex)

If v is the ith closest node to s, then $d'(s, v) = \mathrm{dist}(s, v)$.

Finding the ith closest node

Lemma (d' has the right value for ith vertex)

Given:

1. X: *Set of* $i - 1$ *closest nodes to* s.
2. $d'(s, u) = \min_{t \in X}(\mathrm{dist}(s, t) + \ell(t, u))$

If v *is an* i*th closest node to* s*, then* $d'(s, v) = \mathrm{dist}(s, v)$.

Proof.

Let v be the ith closest node to s. Then there is a shortest path P from s to v that contains only nodes in X as intermediate nodes (see previous claim). Therefore $d'(s, v) = \mathrm{dist}(s, v)$. □

Finding the ith closest node

Lemma (d' has the right value for ith vertex)

If v is an ith closest node to s, then $d'(s,v) = \mathrm{dist}(s,v)$.

Corollary

The ith closest node to s is the node $v \in V - X$ such that $d'(s,v) = \min_{u \in V-X} d'(s,u)$.

Proof.

For every node $u \in V - X$, $\mathrm{dist}(s,u) \leq d'(s,u)$ and for the ith closest node v, $\mathrm{dist}(s,v) = d'(s,v)$. Moreover, $\mathrm{dist}(s,u) \geq \mathrm{dist}(s,v)$ for each $u \in V - X$. $\square$

Algorithm

```
Initialize for each node v: dist(s, v) = ∞
Initialize X = ∅, d'(s, s) = 0
for i = 1 to |V| do
    (* Invariant: X contains the i − 1 closest nodes to s *)
    (* Invariant: d'(s, u) is shortest path distance from u to s
     using only X as intermediate nodes*)
    Let v be such that d'(s, v) = min_{u∈V−X} d'(s, u)
    dist(s, v) = d'(s, v)
    X = X ∪ {v}
    for each node u in V − X do
        d'(s, u) = min_{t∈X} (dist(s, t) + ℓ(t, u))
```

Correctness: By induction on i using previous lemmas.

Running time: $O(n \cdot (n + m))$ time.

1. n outer iterations. In each iteration, $d'(s, u)$ for each u by scanning all edges out of nodes in X; $O(m + n)$ time/iteration.

Algorithm

```
Initialize for each node v: dist(s, v) = ∞
Initialize X = ∅, d'(s, s) = 0
for i = 1 to |V| do
    (* Invariant: X contains the i − 1 closest nodes to s *)
    (* Invariant: d'(s, u) is shortest path distance from u to s
     using only X as intermediate nodes*)
    Let v be such that d'(s, v) = min_{u∈V−X} d'(s, u)
    dist(s, v) = d'(s, v)
    X = X ∪ {v}
    for each node u in V − X do
        d'(s, u) = min_{t∈X} (dist(s, t) + ℓ(t, u))
```

Correctness: By induction on i using previous lemmas.

Running time: $O(n \cdot (n + m))$ time.

1. n outer iterations. In each iteration, $d'(s, u)$ for each u by scanning all edges out of nodes in X; $O(m + n)$ time/iteration.

Algorithm

```
Initialize for each node v: dist(s, v) = ∞
Initialize X = ∅, d'(s, s) = 0
for i = 1 to |V| do
    (* Invariant: X contains the i − 1 closest nodes to s *)
    (* Invariant: d'(s, u) is shortest path distance from u to s
     using only X as intermediate nodes*)
    Let v be such that d'(s, v) = min_{u∈V−X} d'(s, u)
    dist(s, v) = d'(s, v)
    X = X ∪ {v}
    for each node u in V − X do
        d'(s, u) = min_{t∈X} (dist(s, t) + ℓ(t, u))
```

Correctness: By induction on i using previous lemmas.

Running time: $O(n \cdot (n + m))$ time.

1. n outer iterations. In each iteration, $d'(s, u)$ for each u by scanning all edges out of nodes in X; $O(m + n)$ time/iteration.

Algorithm

```
Initialize for each node v: dist(s, v) = ∞
Initialize X = ∅, d'(s, s) = 0
for i = 1 to |V| do
    (* Invariant: X contains the i − 1 closest nodes to s *)
    (* Invariant: d'(s, u) is shortest path distance from u to s
     using only X as intermediate nodes*)
    Let v be such that d'(s, v) = min_{u∈V−X} d'(s, u)
    dist(s, v) = d'(s, v)
    X = X ∪ {v}
    for each node u in V − X do
        d'(s, u) = min_{t∈X} (dist(s, t) + ℓ(t, u))
```

Correctness: By induction on i using previous lemmas.

Running time: $O(n \cdot (n + m))$ time.

1. n outer iterations. In each iteration, $d'(s, u)$ for each u by scanning all edges out of nodes in X; $O(m + n)$ time/iteration.

THE END

...

(for now)