Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \to Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \to Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.

$S = \phi$.

Find $u = \text{argmin}_{v \notin S} d(v)$.

update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \rightarrow Z$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \rightarrow Z$.
$c_\phi(e = (u, v)) = \phi(u) - \phi(v) + w(e)$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V,E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s,v)$ $\forall v \in V$.

$d(s,v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \rightarrow Z$.
$c_\phi(e = (u,v)) = \phi(u) - \phi(v) + w(e)$.
Note: $d(v) \leq d(u) + w(e)$

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \to Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \to Z$.
$c_\phi(e = (u, v)) = \phi(u) - \phi(v) + w(e)$.
Note: $d(v) \leq d(u) + w(e) \implies d(u) + w(e) - d(v) \geq 0$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \to Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \to Z$.
$c_\phi(e = (u, v)) = \phi(u) - \phi(v) + w(e)$.
Note: $d(v) \leq d(u) + w(e) \implies d(u) + w(e) - d(v) \geq 0$.
$\phi(v) = d(v)$ produces non-negative edge weights.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \to Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \to Z$.
$c_\phi(e = (u, v)) = \phi(u) - \phi(v) + w(e)$.
Note: $d(v) \leq d(u) + w(e) \implies d(u) + w(e) - d(v) \geq 0$.
$\phi(v) = d(v)$ produces non-negative edge weights.
Shortest path under $c_\phi(e)$ is same as under $w(e)$.

Bellman-Ford. Djikstra. Price Functions.

Given $G = (V, E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s, v)$ $\forall v \in V$.

$d(s, v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v) = \infty, d(s) = 0$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u, v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \rightarrow Z$.
$c_\phi(e = (u, v)) = \phi(u) - \phi(v) + w(e)$.
Note: $d(v) \leq d(u) + w(e) \implies d(u) + w(e) - d(v) \geq 0$.
$\phi(v) = d(v)$ produces non-negative edge weights.
Shortest path under $c_\phi(e)$ is same as under $w(e)$.
p from s to t, $\sum_{e \in p} c_\phi(e) = \phi(t) - \phi(s) + w(p)$.

Bellman-Ford. Djikstra. Price Functions.

Given $G=(V,E)$, $w : E \rightarrow Z$, on edges, and $s \in V$, find $d(s,v)$ $\forall v \in V$.

$d(s,v)$ - length of shortest path.

Djikstra: Non-Negative edge weights: $d(v)=\infty, d(s)=0$.
$S=\phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e=(u,v), d(v)=\min(d(v), d(u)+w(e)$.
$S = S+u$.

(Reachable) Negative cycle, answer is undefined.

Find price Function: $\phi : V \rightarrow Z$.
$c_\phi(e=(u,v)) = \phi(u) - \phi(v) + w(e)$.
Note: $d(v) \leq d(u)+w(e) \implies d(u)+w(e)-d(v) \geq 0$.
$\phi(v) = d(v)$ produces non-negative edge weights.
Shortest path under $c_\phi(e)$ is same as under $w(e)$.
p from s to t, $\sum_{e \in p} c_\phi(e) = \phi(t) - \phi(s) + w(p)$.

Thus: $d(s,v)$ is a price fucntion whose reduced costs make all edge weights positive.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v)=0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e=(u,v)$, $w(e) \leq 0$, $d(v)=\min(d(v),d(u)+w(e))$.
$S=\phi$.
Find $u=\text{argmin}_{v \notin S} d(v)$.
update(u): for $e=(u,v), d(v)=\min(d(v),d(u)+w(e)$.
$S=S+u$.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

Claim: After k rounds, $d(v) \leq$ path length with $\leq k$ negative edges.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

Claim: After k rounds, $d(v) \leq$ path length with $\leq k$ negative edges.

Induction.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

Claim: After k rounds, $d(v) \leq$ path length with $\leq k$ negative edges.

Induction.

$O(n)$ iterations of Bellman/Dijkstra is good.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

Claim: After k rounds, $d(v) \leq$ path length with $\leq k$ negative edges.

Induction.

$O(n)$ iterations of Bellman/Dijkstra is good.

$O(n(n + m\log n))$

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

Claim: After k rounds, $d(v) \leq$ path length with $\leq k$ negative edges.

Induction.

$O(n)$ iterations of Bellman/Dijkstra is good.

$O(n(n + m\log n))$ or slightly better.
Quadratic time.

Bellman/Dijkstra.

Approach: Add s, with $w(s,v) = 0$ for all $v \in V$.

Bellman/Djikstra Round: Have $d(v)$.
For all $e = (u,v)$, $w(e) \leq 0$, $d(v) = \min(d(v), d(u) + w(e))$.
$S = \phi$.
Find $u = \text{argmin}_{v \notin S} d(v)$.
update(u): for $e = (u,v), d(v) = \min(d(v), d(u) + w(e)$.
$S = S + u$.

Claim: After k rounds, $d(v) \leq$ path length with $\leq k$ negative edges.

Induction.

$O(n)$ iterations of Bellman/Dijkstra is good.

$O(n(n + m \log n))$ or slightly better.
Quadratic time.

Scaling algorithm: $O(m\sqrt{n} \log nC)$ by Goldberg.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s, v) = 0$.
Todo: price function that ensures all edge weights ≥ -1.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.

Todo: price function that ensures all edge weights ≥ -1.

Price function in $G_{\geq -1}$ ($w(e) < 0 \rightarrow w'(e) = w(e) - 1$)

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.

Todo: price function that ensures all edge weights ≥ -1.

Price function in $G_{\geq -1}$ $(w(e) < 0 \rightarrow w'(e) = w(e) - 1)$

Price function, *phi*, in $G_{\geq -1}$.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.

Todo: price function that ensures all edge weights ≥ -1.

Price function in $G_{\geq -1}$ $(w(e) < 0 \rightarrow w'(e) = w(e) - 1)$

Price function, *phi*, in $G_{\geq -1}$.

$w'_\phi(e = (u,v)) = w'(e) - \phi(v) + \phi(u) \geq 0.$

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.

Todo: price function that ensures all edge weights ≥ -1.

Price function in $G_{\geq -1}$ ($w(e) < 0 \rightarrow w'(e) = w(e) - 1$)

Price function, *phi*, in $G_{\geq -1}$.

$w'_\phi(e = (u,v)) = w'(e) - \phi(v) + \phi(u) \geq 0$.

$\implies w_\phi(e) = w(e) - \phi(v) + \phi(u) \geq -1$.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.
Todo: price function that ensures all edge weights ≥ -1.
Price function in $G_{\geq -1}$ ($w(e) < 0 \rightarrow w'(e) = w(e) - 1$)

Price function, *phi*, in $G_{\geq -1}$.
$w'_\phi(e = (u,v)) = w'(e) - \phi(v) + \phi(u) \geq 0$.
$\implies w_\phi(e) = w(e) - \phi(v) + \phi(u) \geq -1$.

Some sort of “Scaling”..

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.
  Todo: price function that ensures all edge weights ≥ -1.
    Price function in $G_{\geq -1}$ ($w(e) < 0 \rightarrow w'(e) = w(e) - 1$)

Price function, *phi*, in $G_{\geq -1}$.
  $w'_{\phi}(e = (u,v)) = w'(e) - \phi(v) + \phi(u) \geq 0$.
  $\implies w_{\phi}(e) = w(e) - \phi(v) + \phi(u) \geq -1$.

Some sort of “Scaling”..
  Max negative weight $W \rightarrow W/2 \rightarrow W/4$....

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.
Todo: price function that ensures all edge weights ≥ -1.
Price function in $G_{\geq -1}$ $(w(e) < 0 \rightarrow w'(e) = w(e) - 1)$

Price function, *phi*, in $G_{\geq -1}$.
$w'_\phi(e = (u,v)) = w'(e) - \phi(v) + \phi(u) \geq 0$.
$\implies w_\phi(e) = w(e) - \phi(v) + \phi(u) \geq -1$.

Some sort of "Scaling"..
Max negative weight $W \rightarrow W/2 \rightarrow W/4$....
$O(\log W)$.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.

Todo: price function that ensures all edge weights ≥ -1.

Price function in $G_{\geq -1}$ $(w(e) < 0 \to w'(e) = w(e) - 1)$

Price function, *phi*, in $G_{\geq -1}$.

$w'_\phi(e = (u,v)) = w'(e) - \phi(v) + \phi(u) \geq 0$.

$\implies w_\phi(e) = w(e) - \phi(v) + \phi(u) \geq -1$.

Some sort of "Scaling"..

Max negative weight $W \to W/2 \to W/4$....

$O(\log W)$.

Some complications due to path length n.

Scaling Idea: Restricted Shortest Path Instance.

$O(\log C)$ Reduction to the following problem.

Edge weights ≥ -2, minimum cycle mean ≥ 1, add s with $w(s,v) = 0$.

- Todo: price function that ensures all edge weights ≥ -1.
  - Price function in $G_{\geq -1}$ ($w(e) < 0 \rightarrow w'(e) = w(e) - 1$)

Price function, *phi*, in $G_{\geq -1}$.

- $w'_{\phi}(e = (u,v)) = w'(e) - \phi(v) + \phi(u) \geq 0$.
- $\implies w_{\phi}(e) = w(e) - \phi(v) + \phi(u) \geq -1$.

Some sort of "Scaling"..

- Max negative weight $W \rightarrow W/2 \rightarrow W/4$....
  - $O(\log W)$.
  - Some complications due to path length n.
    - E.g. Maximum negative length path is nW.

Decomposition.

Working with $G_{\geq -1}$.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either “trivial” (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
(1) Progress: W.h.p. s.c.c, C, in G/S either
(i) either $|C| \leq \frac{3}{4}|V|$

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
  (1) Progress: W.h.p. s.c.c, C, in G/S either
    (i) either $|C| \leq \frac{3}{4}|V|$
    (ii) or $\kappa(C) \leq \kappa/2$.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
(1) Progress: W.h.p. s.c.c, C, in G/S either
(i) either $|C| \leq \frac{3}{4}|V|$
(ii) or $\kappa(C) \leq \kappa/2$.
(2) shortest path P, $|P \cap S| = O(\log n)$.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
(1) Progress: W.h.p. s.c.c, C, in G/S either
(i) either $|C| \leq \frac{3}{4}|V|$
(ii) or $\kappa(C) \leq \kappa/2$.
(2) shortest path P, $|P \cap S| = O(\log n)$.

Algorithm:

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
(1) Progress: W.h.p. s.c.c, C, in G/S either
(i) either $|C| \leq \frac{3}{4}|V|$
(ii) or $\kappa(C) \leq \kappa/2$.
(2) shortest path P, $|P \cap S| = O(\log n)$.

Algorithm:
(1) recursively build price functions.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
  (1) Progress: W.h.p. s.c.c, C, in G/S either
    (i) either $|C| \leq \frac{3}{4}|V|$
    (ii) or $\kappa(C) \leq \kappa/2$.
  (2) shortest path P, $|P \cap S| = O(\log n)$.

Algorithm:
  (1) recursively build price functions.
  (2) do $O(\log n)$ iterations of Bellman/Dijkstra.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either “trivial” (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
(1) Progress: W.h.p. s.c.c, C, in G/S either
(i) either $|C| \leq \frac{3}{4}|V|$
(ii) or $\kappa(C) \leq \kappa/2$.
(2) shortest path P, $|P \cap S| = O(\log n)$.

Algorithm:
(1) recursively build price functions.
(2) do $O(\log n)$ iterations of Bellman/Dijkstra.
All edges in clusters have positive weight.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either “trivial” (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
(1) Progress: W.h.p. s.c.c, C, in G/S either
(i) either $|C| \leq \frac{3}{4}|V|$
(ii) or $\kappa(C) \leq \kappa/2$.
(2) shortest path P, $|P \cap S| = O(\log n)$.

Algorithm:
(1) recursively build price functions.
(2) do $O(\log n)$ iterations of Bellman/Dijkstra.
All edges in clusters have positive weight.
All paths cross clusters $O(\log n)$ times.

Decomposition.

Working with $G_{\geq -1}$.

$\kappa(G)$ – maximum number of negative edges in any shortest path from s.

Note: path is either "trivial" (single edge from s) or negative.

Decomposition Claim: Fast algorithm that finds S, s.t.,
(1) Progress: W.h.p. s.c.c, C, in G/S either
(i) either $|C| \leq \frac{3}{4}|V|$
(ii) or $\kappa(C) \leq \kappa/2$.
(2) shortest path P, $|P \cap S| = O(\log n)$.

Algorithm:
(1) recursively build price functions.
(2) do $O(\log n)$ iterations of Bellman/Dijkstra.
All edges in clusters have positive weight.
All paths cross clusters $O(\log n)$ times.

$O((m + n\log n)\log^2 n)$ time.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Diameter of strongly connected component, C:

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Diameter of strongly connected component, C:
$D(C) = \max_{u,v \in C} d(u, v)$.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Diameter of strongly connected component, C:
$D(C) = \max_{u,v \in C} d(u,v)$.

Claim: Any negative path uses at most $D(C)$ negative edges in $G_{\geq -1}$ in C.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Diameter of strongly connected component, C:
$D(C) = \max_{u,v \in C} d(u,v)$.

Claim: Any negative path uses at most $D(C)$ negative edges in $G_{\geq -1}$ in C.

Proof: κ - number of neg. edges in path.

Neg. edge$G_{\geq -1}$ is one more negative in G.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Diameter of strongly connected component, C:

$D(C) = \max_{u,v \in C} d(u,v)$.

Claim: Any negative path uses at most $D(C)$ negative edges in $G_{\geq -1}$ in C.

Proof: κ - number of neg. edges in path.

Neg. edge$G_{\geq -1}$ is one more negative in G.

path $< -\kappa$ where κ is negative edges.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Diameter of strongly connected component, C:
$D(C) = \max_{u,v \in C} d(u,v)$.

Claim: Any negative path uses at most $D(C)$ negative edges in $G_{\geq -1}$ in C.

Proof: κ - number of neg. edges in path.

Neg. edge$G_{\geq -1}$ is one more negative in G.

path $< -\kappa$ where κ is negative edges.

but path between endpoints of length $\leq D(C)$.

Low Diameter Strongly Connected Components.

$G_{\geq 0}$ All negative weights set to 0.

Diameter of strongly connected component, C:
$D(C) = \max_{u,v \in C} d(u,v)$.

Claim: Any negative path uses at most $D(C)$ negative edges in $G_{\geq -1}$ in C.

Proof: κ - number of neg. edges in path.

Neg. edge$G_{\geq -1}$ is one more negative in G.
path $< -\kappa$ where κ is negative edges.
but path between endpoints of length $\leq D(C)$.
negative cycle in G.

$\square$

Low Diameter Decomposition.

Categorization:

Low Diameter Decomposition.

Categorization:

In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n / \Delta$.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u, v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u, v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n / \Delta$.
Region: $\{u : d(u, v) \leq \ell\}$

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\Delta$.
Region: $\{u : d(u,v) \leq \ell\}$

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\Delta$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\Delta$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\Delta$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Low Diameter Decomposition.

Categorization:
In-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.
Out-Balls(Δ, v): $\{u : d(u,v) \leq \Delta\}$.

In-Light-Vertices: Size of In-Balls $\leq \frac{3}{4}|V|$.
In-Heavy: otherwise.

Out-Light and Out-Heavy similar.

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n / \Delta$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Out-Region growing....symmetric.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

Decomposition: analysis.

Graph is SCC of with *kappa* $= \kappa(G)$

In-Region-growing around v:

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n / \kappa$.
Region: $\{u : d(u, v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n / \kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n / \kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e) \log n/\kappa$

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e) \log n/\kappa$

Proof: Pr[edge (x,y) in different region.]

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e) \log n/\kappa$

Proof: Pr[edge (x,y) in different region.]
$\sum_r Pr[y \text{ not in } r | x \text{ in } r] Pr[x \text{ in } r]$.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20\log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e)\log n/\kappa$

Proof: Pr[edge (x,y) in different region.]
$\sum_r Pr[y \text{ not in } r | x \text{ in } r] Pr[x \text{ in } r]$.
Note: $\sum_r Pr[x \text{ in } r] \leq 1$.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e) \log n/\kappa$

Proof: Pr[edge (x,y) in different region.]
$\sum_r Pr[y \text{ not in } r | x \text{ in } r] Pr[x \text{ in } r]$.
Note: $\sum_r Pr[x \text{ in } r] \leq 1$.
Even in Texas probabilities ≤ 1.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e) \log n/\kappa$

Proof: Pr[edge (x,y) in different region.]
$\sum_r Pr[y$ not in $r|x$ in $r]Pr[x$ in $r]$.
Note: $\sum_r Pr[x$ in $r] \leq 1$.
Even in Texas probabilities ≤ 1.
$Pr[y \notin r|x \in r] \leq Pr[\ell \in [d(v(r),x), d(v(r),y]]] \leq pw(e)$

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e) \log n/\kappa$

Proof: Pr[edge (x,y) in different region.]
$\sum_r Pr[y$ not in $r|x$ in $r]Pr[x$ in $r]$.
Note: $\sum_r Pr[x$ in $r] \leq 1$.
Even in Texas probabilities ≤ 1.
$Pr[y \notin r | x \in r] \leq Pr[\ell \in [d(v(r),x), d(v(r),y)]] \leq pw(e)$ □

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e) \log n/\kappa$

Proof: Pr[edge (x,y) in different region.]
$\sum_r Pr[y$ not in $r|x$ in $r]Pr[x$ in $r]$.
Note: $\sum_r Pr[x$ in $r] \leq 1$.
Even in Texas probabilities ≤ 1.
$Pr[y \notin r | x \in r] \leq Pr[\ell \in [d(v(r),x), d(v(r),y]]] \leq pw(e)$ □

Implies that $O(\log n)$ expected edges in weight $\leq \kappa$ positive weight path.

Decomposition: analysis.

Graph is SCC of with $kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex for $\Delta = \kappa/4$.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: edge is between regions with probability
$pw(e) = 20w(e)\log n/\kappa$

Proof: Pr[edge (x,y) in different region.]
$\sum_r Pr[y$ not in $r|x$ in $r]Pr[x$ in $r]$.
Note: $\sum_r Pr[x$ in $r] \leq 1$.
Even in Texas probabilities ≤ 1.
$Pr[y \notin r|x \in r] \leq Pr[\ell \in [d(v(r),x), d(v(r),y]]] \leq pw(e)$ □

Implies that $O(\log n)$ expected edges in weight $\leq \kappa$ positive weight path.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
  Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
  Region: $\{u : d(u,v) \leq \ell\}$

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u, v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20\log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: Removing between edge regions w.h.p. leave
(i) SCC of size $\leq \frac{3}{4}|V|$.
(ii) or SCC's C of $\kappa(C) \leq \kappa/2$.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: Removing between edge regions w.h.p. leave
(i) SCC of size $\leq \frac{3}{4}|V|$.
(ii) or SCC's C of $\kappa(C) \leq \kappa/2$.

(i) Regions from light vertices, thus are small, w.h.p.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: Removing between edge regions w.h.p. leave
(i) SCC of size $\leq \frac{3}{4}|V|$.
(ii) or SCC's C of $\kappa(C) \leq \kappa/2$.

(i) Regions from light vertices, thus are small, w.h.p.
(ii) Remaining vertices have heavy in-balls and out-balls.

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: Removing between edge regions w.h.p. leave
(i) SCC of size $\leq \frac{3}{4}|V|$.
(ii) or SCC's C of $\kappa(C) \leq \kappa/2$.

(i) Regions from light vertices, thus are small, w.h.p.
(ii) Remaining vertices have heavy in-balls and out-balls.
In cycle of diameter $\leq \kappa/2$

Decomposition: analysis.

Graph is SCC with $\kappa = \kappa(G)$

In-Region-growing around v:
  Random geometric $\ell \in G(p)$, $p = 20 \log n/\kappa$.
  Region: $\{u : d(u,v) \leq \ell\}$

(1) Region-grow from light vertex.
(2) Remove region from light-vertices.
(3) Repeat.

Claim: Removing between edge regions w.h.p. leave
  (i) SCC of size $\leq \frac{3}{4}|V|$.
  (ii) or SCC's C of $\kappa(C) \leq \kappa/2$.

(i) Regions from light vertices, thus are small, w.h.p.
(ii) Remaining vertices have heavy in-balls and out-balls.
  In cycle of diameter $\leq \kappa/2$
    $\implies \leq \kappa/2$ edges in neg path.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's
or smaller diameter SCC's.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's
or smaller diameter SCC's.
And:

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's
or smaller diameter SCC's.
And:
Expected edges between components on short path $O(\log n)$.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's
or smaller diameter SCC's.
And:
Expected edges between components on short path $O(\log n)$.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's
or smaller diameter SCC's.
And:
Expected edges between components on short path $O(\log n)$.

Alg:
(1) Local price functions with recursion.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's
or smaller diameter SCC's.
And:
Expected edges between components on short path $O(\log n)$.

Alg:
(1) Local price functions with recursion.
(2) Dijkstra/Bellman with expected $O(\log n)$ iterations.

Quick Review.

Price functions: Find ϕ takes $-W$ edges to $-W/2$ edges. Repeat.
$O(\log nW)$ Subtlety is path length.

Small diameter SCC's with many hop negative path in $G_{\geq -1}$
$\implies$ have negative cycle in $G_{\geq -2}$.

Decompose graph by removing edges into smaller SCC's
or smaller diameter SCC's.
And:
Expected edges between components on short path $O(\log n)$.

Alg:
(1) Local price functions with recursion.
(2) Dijkstra/Bellman with expected $O(\log n)$ iterations.
Slightly subtle, expected requeing is $O(\log n)$ per vertex.