Seidel's Algo.

$\ell(e)=1$ $\forall e \in E$

Assume G is "undirected" and "unweighted" now.

- Faster Algo!
- But only computing dist[u,v] — forget prev[u,v].
  (can be done using randomization)

Faster Matrix Multiplication (CLRS 4.2)

Let A, B be nxn matrices: How fast can we compute $A \cdot B$?

$$\begin{bmatrix} \text{---} \\ A \\ \text{---} \end{bmatrix} \begin{bmatrix} | & B & | \end{bmatrix} = \begin{bmatrix} \bullet \\ A \cdot B \end{bmatrix}$$

Naively $O(n^3)$ time.

Strassen 1969: $O(n^{2.81})$

⋮

Alman-Williams 2021: $O(n^{2.3729})$ ($O(n^{2.376})$ in 1990)

Williams-Xu-Xu-Zhou 2024: $O(n^{2.371552})$

Main Idea, Given $G=(V,E)$, let $H=(V,E')$ s.t.

$(u,v) \in E'$ if u and v have distance ≤ 2 in G.

($(u,v) \in E$ or $(u,r),(r,v) \in E$ for some r)

G. H

Suppose we computed $d_H[u,v]$ for all $u,v \in V$. ($d_H[u,v]$: dist[u,v] in H.)
How can we use it to compute $d_G[u,v]$?

Lemma 1 $d_H[u,v] = \lceil d_G[u,v]/2 \rceil$. $\forall u,v \in V$ □

Lemma 2 If $d_G[u,v] = 2d_H[u,v]$,

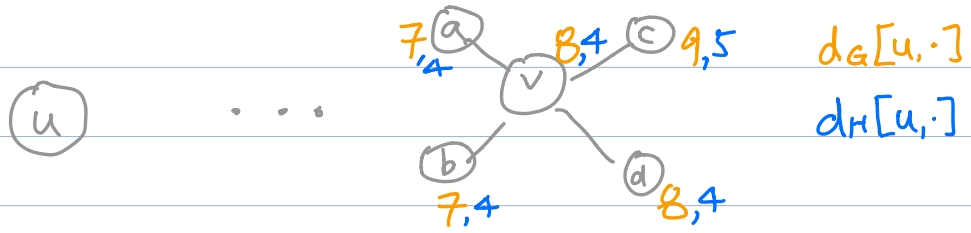
(i) $\forall w \in nbr(v)$, $d_H[u,w] \geq d_H[u,v]$

"neighborhood, $\{w : (w,v) \in E\}$"

(ii) $\sum_{w \in nbr(v)} d_H[u,w] \geq \deg(v) \cdot d_H[u,v]$

$= |nbr(v)|$

Proof



□

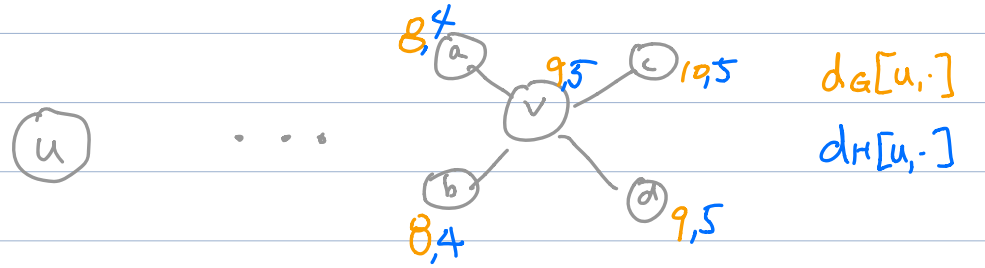
Lemma 3 If $d_G[u,v] = 2d_H[u,v] - 1$

(i) $\forall w \in nbr(v)$, $d_H[u,w] \leq d_H[u,v]$

and $\exists w \in nbr(v)$ s.t. $d_H[u,w] = d_H[u,v] - 1$

(ii) $\sum_{w \in nbr(v)} d_H[u,w] \leq \deg(v) \cdot d_H(u,v) - 1$

Proof



□

Seidel's Algo

How to compute H from G?

Let A_G (resp. A_H) be the adjacency matrix of G (resp. H).

Consider $A_G^2[u,v] = \sum_{w \in V} A_G[u,w] \cdot A_G[v,w]$

$= (\#$ of common neighbors of u and $v)$.

So $A_H[u,v] = \begin{cases} 0 & \text{if } u=v \\ 1 & \text{if } A_G[u,v]=1 \text{ or } A_G^2[u,v] \geq 1 \\ 0 & \text{otherwise} \end{cases}$

A_H can be computed in $O(n^{2.37\ldots})$ time!

upper bound on "diameter of G" — initially $n-1$ — $\max_{u,v \in V} \text{dist}[u,v]$

Seidel(G, k)

Compute H from G.

If $k=1$, return d_G s.t. $d_G[u,v]=1$ if $u \neq v$ and 0 o.w.

d_H = Seidel$(H, \lceil k/2 \rceil)$ Let $B = (d_H \cdot A_G)$

$n \times n$ matrices.

For u in V

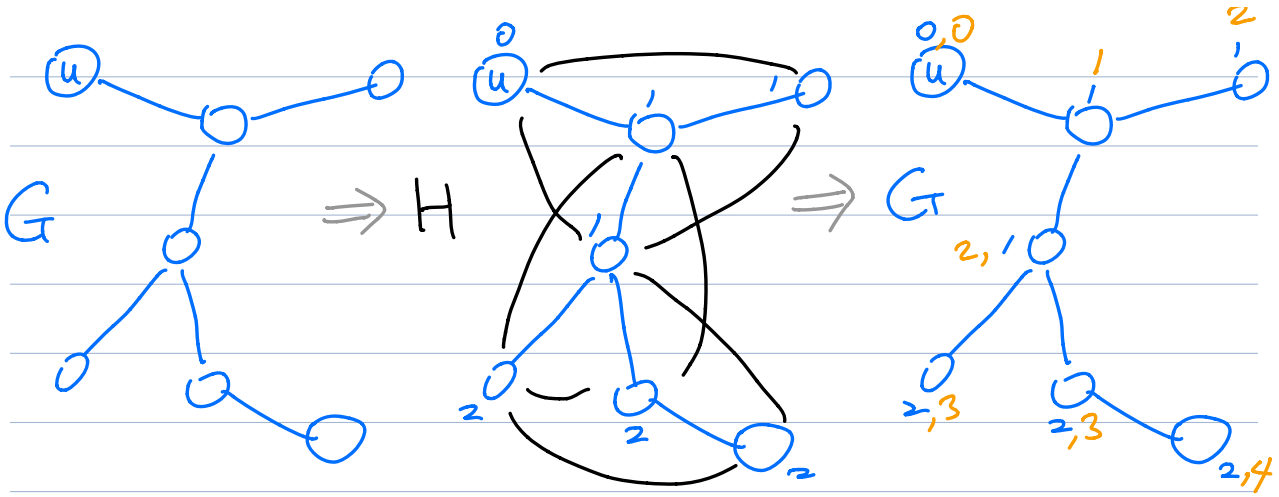
For v in V

If $u=v$, $d_G[u,v]=0$

Else $t = \begin{cases} 1 & \text{if } \boxed{\sum_{w \in \text{nbr}(v)} d_H[u,w]} \leq \deg(v) \cdot d_H(u,v) - 1 \\ 0 & \text{otherwise.} \end{cases}$ ← $B[u,v]$

$d_G[u,v] = 2d_H[u,v] - t.$

Return d_G.



Correctness, From Lemmas 1, 2, 3.

Running time, Each call to Seidel takes $O(n^{2.37\ldots})$ except the recursive call.

(# calls) = $O(\log n)$ because k is "halved" each time.

So total running time = $O(n^{2.37\ldots} \cdot \log n)$

But if we allow arbitrary (positive) lengths, it is conjectured that there is no $O(n^{2.99})$-time algo!

Single-Source Shortest Path

(Erickson Chap 8)

Single-Source Shortest Path (SSSP)

Input: Directed graph $G=(V,E)$ with edge weight $\ell: E \to \mathbb{R}$, and source $s \in V$.

Output: "shortest path" from s to v all $v \in V$.

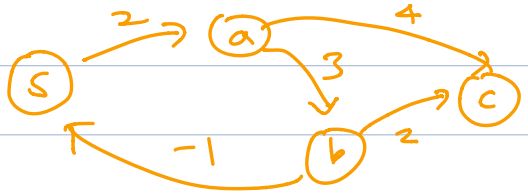
$(u,v) \in E$ denotes $u \to v$.

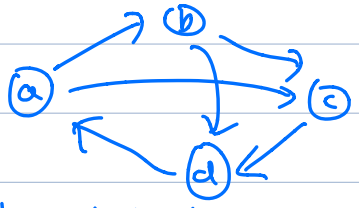
① "Shortest path"?

- walk: sequence of vtcs $w=(v_0,\dots,v_k)$ s.t. $(v_i,v_{i+1}) \in E$.
  (any vertex repetition allowed)
  Let "length of w", $\ell(w) := \sum_{i=0}^{k-1} \ell(v_i,v_{i+1})$.
- closed walk: walk with $v_0 = v_k$.
- (simple) cycle: a closed walk s.t. no vertex is repeated except $v_0 = v_k$.
- (simple) path: a walk s.t. no vertex is repeated.

(*) Assume $\nexists$ cycle $C=(v_0,\dots,v_k)$ s.t. $\ell(C) := \sum_{i=0}^{k-1} \ell(v_i,v_{i+1}) < 0$.

Obviously true if $\ell(e) \geq 0 \ \forall e \in E$, but there are other cases





walk: abdabc

closed walk: abdacda

cycle: abcda

path: abd

Lemma: Assuming (*), for any $u, v \in V$
(length of shortest walk from u to v) = (length of shortest path from u to v)

Pf: $\leq$) by definition

$\geq$) if walk contains cycle, "removing it" doesn't increase the length.

e.g., s a b s a c $\Rightarrow$ s a c $\square$

What if we allow negative cycles and find shortest simple path? NP-hard.

(Setting $\ell(e) = -1$ and asking shortest path from s to t is the same as asking longest (unweighted) path.)

② Undirected graph?

Replace each undirected edge (u,v) with two directed edges (u,v) and (v,u) w/ same length.

Then to ensure "no negative cycle" in new directed graph implies that "all edge length nonnegative" in original undirected graph.

③ How to "represent" or "output" s-t shortest path for all $t \in V$?

Today's algorithms maintain two "arrays" (path from s to v.)

dist[v]: length of current shortest s-v path.

prev[v]: In current shortest s-v path, the vertex right before v.

(**) At any point, if $dist[v] < \infty$,

$$\ell(\underbrace{s, \ldots, prev(prev(v)), prev(v), v}_{\text{current shortest path.}}) = dist[v].$$

So, at the end, outputting dist[] and prev[] is enough.

Initialization

$dist[s] = 0$, $prev[s] = \emptyset$.

$\forall v \in V \setminus \{s\}$, $dist[v] = \infty$, $prev[v] = \emptyset$.

Say $(u,v) \in E$ "tense" if $dist[u] + \ell(u,v) < dist[v]$.

If (u,v) is tense, we can do the following.

Relax (u,v)

$dist[v] = dist[u] + \ell(u,v)$.

$prev[v] = u$.

Dijkstra's Algo.

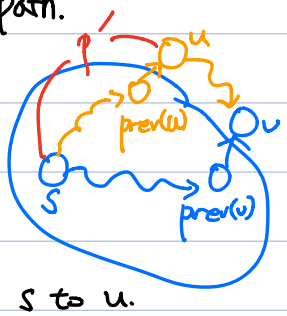
Only works $\ell(e) \geq 0 \ \forall e \in E$.

Intuition: At some point during algo, let $A \subseteq V$ be a set we are "sure";

Q: Can we add another vertex in $V \setminus A$ to A?

Lemma Let $A \subsetneq V$ s.t. $s \in A$. Suppose
(i) $\forall v \in A$, dist[v] is length of shortest s-v path.
(ii) $\forall (u,v) \in \partial^+ A \ (:= \{(u,v) \in E : u \in A, v \notin A\})$, (u,v) is not tense. ($dist[v] \leq dist[u] + \ell(u,v)$)
Let $v = \underset{v \in V \setminus A}{\operatorname{argmin}} dist[v]$. Then dist[v] is length of shortest s-v path.

Proof Let $p = (s, v_1, \ldots, v_k = v)$ be any s-v path.
Let u be the first vertex outside A,
and $p' = (s, \ldots, prev(u), u)$ be the prefix of p from s to u.



$\ell(p') = \ell(s, \ldots, prev(u)) + \ell(prev(u), u)$
$\geq dist[prev(u)] + \ell(prev(u), u)$ ($prev(u) \in A$)
$\geq dist[u] \geq dist[v]$. Since $\ell(p) \geq \ell(p')$, $\ell(p) \geq dist[v]$

($(prev(u), u)$ not tense; $dist[u] \geq dist[v]$ choice of v; $\ell(p) \geq \ell(p')$ every edge is nonnegative.)

□

Dijkstra(s)

dist[s]=0, dist[v]=∞ $\forall v \neq s$. prev[s] = ϕ.

A = ∅.

While $A \neq V$

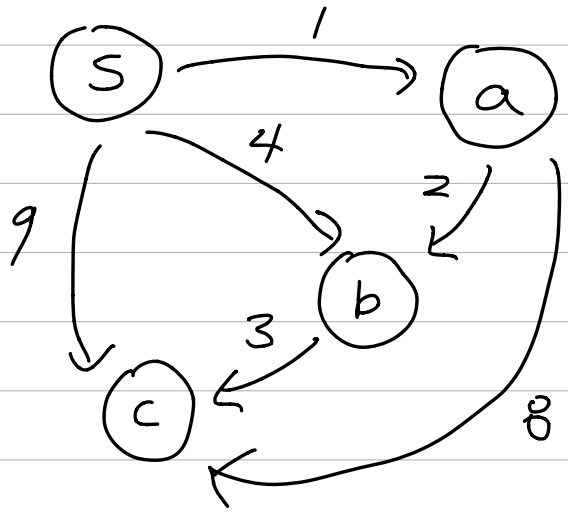
$v = \underset{v \in V \setminus A}{\operatorname{argmin}}\ dist[v]$

$A = A \cup \{v\}$.

For each $(v, w) \in E$

If $dist[w] > dist[v] + \ell(v,w)$

$dist[w] = dist[v] + \ell(v,w)$, $prev[w] = v$.



	A	A	A	A
	s	a	b	c
dist	0	1	3	6

Correctness

Claim: At the end of each while loop,

(i) $\forall v \in A$, dist[v] is length of shortest s-v path.

(ii) $\forall (u,v) \in \partial^+ A (:= \{(u,v) \in E : u \in A, v \notin A\})$, (u,v) is not tense. ($dist[v] \leq dist[u] + \ell(u,v)$).

Proof: Induction. First loop: $v = s$ and all $(s,u) \in E$ is not tense.

Assuming $1, \ldots, i^{th}$ loops are good, for $(i+1)^{th}$ loop, consider v chosen in the loop. By the lemma in the last page, it is true that dist[v] = (length of shortest s-v path). So (i) holds.

Whenever v is added to A, all edges from v become not tense. And these edges will never become tense again. So (ii) holds.

($\forall (v,w) \in E$, $dist[w] \leq dist[v] + \ell(v,w)$ at the end of while loop, but dist[v] is optimal, so will not change after.) □

Implementation of Dijkstra

Priority Queue (CLRS 6.5)

Maintain a set Q of elements with a key value.

Insert (Q, x, k)
Extract-Min(Q)
Decrease-Key(Q, x, k)

$\Big\}$ $O(\log |Q|) \leq O(\log n)$ $\quad (n=|V|, m=|E|)$

Dijkstra(s)

dist[s]=0, dist[v]=∞ $\forall v \neq s$. prev[s]=$\emptyset$.

$A = \emptyset$. $Q = \emptyset$, $\forall v \in V$, Insert$(Q, v,$ dist[v]$)$ — $O(n \log n)$

While $A \neq V$

~~$v = \underset{v \in V \setminus A}{\text{argmin}}$ dist[v]~~ $v =$ Extract-Min(Q) — $O(n \log n)$

$A = A \cup \{v\}$.

For each $(v, w) \in E$

If dist[w] > dist[v] + $\ell(v,w)$ — $O(m)$

dist[w] = dist[v] + $\ell(v,w)$, prev[w] = v.

Decrease-Key$(Q, w,$ dist[w]$)$ — $O(m \log n)$

Total running time: $O(m \log n)$.