CS6033 Lectures 12-13 Slides/Notes

Single-Source Shortest Paths; All-Pairs Shortest Paths (Notes, Ch 22, Ch 23)

By Prof. Yi-Jen Chiang
CSE Dept., Tandon School of Engineering
New York University

1

Single-Source Shortest Paths: | All-Pairs Shortest Paths

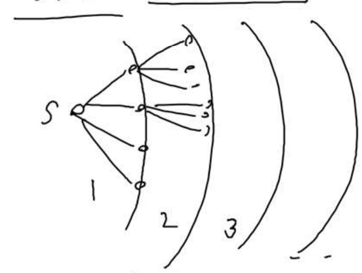
3 Algorithms

1. Dijkstra's Alg.
2. DAG
3. Bellman-Ford Alg.

Given a graph G and a vertex $s \in G$, find the shortest paths from s to all other vertices in G.

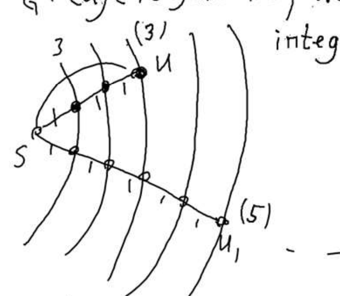
Consider unweighted graphs: $G=(V,E)$ where each edge has length 1.

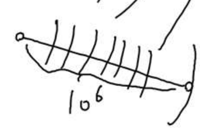
BFS BFS tree



wave-front

G: edge lengths are positive integers





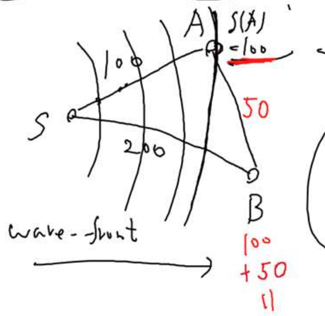
too slow!

2

Alarm clock idea

$\boxed{\delta(v): \text{shortest path distance from } s \text{ to } v}$

Alarm time(v): nearest time in the future that $\delta(v)$ needs to be updated.



$100 + 50 = 150$

Alarm(A) : 100.
Alarm(B) = 200

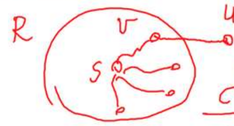
P.Q. Extract_Min(Q)
Q

(A)

min { Alarm(B), $\delta(A) + 50$ }

$\delta(u) = \delta(v) + \ell(u,v)$
since $\ell(u,v) > 0$
$\Rightarrow \delta(v) < \delta(u)$
if $v \notin R$.
then u can NOT be the vertex $\notin R$ with smallest $\delta()$ value

Region R : vertices currently reachable from S by the wave-front.



Let u be the next vertex to be included in R
i.e. u is the vertex NOT in R & $\delta(u)$ is the min among those vertices $\notin R$.

claim : $\delta(u) = \delta(v) + \ell(u,v)$ for some $v \in R$. $\ell(u,v) > 0$

pf : shortest path $s \rightsquigarrow u$ must be $s \rightsquigarrow v \to u$

3

(MY LOGIC) Proof: u --> v can be the shortest edge only if: v is in region R, which then makes the edge u,v as light edge

Conclusion :

Next vertex u to be included into region R is the vertex with min value of $\boxed{\delta(v) + \ell(v,u)}$ where $v \in R$.

Dijkstra's Alg. :

0. $R \leftarrow \{s\}$.
1. Put all vertices $v \in V$, $v \neq s$ into P. Q Q

   key(v) = ∞ except for neighbors of s (neighbor u of s: key(u) $\leftarrow \ell(s,u)$)
2. Repeat : $\to$ # : V

   $u \leftarrow$ Extract_Min(Q);

   $\delta(u) \leftarrow$ key(u) $R \leftarrow R \cup \{u\}$

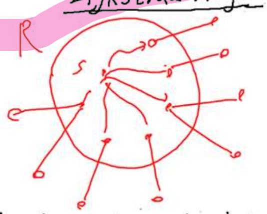
   for each neighbor w of u, $w \notin R$

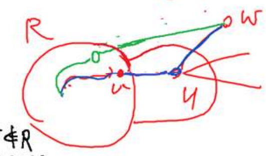
   { key(w) $\leftarrow$ min { key(w), $\delta(u) + \ell(u,w)$ } }

   $\to$ Decrease_Key(w, v)

   # : O(E)

   Until |R| = V





(for directed & undirected graphs)

$\boxed{\text{Edge lengths} > 0 \text{ (need not be integers)}}$

V ops Extract_Min()
O(E) " Decrease_key()

same as Prim's alg.
Fibonacci Heap : $O(E + V \log V)$ time

4

More Details: Decrease_key (w, k, Q) in Priority Queue Q

(Same as what we do in Prim's Algorithm)

③ Repeat

$u \leftarrow$ Extract_Min(Q) → #: V

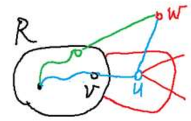
$\delta(u) \leftarrow key(u)$, $R \leftarrow R \cup \{u\}$, mark u

for each neighbor w of u, $w \notin R$

{ $key(w) \leftarrow \min\{key(w), \delta(u) + \ell(u,w)\}$

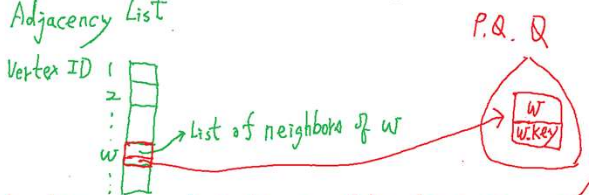
} → Decrease_key$(w, \ \ )$

Until $|R| = V$



* How do we access w in Q efficiently?

Sol: Adjacency List



+ Each vertex w has a ptr to its representative item in Q.

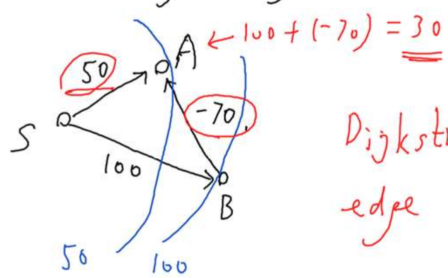
→ such ptr supports access to w in $O(1)$ time

in general, this is needed for Decrease_key(w, k, Q) & Delete(w, Q) for Q.

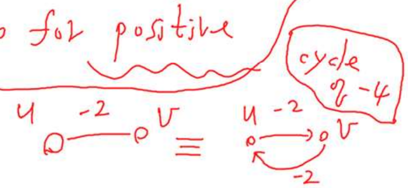
+ In Adjacency List, we can also mark u for "$u \in R$".

5

What happens to Dijkstra's Alg. if there are negative-weight edges? $\Rightarrow$ It can NOT work.

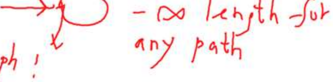


Dijkstra's Alg. only works for positive (≥ 0) edge lengths



What if there are negative edge weights?

1. If there is a cycle of negative weight, then there is NO shortest path!



2. Negative edge weight should NOT be in an undirected graph!

6

2 major important special cases to avoid negative cycles:

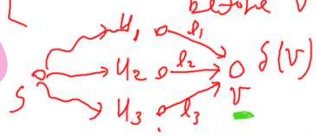
(1) Edge weights are all positive (≥ 0): Dijkstra's Alg. ✓
(directed & undirected graphs)

(2) DAG (directed acyclic graph): can have edge weights < 0, ≥ 0.

But there is No cycle $\Rightarrow$ No negative cycle.

Shortest paths on a DAG: Use D.P.

[Tips: For problems involving a DAG, always think about Topological Sort.
Review: If v has an in-coming edge (u,v) $u \circ\!\!\longrightarrow\!\!\circ v$, then u always goes before v in top. sort order.]



$$\boxed{\delta(v) = \min\{\delta(u) + \ell(u,v) \mid (u,v) \in E\}}$$

Compute $\delta(\,)$ in the top. sort order starting from s.
$\delta(s) = 0$.

$O(V+E)$ time.

7

In general directed graphs with negative edge weights (possibly negative cycles):

Bellman-Ford Alg.

$\delta(s) \leftarrow 0$. $\delta(v) \leftarrow \infty \;\; \forall v \neq s$.

$\boxed{O(VE) \text{ time}}$

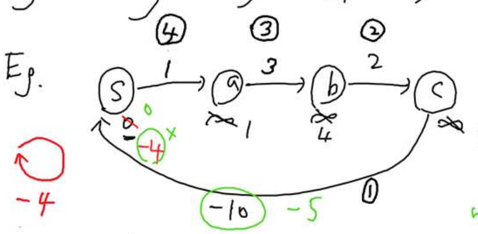
Repeat $V-1$ times
{ for each edge $(u,v) \in E$
  $\delta(v) \leftarrow \min\{\delta(v), \delta(u) + \ell(u,v)\}$
  (Relaxation)
}

$u \circ \xrightarrow{\ell(u,v)} \circ v$

Do one more iteration of Relaxation.
[for each edge $(u,v) \in E$
  $\delta(v) \leftarrow \min\{\delta(v), \delta(u) + \ell(u,v)\}$

If there is an update to any $\delta(\,)$ then negative cycle!
else $\delta(\,)$ is the shortest path length for each vertex.

Eg.



$6-5=1$
$\min\{0,1\}=0$

	#	s	a	b	c	
	0	0	∞	∞	∞	
$V-1$	1	0	1	∞	∞	
	2	0	1	4	∞	
	3	0	1	4	6	
V	4	-4	-3	4	6	X
	4	0	1	4	6	No update!

$V = 4$
$V-1 = 3$.

edges in any simple path $\leq V-1$.

8

All-Pairs Shortest Paths (Ch 23)

For each pair of vertices (i,j). find shortest path (lengths) $i \rightsquigarrow j$.

(ie All pairs)

1. Floyd-Warshall's Algorithm. D.P Assume: There is No negative cycle.

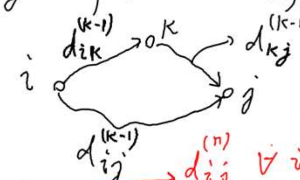
(If directed graph. there can have negative edges, but)

$G=(V,E)$

$|V|=n$

Def:

$d_{ij}^{(k)}$ = shortest-path length from vertex i to vertex j using $S_k = \{1, 2, \dots, k\}$ as intermediate vertices in the path.



$$\boxed{\begin{array}{ll} d_{ij}^{(k)} = \min\left\{ d_{ik}^{(k-1)} + d_{kj}^{(k-1)},\ d_{ij}^{(k-1)} \right\} \quad (*) & k \ge 1 \\ d_{ij}^{(0)} = \begin{cases} \ell(i,j) & \text{if } (i,j) \in E \\ \infty & \text{if } (i,j) \notin E. \end{cases} & k = 0. \end{array}}$$

$\boxed{O(n^3) \text{ time} = O(V^3).}$

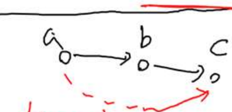
Answer: $\rightarrow d_{ij}^{(n)} \ \forall i,j \in V.$

For $k=1$ to n $\leftarrow$ $\boxed{i: 1 \rightarrow n,\ j: 1 \rightarrow n}$

(*).

9

Transitive closure:



transitive op.

Apply transitive op ∞ times.

Modify Floyd-Warshall Alg.

$$t_{ij}^{(k)} = \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)} \right) \vee \left(t_{ij}^{(k-1)} \right) \quad k \ge 1$$

$$t_{ij}^{(0)} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{else.} \end{cases} \quad k = 0$$

$O(V^3)$ time.

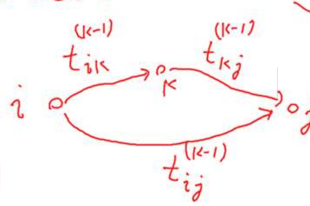
Def:

$$t_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ can reach vertex } j \\ 0 & \text{else.} \end{cases}$$

$\boxed{1: \text{true},\ 0: \text{false.}}$

Equivalent to all-pairs shortest path.

* DP. the cost function is a Boolean function



$\downarrow$

$\boxed{\text{For decision problem}}$

yes/no.

10

Johnson's Alg: Ideas: 1. Use Dijkstra's Alg. from each vertex as the source.

Dijkstra's Alg: $O(E + V\log V)$ | Do it for each vertex:

$\Rightarrow O(V(E + V\log V)) = \boxed{O(VE + V^2 \log V)}$

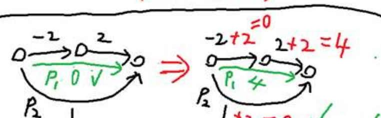
Cf: Floyd-Warshall Alg.: $O(V^3)$ | $E = O(V^2)$ Sparse graph $E = o(V^2)$

Better than $O(V^3)$ for sparse graphs.

Idea 2. Re-weight edges so that ① $w'(e) \geq 0 \quad \forall e \in E$.
($w \longrightarrow w'$) ② Paths $P_1, P_2 : u \rightsquigarrow v$. $w(P_1) < w(P_2)$
$\Longleftrightarrow w'(P_1) < w'(P_2)$

shortest path remains to be a shortest path

e.g.



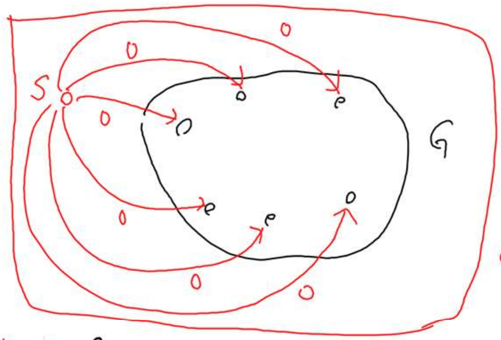
After re-weighting, P_1 is no longer the shortest!

11

Alg:

1. Add a new vertex s and new edge (s, u) with length 0 for each vertex u. $O(V)$ time

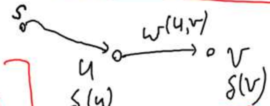
(This does NOT produce any new cycle.)



Total: $O(VE + V^2 \log V)$ time

2. Run Bellman-Ford alg. on new graph G' from s. $O(VE)$ time

[If negative cycle is found $\Rightarrow$ return "No shortest paths"
Else each vertex v gets $\delta(v)$, i.e. shortest-path length $s \rightsquigarrow v$ in G']



$\underbrace{\delta(v) \leq \delta(u) + w(u,v)}$

$\underbrace{w(u,v) + \boxed{\delta(u) - \delta(v)} \geq 0}_{w'(u,v)}$

$w'(u,v) = w(u,v) + \delta(u) - \delta(v) \geq 0$

3. For each edge (u,v), re-weight (u,v)
$w'(u,v) \leftarrow w(u,v) + \delta(u) - \delta(v) \ (\geq 0)$.

$O(VE + V^2 \log V)$ time [4. Run Dijkstra's alg. on G' from each $v \in G$ as source.
5. Update shortest-path distances

12

proving correctness of Johnson's algorithm

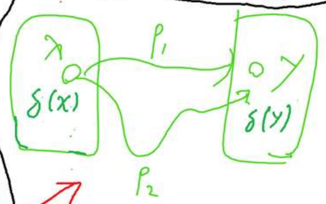
Path $x \rightsquigarrow y$: $p = (x_0 = x, x_1, x_2, x_3, \ldots, x_t = y)$.

$x_0 = x \to x_1 \to x_2 \to \cdots \to x_{t-1} \to x_t = y$

$$\boxed{w'(p)} = [w(x_0,x_1) + \boxed{\delta(x_0)} - \cancel{\delta(x_1)}] + [w(x_1,x_2) + \cancel{\delta(x_1)} - \cancel{\delta(x_2)}] + [w(x_2,x_3) + \cancel{\delta(x_2)} - \cancel{\delta(x_3)}] + \cdots + [w(x_{t-1},x_t) + \cancel{\delta(x_{t-1})} \boxed{- \delta(x_t)}]$$

$$= w(p) + \delta(x_0) - \delta(x_t)$$

$$= w(p) + \boxed{\delta(x) - \delta(y)}$$



$w'(p_1) = w(p_1) + \boxed{\delta(x) - \delta(y)}$

$w'(p_2) = w(p_2) + \boxed{\delta(x) - \delta(y)}$

$w(p_1) < w(p_2) \iff w'(p_1) < w'(p_2)$

The alg. gives correct shortest paths !!

$$\boxed{w(p) = w'(p) - \boxed{(\delta(x) - \delta(y))}}$$

Adjustment of shortest-path lengths needed in step 5.

13