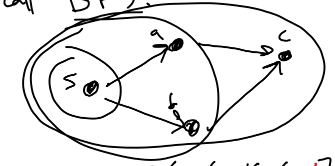
Graphs 4 (Shortest Paths)

Thursday, April 2, 2020 4:20 PM

Shortest Paths

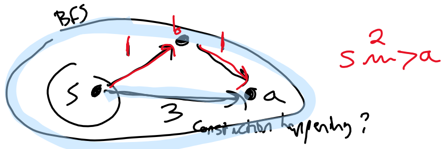
Recall BFS:



Queue: [~~s~~ ~~a~~ ~~b~~ ~~c~~ ~~c~~]

— BFS finds shortest paths (from Node s) in unweighted.

Today: weighted graphs.



Starting point:

recursive description of problem.

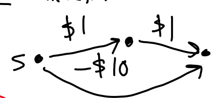
dist(u) be distance of shortest path from s to u.

$$\text{dist}(u) = \begin{cases} 0 & \text{is } u = s \\ \min\limits_{\substack{v \xrightarrow{w} u \\ v \in in(u)}} & w + \text{dist}(v) \end{cases}$$

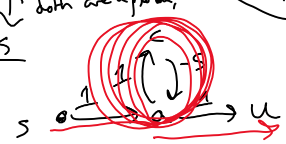
in(v): nodes with edge to v

recursive but not a recurrence (in terms of smaller problems).

— negative weight? both are a problem!



— cycles



sp: s ~> u?

no shortest path! $-\infty$

— Special case: DAG.

Original recursive definition is a recurrence relation.

dist(u) depends on dist(v) for each $v \in in(u)$.

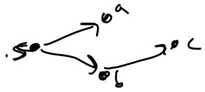
DAGs have an topo. ordering $\prec$.

$v \to u \Rightarrow v \prec u$ Analyze: $O(|V|+|E|)$

---

s ~> u sp from s to u

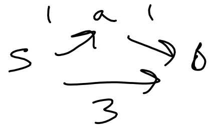
SSSP = Single Source Shortest path from s to every node



Later All pairs

— Bellman-Ford: "hop" counter

$\text{dist}_s(i, u)$: distance of sp from s to u, taking at most i hops (∞ if unreachable in i edges)



dist(0, s) = 0

dist(0, a) = ∞

$dist(1,b) = 3$
$dist(2,b) = 2$

$dist(\underline{i},\underline{v})$:
- if $i = 0$: 0 if $v = s$, ∞ otherwise
- else: $\min\left(\min_{u \in in(v)} w(u \to v) + dist(i-1,u),\ dist(i-1,v) \right)$

$dist(i,v)$ depends on
~~$dist(i-1,v)$~~;
$dist(i-1,u)$ for all $u \in in(v)$

- \# of subproblems: $(n \times n)$ ~~n^3~~
- ~~time~~ for each subproblem (n)

overall running time: $O(m \cdot n)$

Lemma: If there is a nonnegative cycle,

$s \rightsquigarrow a \rightsquigarrow a \rightsquigarrow u$ (cycle)

then there is a shorter (or equal) path w/ no cycle

$s \overset{w_1}{\rightsquigarrow} a \overset{w_2}{\rightsquigarrow} u$ (u_1, u_2) $\Rightarrow$ $s \rightsquigarrow a \rightsquigarrow u$

$\Downarrow$ $a \overset{c}{\rightsquigarrow} a$ $c \geq 0$ $\Downarrow$

$\text{length}(w_1 + c + w_2) \geq \text{length}(w_1 + w_2)$

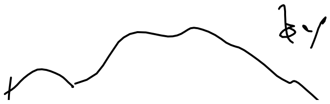
Lemma: Any path from $s \rightsquigarrow u$, with no cycles takes n or fewer hops. (regardless of m)
Proof: pigeonhole.

$s \xrightarrow{1} a \xrightarrow{1} u$ $dist(2,u) = 2$
0 ($a \rightleftarrows c$) 0 $|s \to a \to c \to a \to c \to a \to u| \; 2$

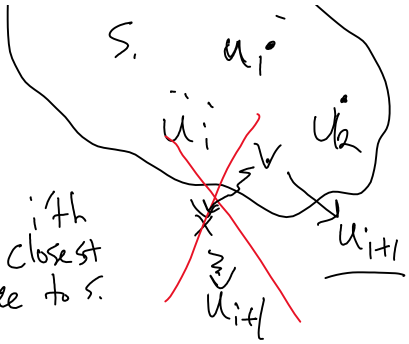
Dijkstra: Assume no negative edges.

Performance vs B-F

Main idea: find the shortest paths for each node, in increasing order by shortest path length.



$(i+1)$th closest



u_i : i'th closest node to s.

Lemma: the (i+1)'th ... node to s, is exactly one hop away (adjacent) to one of i closest nodes.

Proof: Suppose $s \overset{w_1}{\rightsquigarrow} x \overset{w_2}{\rightsquigarrow} u_{i+1}$ is shortest path from s to u_{i+1}, and $x \notin$ first i closest nodes.

X is closer than u_{i+1}

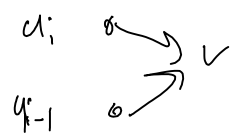
$w_1 + w_2$: dist to u_{i+1}

w_1 : dist to x, ... so x is the (i+1)'th closest node.

Recursive: dist(i) : distance to (i)'th closest node.

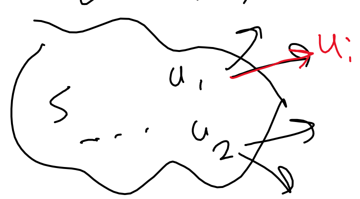
$$dist(i+1) = \min_{0 \le j \le i} \left(\min_{u_j \to v} \left(dist(j) + w(u_j, v) \right) \right)$$

u_i is i'th closest node.

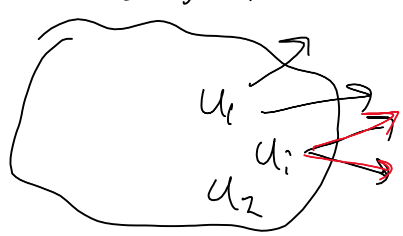


running time ... $n \times (n \times m)$

dist(i)



dist(i+1)



- remove $v \to u_i$
- and add all edges $u_i \to X$

- first approach:
  - Store all $(dist(j) + w(u_j, v))$ values in a BST.

for T elements in the data structure:

	insert	delete	findMin
BST	log T	log T	log T
fibheap		O(1)	

- In each iteration, take the minimum (log T) to find u_{i+1} ...

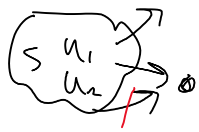
— to and u_{i+1}

— removing all $v \to u_{i+1}$ $\quad$ $O(m)$ ins/del,

— and add all $u_{i+1} \to v$ $\quad$ $O(m \cdot \log T)$

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

— Second: only store one edge for each dst



limits size of the BST to n

$O(n + m \log n)$

— Third: Amortized data structure (~~better~~ overall perf. even if each ins/del/find worst case is same)

(fibo heap)

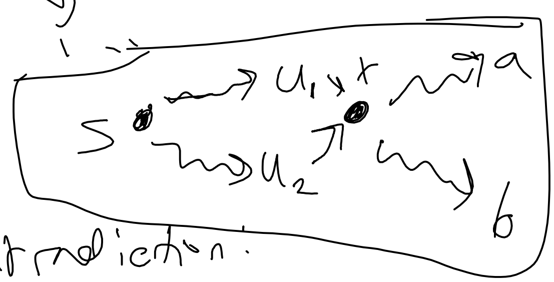
Reach $O(m + n \log n)$ when no neg. edges.

— Shortest Path Tree

Lemma: There is a tree rooted at s that contains a shortest path from s to each node u.

tree
— only one parent
— no cycles.

Proof:



Suppose for contradiction:

SP $s \rightsquigarrow a$ goes through $s \rightsquigarrow u_1 \to x \rightsquigarrow a$

$s \rightsquigarrow b$ but $s \rightsquigarrow u_2 \to x \rightsquigarrow b$

Suppose $\underbrace{s \rightsquigarrow u_1 \to x}_{w_1} \underbrace{\rightsquigarrow a}_{w}$

$\underbrace{s \rightsquigarrow u_2 \to x}_{w_2} \underbrace{\rightsquigarrow b}_{w}$

$s \rightsquigarrow u_2 \to x \to b$

Wlog, suppose $w_1 \le w_2$.

Then $s \rightsquigarrow u_1 \to x \to b$ is a shorter (or eq) path to b. $w_1 + w \le w_2 + w$

Because of this, can represent shortest paths with a parent array

parent[i] is parent of node i in the shortest path tree

$s \to u_1 \to u_2 \to u_3$

$s \to b$

Negative Cycle detection.

$s \xrightarrow{1} a \xrightarrow{3} c$

$a \xrightarrow{1} b$, $b \xrightarrow{-2} a$

$s \xrightarrow{1} a \xrightarrow{3} c$ 4

$s \xrightarrow{1} a \xrightarrow{1} b \xrightarrow{-2} a \xrightarrow{3}$ 3

$s \to a \to b \to a \to b \to a \to c$ 2

n

n+1

BF $\quad dist(i, v)$: sp to v taking i or fewer edges

If no neg cycles, then $0 \le i \le n$ suff.

i	s	a	b	c
0	0	∞	∞	∞
1	0	1	∞	∞
2	0	1	2	4
3	0	0	2	4
n 4	0	0	1	3
n+1		-1	1	3
		-1	0	2
		-2	0	2
		-2	-1	1
			-1	1

Lemma: Iff $\exists u$, s.t. $dist(n, u) > dist(n+1, u)$ then there is a neg. cycle reachable from s.

Case I $\Rightarrow$

w no. of fewer hops

$s \overset{w}{\rightsquigarrow} u$

$s \overset{w'}{\rightsquigarrow} u$ n+1 or fewer

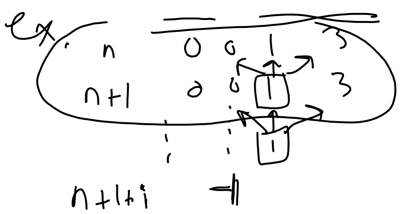
$w' < w$ then $s \overset{w'}{\rightsquigarrow} u$ has $n+1$ hops.

By pigeon, has a cycle.

By earlier lemma, positive cycles are never on SP, zero cycles can be removed, so this cycle must be negative.

Case II Need to show this can't happen.

ex.



$\{dist(i+1, u)\}_u$ depends only on $\{dist(i, u)\}_u$

$\forall u,\ dist(i+1, u) = dist(i, u)$

$\Rightarrow \forall u,\ dist(i+2, u) = dist(i+1, u)$