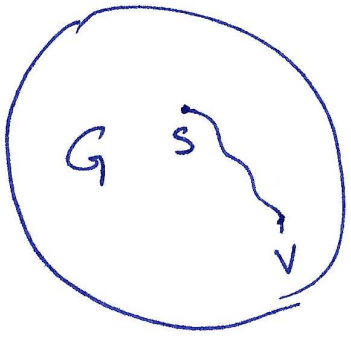
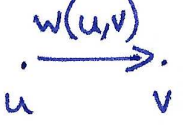
SSSP: Single Source Shortest Paths



$d(s,v)$ is the shortest path distance.

Given G and a source s, compute all $d(s,v)$. $\forall v$

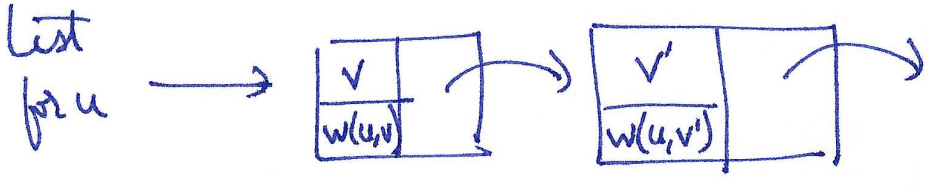
Directed, weighted graphs



$w(u,v)$ and $w(v,u)$ may be different.

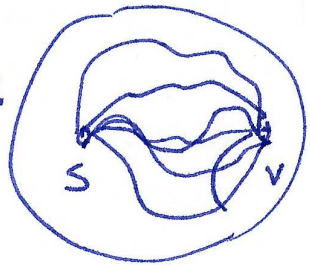
If G is unweighted ($w(u,v) = 0, 1$),
BFS solves SSSP in $\Theta(m+n)$ time.

G represented as adjacency list



The length of a path $s, u_1, u_2, \ldots, u_k$ ($u_0 = s$)

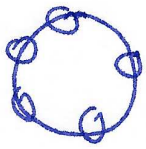
$$= \sum_{i=0}^{k-1} w(u_i, u_{i+1})$$

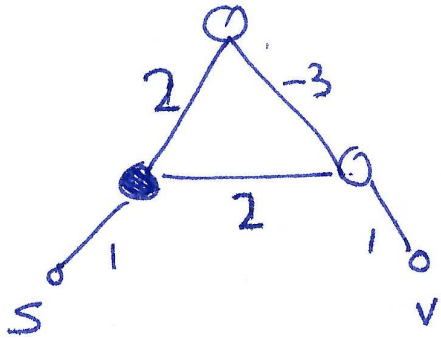


$d(s,v)$ = length of shortest path = min. length of a path from s to v

When weights are negative, $d(s,v)$ might ~~be~~ NOT be defined.

When there is a negative weight CYCLE, $d(s,v)$ may NOT be defined.





SSSP Algorithm maintains two arrays. $V = \{1, 2, \ldots, n\}$.

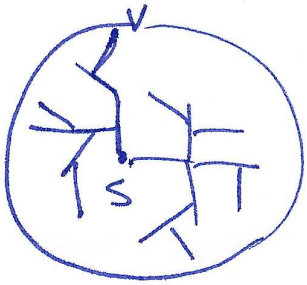
(1) dist[v] ← estimate of $d(s,v)$

(2) pred[v] ← a vertex preceding v on a shortest path from s

INVARIANTS (Hold at all times)

(1) $dist[v] \geq d(s,v)$

(2) v, pred[v], pred[pred[v]], ..., s is a path (in reverse) from s to v of length dist[v].

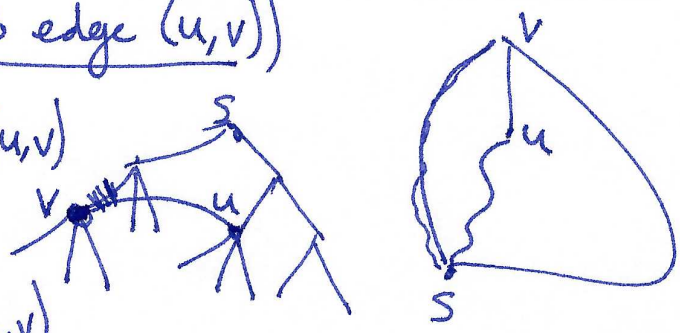


Initialization

$dist[s] = 0$ $\quad \forall\, v \neq s,\ dist[v] = \infty$

$\forall\, v$ $\quad pred[v] = NULL$

Relax(u,v) (with a ptr to edge (u,v))



(1) If $dist[v] > dist[u] + w(u,v)$

$O(1)$ (a) $pred[v] = u$

(b) $dist[v] = dist[u] + w(u,v)$

Relax is "safe". All invariants are maintained.

Bellman-Ford

#vertices, #edges

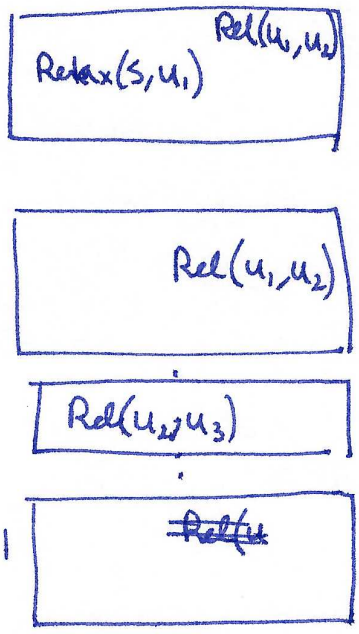
(1) Initialize pred, dist arrays $\quad \Theta(n)$

(2) For $i = 0$ to $n-1$: $\quad \Theta(mn)$

(a) For every edge (u,v): $\Theta(m)$

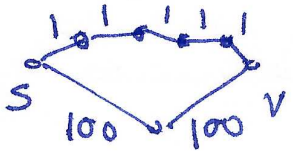
Relax(u,v) $\quad \Theta(1)$

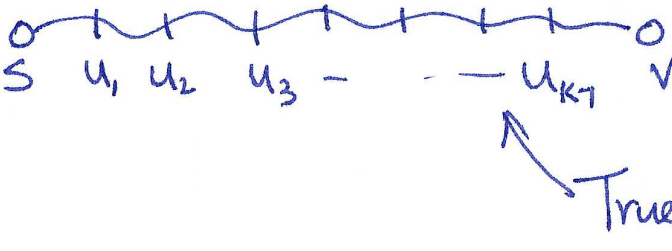
$\Theta(mn)$ running time



Works even when weights are negative.

It can find a negative weight cycle. (Exercise)





Suppose we called Relax(s,u_1), Relax(u_1,u_2), Relax(u_2,u_3) -- = Relax(u_{k-1},v) in <u>this</u> order.

Then dist$[v] = d(s,v)$.

edges in any shortest path $\leq n-1$

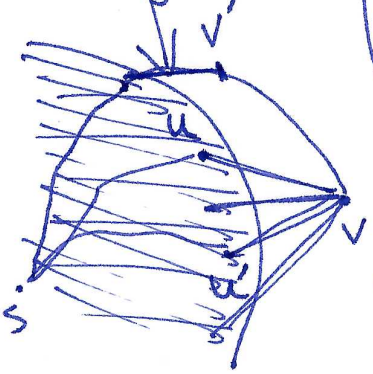
Relax(s,u_1) — Other Relax calls — Relax(u_1,u_2) — Other Relax — Relax(u_2,u_3) — Relax(u_3,u_4) — — — Relax(u_{k-1},v)

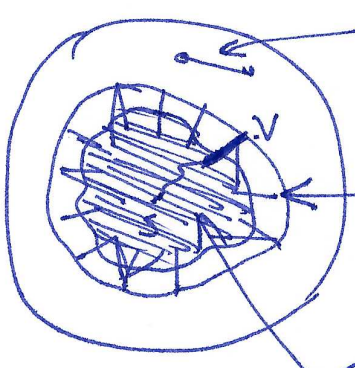
dist$[v] = d(s,v)$

Dijkstra's Algorithm

Assumes all weights are <u>NON-NEGATIVE</u>.

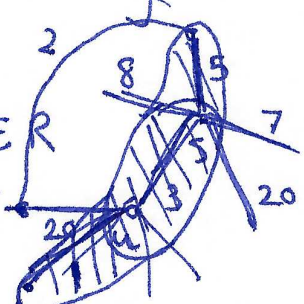
Find least weight edge leaving S



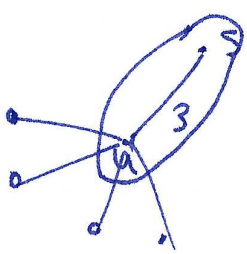


No point in calling Relax

FRONTIER



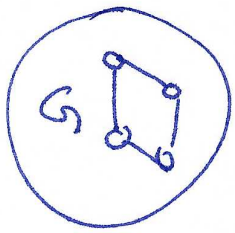
S: all shortest path distances in S have been found



For every $u \in S$:

for every edge (u,v), ~~call Relax(u,v)~~

consider $dist[u] + w(u,v)$



Find the min. value of $dist[u] + w(u,v)$. Call relax on THAT edge.

Dijkstra(G, s)

(1) Init. dist and pred arrays. $\Theta(n)$

(2) S is an empty list

(3) Q is a priority queue (min heap) storing vertices with dist[v] as key.

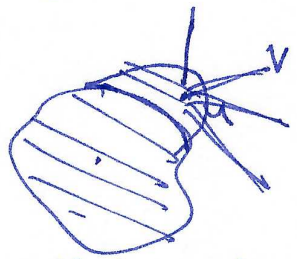
(4) while Q is non-empty:

(a) u = ExtractMin(Q) ← $O(\log n)$ called n times

(b) Append u to S (and $v \notin S$)

(c) For every $v \in N(u)$: ← traversal of adj. list

Call Relax(u,v) and decrease-key(v) in Q.
update key(Q, v, dist[v]) — Called m times, $O(\log n)$



Total running time is $\Theta((m+n)\log n)$

Using fancier heap, $\Theta(m + n\log n)$

changed because of Relax

Summary of SSSP

G is unweighted. BFS $\Theta(m+n)$

G is weighted.

If there are negative weight cycles, shortest paths are not defined.

Relax operation, distance invariants.

"Safe" operation, preserves invariants

An SSSP path algorithm can be thought of as a sequence of Relax operations

Bellman-Ford: Can handle negative weights. Can discover negative weight cycles. $\Theta(mn)$

Dijkstra: Assume non-negative weights $\Theta((m+n)\log n)$

Classic use of minheap/priority queue