TODAY
- Finish shortest paths
  - Dijkstra analysis
  - Bidirectional Dijkstra, A^*
  - Bellman-Ford
- Start greedy algorithms
  - Shortest job first
  - Class scheduling
  - Gale-Shapley

Dijkstra (generic version)

Initialize $D[v] = \infty$ for all v
$D[s] = 0$. Mark all nodes unfinished

$O(V)$ while there are unfinished nodes $O(V)$

Let v = unfinished node w/ min dist ($D[v]$)

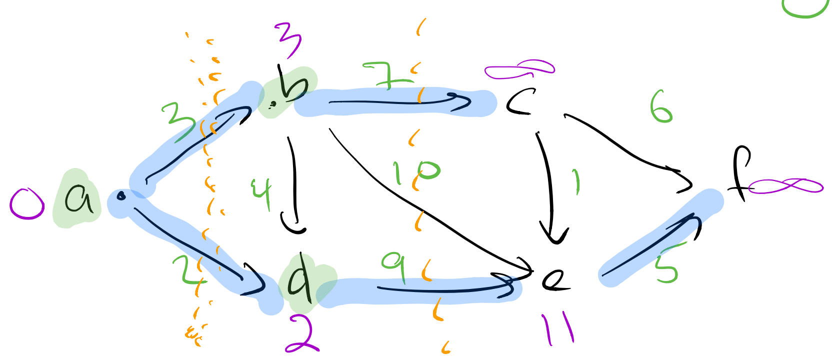
$O(E)$ For edges $v \to u$

If $D[u] > D[v] + \ell(v \to u)$: tense edge

$D[u] = D[v] + \ell(v \to u)$ relax

mark v as finished

$O(E + V^2) = O(V^2)$



$u_1 = a$
$d_1 = 0$
$u_2 = d$
$d_2 = 2$
$u_3 = b$
$d_3 = 3$

Let u_i be vertex extracted at i-th iteration
d_i be its distance at that time.

Lemma $d_i \le d_j$ for $i < j$, if no negative edges

Proof $d_i \le d_{i+1}$ for all i

At ith iteration $d(u_{i+1}) \ge d(u_i)$

relax edges from u_i
$d(u_{i+1}) = d(u_i) + \ell(u_i \to u_{i+1})$

Lemma 2 No weight of finished nodes is ever changed (no neg edges)

Proof if u_i is ~~finished~~ finished at it i
in a later iteration $j>i$ $d(u_j) \geq d_i$
any edge $u_j \to u_i$ cannot be tense
$$d(u_j) \geq u_i$$
$$d(u_j) + l(j \to i) \geq d_i$$

d_i is the final distance of node u_i
nodes processed (marked finished) in order of final distance

Lemma 3 For any path $s \to v_1 \to v_2 \to v_3 \dots \to v_n$
final $\boxed{d(v_n)} \leq l(\text{path}) = l(s \to v_1) + l(v_1 \to v_2) + \dots$

Proof by induction. Assume true for paths shorter than n

$$d(v_{n-1}) \leq l(\text{path } s \dots v_{n-1})$$

when v_{n-1} was extracted, $d(v_n) \leq d(v_{n-1}) + l(v_{n-1} - v_n)$

$$d(v_n) \leq l(\text{path } s \dots v_{n-1}) + l(v_{n-1} \to v_n)$$

$\therefore$ shortest path $s \to v$ is $\geq d(v)$
$\leq d(v)$

$\therefore$ Dijkstra computes shortest paths

Priority Queue

- Insert (key, value) — key: dist, value: graph node
- ExtractMin () $\to$ min key, value
- DecreaseKey (newkey, value) $\to$ adjust the key dist of a node

Dijkstra (with PQ)

Init $D[v] = \infty$, $D[s] = 0$
Insert(Q, (0, s))
while Q not empty:

$V \cdot \log V$ $v = \text{ExtractMin}(Q)$ $O(\log V)$

    if $v = t$ stop

for edges $v \to u$

  if $D[u] > D[v] + \ell(v \to u)$

    $D[u] = D[v] + \ell(v \to u)$

    if $u \in Q$

$E \cdot \log V$ Decrease key $(D[u], u)$

    else

$E \cdot \log V$ Insert $(D[u], u)$

binary heap → EM, DK, I $O(\log V)$

$O((E+V) \log V)$

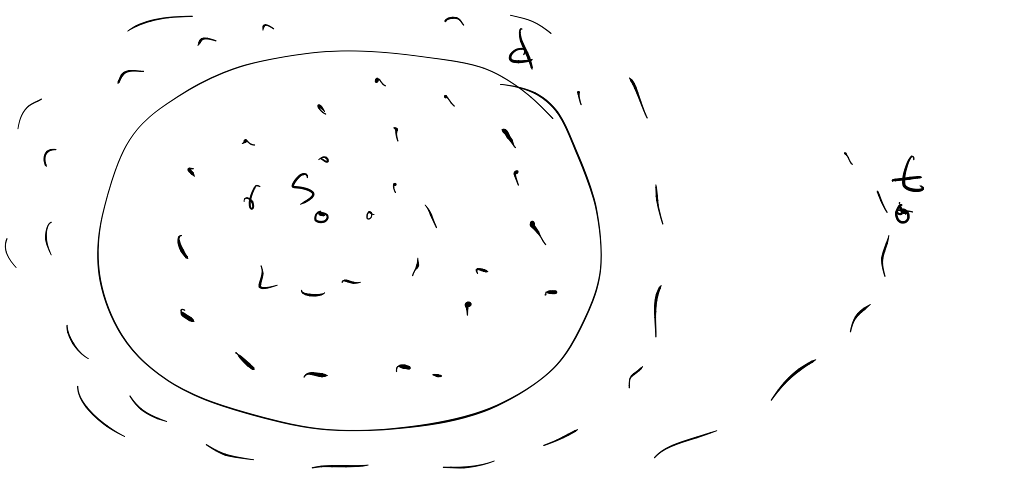
fibonacci heap → I, DK is amortized $O(1)$

EM is $O(\log N)$

$O(E + V \log V)$ fastest SSSP on weighted graphs with cycles and no neg edges

Dijkstra gives us shortest path to all nodes from one source

Shortest path from s to t



Bidirectional Dijkstra

all nodes at distance d from s and t

stop after $d = \ell(s \rightsquigarrow t)/2$

→ store $D[v] = \infty$ $D[s] = 0$ fw dist from s

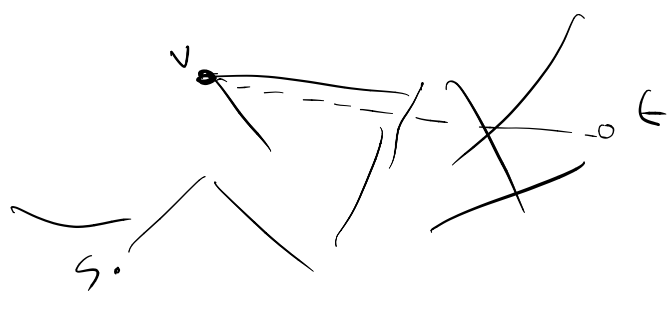
$D^{-1}[v] = \infty$ $D^{-1}[t] = 0$ b/w dist from t

Insert $(s, 0)$
Insert $(t^{-1}, 0)$
while
Extract Min (both backwards & forward nodes)
if both forward & b/w has been extracted for v then $SP[s \rightsquigarrow t]$ is $D[v] + D^{-1}[v]$

A^* same as dijkstra

$\min D[v] + h(v)$ ← heuristic function

$h(v) \leq d(v \rightsquigarrow t)$ condition



Bellman-Ford

while there are tense edges
relax tense edges

Init $D[v] = \infty$, $D[s] = 0$

Flag = true

while Flag: ⇐

    Flag = False

    for each edge $u \to v$
        if tense
            relax $(u \to v)$ (update $D[v]$)
            Flag = True ⇐

<u>Lemma</u> after i iterations in B-F

$D[v]$ = length of shortest path from $s \to v$ with $\leq i$ hops

<u>Proof</u> assume true for $i-1$

SP $s \to v$ in i hops ℓ'

$s \to \underline{u} \to v$ in $i-1$ hops $= \ell$

after $i-1$ iterations $D(u)$ is ℓ

after i iterations $D(v)$ is $\ell + e(u \to v) = \ell'$

All paths are at most $V-1$ hops

After $V-1$ iterations $D(v)$ is shortest path $s \to v$

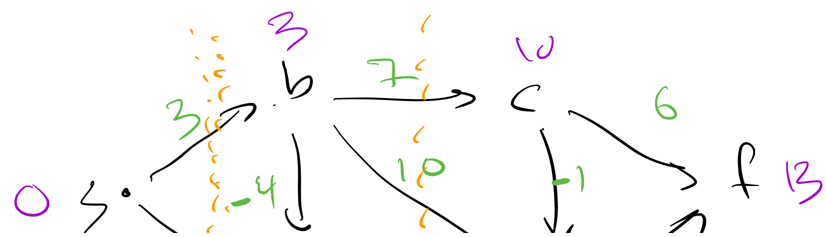
Bellman-Ford

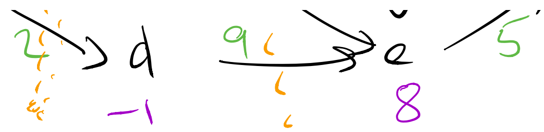
Init $D[v]=\infty$, $D[s]=0$

V→ for $i=1$ to $V-1$

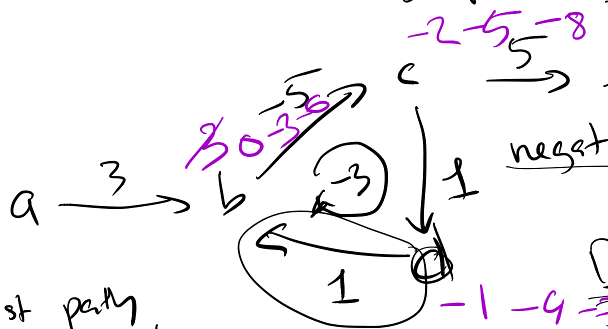
E→ for each edge $u \to v$
    if tense
        relax $(u \to v)$ (update $D[v]$)

<u>$O(VE)$</u> slower than $O(E+V\log V)$





B-F works on all graphs
- weighted
- cycles
- negative edges



negative weight cycle

$D(d) + e(d \to b) \le D(b)$

least cost path

$a \to b \to c \to e \qquad 3 - 5 + 5 = 3$

least cost walk

$a \xrightarrow{3} b \xrightarrow{-5} c \xrightarrow{1} d \xrightarrow{1} b \xrightarrow{-5} c \xrightarrow{5} e = 0$

$a \to bcd \to bcd \to b \to c \to e = -3$

...

there is no least cost walk

BF explores walks, not paths

if no -ve cycle → shortest paths = shortest walks

Bellman-Ford

```
Init D[v]=∞, D[s]=0
for i=1 to V-1
   for each edge u→v
      if tense
         relax (u→v)   (update D[v])

// check for -ve cycle
for each edge (u→v)
   if tense
      return "negative cycle"
```

Shortest path algs

1. Unweighted graph — BFS $O(V+E)$
2. Weighted dag — DAG-DP $O(V+E)$
3. No -ve edges — Dijkstra $O(E+V\log V)$
4. Otherwise — B-F $O(VE)$
   error if -ve cycle

$$\boxed{\begin{aligned} \text{Dist}[s] &= 0 \\ \text{Dist}[v] &= \min_{u\to v} D[u] + \ell(u\to v) \end{aligned}}$$

true for all graphs

if cycles no order of evaluation?

$\text{Dist}[v,i]$ = shortest path after $\leq i$ hops

$$\text{Dist}[s,i] = 0$$

$$\text{Dist}[v,0] = \infty \quad \text{for } v \neq s$$

$$\text{Dist}[v,i] = \min\left(\min_{u\to v} D[u,i-1] + \ell(u\to v),\ D[v,i-1]\right)$$

$\text{Dist}[\cdot, |V|-1]$ = shortest path distance

DP formulation → Bellman-Ford algorithm