TODAY
- Finish shortest paths
- Dijkstra analysis
- Bidirectional Dijkstra
- Bellman-Ford
- Shortest greedy algorithms
- Shortest job first
- Class scheduling
- Gale-Shapley

Dijkstra (generic version)
Initialize $D[u] = \infty$ for all $u$
$D[S] = 0$. Mark all nodes unfinished

(a) While there are unfinished nodes $O(v)$
Let $v = \text{ unfinished node } w / \min \text{ dist } (D[v])$
For edges $v \rightarrow y$
If $D[u] > D[v] + c(v \rightarrow u)$: tense edge
$D[u] = D[v] + c(v \rightarrow u)$
mark $v$ as finished

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

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

Lemma $d_i \leq d_j$ for $i < j$

Proof $d_i \leq d_{i+1}$ for all $i$
At $i$-th iteration $d(u_{i+1}) \geq d(u_i)$
relax edges from $u_i$
$d(u_{i+1}) = d(u_i) + c(u_{i+1})$

Lemma 2 No weight of finished nodes is
ever changed (no new edges)
Proof. If \( u_i \) is unprocessed, \( d_i \geq d(u_i) \) in a later iteration, \( j = 1 \) and \( d(u_i) \geq d_i \); any edge \( u_j \rightarrow u_i \) cannot be tense.

\[ d(u_{j+1}) = u_i; \]
\[ d(u_{j+1}) + l(j+1) \geq u_i; \]

\( d_i \) is the final distance of node \( u_i \) not processed (marked finished) in order by final distance.

**Lemma 3**. For any path \( s = v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_n \) with final \( d(v_n) \),

\[ \text{final } d(v_n) \leq \ell(\text{path}) = \ell(s \rightarrow v_1) + \ell(v_1 \rightarrow v_2) + \ldots \]

Proof by induction. Assume true for paths shorter than \( n \).

\[ d(v_{n-1}) \leq \ell(\text{path } s \ldots v_{n-1}) \]

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

\[ d(v_n) \leq \ell(\text{path } s \ldots v_{n-1}) + \ell(v_{n-1} \rightarrow v_n) \]  

is shortest path \( s \rightarrow v \) is \( \geq d(v) \)

\[ \leq d(v) \]

Dijkstra computes shortest paths

**Priority Queue**

- \( \text{dist graph node} \)

  - Insert (key, value)
  - Extract Min () \( \Rightarrow \text{min key, value} \)
  - Decrease Key (new key, value) \( \Rightarrow \text{adjust the dist of a node} \)

**Dijkstra** (with PQ)

- \( \text{init } D[s] = \infty, D[S] = 0 \)
- Insert \( (Q, (0, s)) \)
- while \( Q \) not empty:

\[ \]
\[ V \cdot \log V \]

\[ v = \text{Extract Min}(Q) \quad O(\log V) \]

\[ \text{if } v = t \text{ stop} \]

\[ \text{for edges } v \rightarrow u \]

\[ \text{if } D[v] \geq D[u] + E(v \rightarrow u) \]

\[ D[u] = D[v] + E(v \rightarrow u) \]

\[ \text{if } u \notin Q \text{ Decrease key } (D[u], u) \]

\[ \text{else Insert } (D[u], u) \]

\[ E \cdot \log V \]

\[ E \cdot \log V \]

binary heap \rightarrow EM, DK, I \quad O(C \log V) \]

\[ O((E+V) \log V) \]

fibonacci heap \rightarrow I, DK is amortized \( O(1) \)

EM is \( O(C \log N) \)

\[ O(E + V \log V) \]

fastest SSSP on weighted graphs with \( n \) edges

Dijkstra gives us shortest path from any node to all nodes

shortest path from \( s \) to \( t \)

Bidirectional Dijkstra
all nodes at distance \( d \) from \( s \) and \( t \\
\) stop after \( d = \ell(s \rightarrow t) / 2 \)

store \( D(s) = \infty \), \( D(s,t) = 0 \) for dist from

\( D(s) = \infty \), \( D(s,t) = 0 \) btw dist from

\( v \)

\( \in \)

\( \in \)

\( \in \)

while

Extract Min ( both backwards or forward node )

if both forward - b/w has been extracted

for \( u \) then \( SP(u \rightarrow t) = D(s,t) + D(u) \)

\( A^* \) same as dijkstra - heuristic function

\( \min \ D(s,v) + h(v) \)

\( h(v) \leq \ell(v \rightarrow t) \) - condition

Bellman - Ford

while there are tense edges

relax tense edge

\( Init \ D(s) = \infty \), \( D(s,t) = 0 \)
Flag = true
while Flag =<
    Flag = False
    for each edge u→v if tense
        relax (u→v) (update D(u))
        Flag = True =<

Lemma after i iterations in B-F
D(u) = length of shortest path from s to v
with ≤ i hops

Proof assume true for i-1
so s→u in i hops ≤
s→u→v in i-1 hops =

after i-1 iterations D(u) is ≤
after i iterations D(u) is ≤ + e(u→v)

All paths are at most V-1 hops
After V-1 iterations D(u) is shortest path
s→v

Bellman-Ford

Init D(s) = ∞, D(s) = 0
for i = 1 to V-1
for each edge u→v
if tense
    relax (u→v) (update D(u))
O(VE) slower than O(E+VlogV)
B-F works on all graphs - weighted edges
-2 -5 -8
Graph that contains a negative weight cycle
least cost path
\[ a \to b \to c \to e \]
\[ D(c) + e(c \to b) \leq D(b) \]
least cost walk
\[ a \to b \to c \to d \to b \to c \to e = 0 \]
\[ 3 -5 1 1 -5 5 \]
\[ a \to b \to c \to d \to b \to c \to e = -3 \]
there is no least cost walk
B-F explores walks, not paths
if no -ve cycle \( \Rightarrow \) shortest path = shortest walk
Bellman-Ford

Init \( D(v) = \infty \), \( D(s) = 0 \)
for \( i = 1 \) to \( U-1 \)
for each edge \( u \to v \)
if tense
relax \( (u \to v) \) (update \( D(v) \))

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

Shortest path algo
1. Unweighted graph
2. Weighted DAG
3. No -ve edges
4. Other mix

\[ \text{Dist} \{s, 0\} = 0 \]
\[ \text{Dist} \{v, i\} = \min_{u \rightarrow v} \text{Dist} \{u, i-1\} + e(u \rightarrow v) \]

true if all graphs no order of evaluation?

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

\[ \text{Dist} \{s, i\} = 0 \]
\[ \text{Dist} \{v, 0\} = \infty \text{ for } v \neq s \]
\[ \text{Dist} \{v, i\} = \min_{u \rightarrow v} \left\{ \text{Dist} \{u, i-1\} + e(u \rightarrow v) \right\} \]

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

DP formulation \rightarrow Bellman-Ford algorithm