- Structure of Shortest Paths
- Dijkstra's algorithm
- Bellman-Ford Algorithm

- Given a graph $G$, edge $(u, v)$ has length $l(u, v)$
  find the shortest path from $s$ to $t$.
  \[
  \text{length of path} = \sum_{(u, v) \in \text{path}} l(u, v)
  \]

- Recall: Shortest path on DAG.
  Shortest path using BFS
  \[
  s \rightarrow a \rightarrow b \rightarrow t, \text{ len: } \]

- Basic property of shortest path

  Let $S, u_1, u_2, \ldots, u_k, t$ is a shortest path from $s$ to $t$.
  Claim: $S, u_1, u_2, \ldots, u_i$ is also a shortest path from $s$ to $u_i$.

  \[
  S \quad u_1 \quad u_2 \quad u_3 \quad \ldots \quad U_k \quad t
  \]

  Proof: If there is a shorter path $s, v_1, v_2, \ldots, v_k, u_i$
  then $s, v_1, \ldots, v_k, u_i, u_{i+1}, \ldots, U_k, t$ will also be
  shorter than $s, u_1, u_2, \ldots, U_k, t$, but that is impossible.

- Dynamic Programming:
  \[
  d(u) : \text{distance from } s \text{ to } u.
  \]
  \[
  \text{length of the shortest path}
  \]

  \[
  d(u) = \min_{(u, v) \in F} \{ d(u) + l(u, v) \} \quad \text{(Same as the recursion for DAG)}
  \]

  Problem: If graph has cycles, it's hard to determine ordering.

- Dijkstra's Algorithm
main idea: compute \( d(v) \) in increasing order.

\[ \text{initialize } d(s) = 0, \text{ mark } s \text{ as visited} \]

\[ \text{for any edge } (s,u), \quad h(u) = d(s,u) \quad \text{(Prev}(u) = s) \]

\[ \text{for any other vertex } h(u) = +\infty \]

\[ \text{for } i = 2 \text{ to } n \]

let \( u \) to be the vertex with minimum \( h(u) \) among unvisited vertices

\[ d(u) = h(u), \text{ mark } u \text{ visited} \]

for all edges \((u,v)\)

if \( d(u) + (u,v) < h(v) \)

\[ h(v) = d(u) + (u,v) \quad \text{(Prev}(v) = u) \]

Correctness: use induction

Property: at any iteration; \( d(u) \) for any visited \( u \) is the correct distance from \( s \) to \( u \); \( h(v) \) for any unvisited \( u \) is length of shortest path from \( s \) to \( v \) but last step uses a visited vertex.
Base case: only $s$ is visited, property is clearly true.

Induction: suppose property is true at the beginning of iteration $i$, it is also true after the iteration.

Need: (1) for the vertex $u$, we pick, $d(u)$ is the length of shortest path.
(2) $h(u)$ updated correctly.

1. Assume shortest path from $s$ to $u$ is $s, v_1, v_2, \ldots, v_r, u$.
   - If $v_r$ is visited, length $\geq h(u)$.
   - If $v_r$ is not visited, let $v_j$ be the first unvisited vertex.
     \[ \text{length}(s, v_1, v_2, \ldots, v_{j-1}, v_j) \geq h(v_j) \geq h(u) \]

2. After visit $u$.
   \[ h(u) = \min_{u: \text{visited}} d(u) + (u, u) \]

- Running time:
  - naive: $O(n^2)$
  - heap: $O((n+m) \log n)$
  - Fibonacci heap: $O(n \log n + m)$

- Negative edges:
  - Dijkstra fails

- Negative cycle
- Negative cycle
  shortest path does not exist
  \( S \rightarrow a \rightarrow b \rightarrow S \rightarrow a \rightarrow b \rightarrow \ldots \)

- Claim: If \( G \) does not have a negative cycle, shortest path will not visit any vertex twice.

Proof: consider path \( S, u_1, u_2, \ldots, t \) in \( S \).

  if \( u_i = u_j \)

  because \( u_i, u_{i+1}, \ldots, u_j \) is a cycle

  length \( (u_i, u_{i+1}, \ldots, u_j) \geq 0 \)

  remove the cycle and \( S, \ldots, u_i, u_{j+1}, u_{j+2}, \ldots, t \)

  has length \( \leq \) length of original path

\[ \Rightarrow \text{number of steps of shortest path} \leq n-1 \]

- Bellman-Ford

  \( d(u, i) = \) shortest path from \( S \) to \( u \) using at most \( i \) steps.

  initialize \( d(S, 0) = 0 \), \( d(u, 0) = +\infty \)

  \[ d(u, i) = \min \{ d(u, i-1), \min_{(v, u) \in E} d(v, i-1) + (v, u) \} \]

Claim: if the graph has no negative cycles, \( d(u, n-1) \) is the distance to \( u \).

Running time: \( O(nm) \)

Claim: \( d(u, n) \neq d(u, n-1) \) for some \( u \), there is a negative cycle.

if there is a negative cycle, then \( \exists u \), \( d(u, n) \neq d(u, n-1) \)