CSE 431/531: Analysis of Algorithms (Summer 2023)
Single Source Shortest Path

Chen Xu

July 25, 2023

Single Source Shortest Path Problem

SSSP with non-negative weight

Input: edge weighted directed graph $G = (V, E, W_{\geq 0})$, $s \in V$

Output: $d[u], u \in V \setminus \{s\}$: length of shortest path from s to u.
$\pi[u], u \in V \setminus \{s\}$: parent of u in the shortest path tree.

Recall Prim's algorithm

Prim MST

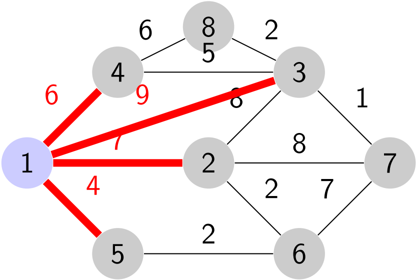
$s \leftarrow$ arbitrary vertex in G
$S \leftarrow \emptyset$, $d(s) \leftarrow 0$ and $d[v] \leftarrow \infty$ for every $v \in V \setminus \{s\}$
$Q \leftarrow$ empty queue, for each $v \in V$: Q.insert(v, $d[v]$)
while $S \neq V$ **do**
    $u \leftarrow Q$.extract_min()
    $S \leftarrow S \cup \{u\}$
    for each $v \in V \setminus S$ such that $(u, v) \in E$ **do**
        if $w(u, v) < d[v]$ **then**
            $d[v] \leftarrow w(u, v)$, Q.decrease_key(v, $d[v]$)
            $\pi[v] \leftarrow u$
return $\{(u, \pi[u]) | u \in V \setminus \{s\}\}$

Dijkstra's algorithm

Dijkstra SSSP

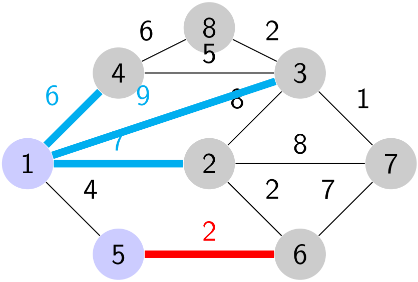
$S \leftarrow \emptyset$, $d(s) \leftarrow 0$ and $d[v] \leftarrow \infty$ for every $v \in V \setminus \{s\}$
$Q \leftarrow$ empty queue, for each $v \in V$: Q.insert(v, $d[v]$)
while $S \neq V$ **do**
    $u \leftarrow Q$.extract_min()
    $S \leftarrow S \cup \{u\}$
    for each $v \in V \setminus S$ such that $(u, v) \in E$ **do**
        if $d[u] + w(u, v) < d[v]$ **then**
            $d[v] \leftarrow d[u] + w(u, v)$, Q.decrease_key(v, $d[v]$)
            $\pi[v] \leftarrow u$
return (π, d)

Example



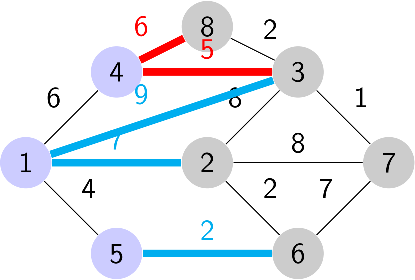
	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞

Example



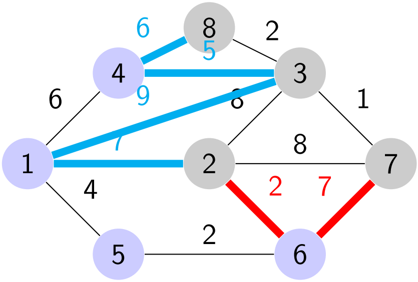
	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞
5	7	9	$\boxed{6}$	-	6	∞	∞

Example



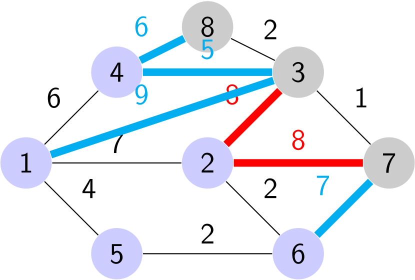
	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞
5	7	9	$\boxed{6}$	-	6	∞	∞
4	7	9	-	-	$\boxed{6}$	∞	12

Example



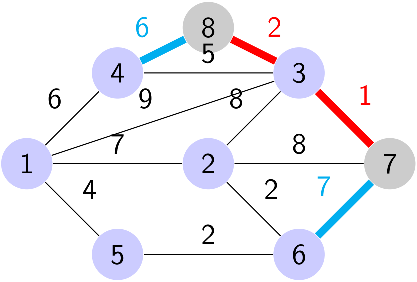
	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞
5	7	9	$\boxed{6}$	-	6	∞	∞
4	7	9	-	-	$\boxed{6}$	∞	12
6	$\boxed{7}$	9	-	-	-	13	12

Example



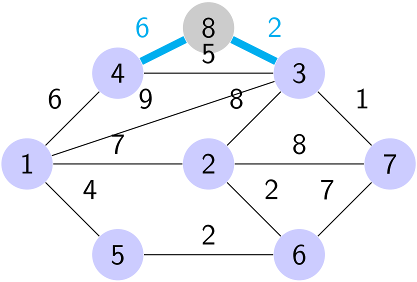
	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞
5	7	9	$\boxed{6}$	-	6	∞	∞
4	7	9	-	-	$\boxed{6}$	∞	12
6	$\boxed{7}$	9	-	-	-	13	12
2	-	$\boxed{9}$	-	-	-	13	12

Example



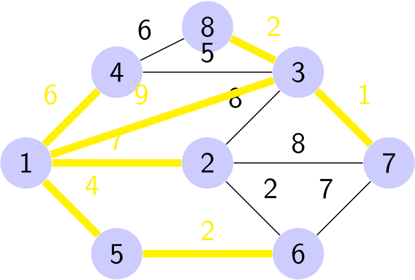
	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞
5	7	9	$\boxed{6}$	-	6	∞	∞
4	7	9	-	-	$\boxed{6}$	∞	12
6	$\boxed{7}$	9	-	-	-	13	12
2	-	$\boxed{9}$	-	-	-	13	12
3	-	-	-	-	-	$\boxed{10}$	11

Example



	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞
5	7	9	$\boxed{6}$	-	6	∞	∞
4	7	9	-	-	$\boxed{6}$	∞	12
6	$\boxed{7}$	9	-	-	-	13	12
2	-	$\boxed{9}$	-	-	-	13	12
3	-	-	-	-	-	$\boxed{10}$	11
7	-	-	-	-	-	-	$\boxed{11}$

Example



	2	3	4	5	6	7	8
1	7	9	6	$\boxed{4}$	∞	∞	∞
5	7	9	$\boxed{6}$	-	6	∞	∞
4	7	9	-	-	$\boxed{6}$	∞	12
6	$\boxed{7}$	9	-	-	-	13	12
2	-	$\boxed{9}$	-	-	-	13	12
3	-	-	-	-	-	$\boxed{10}$	11
7	-	-	-	-	-	-	$\boxed{11}$
8	-	-	-	-	-	-	-

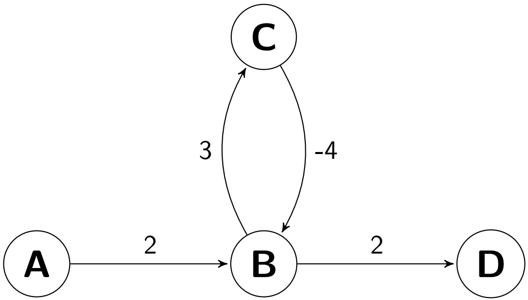
Single Source Shortest Path with negative weight

SSSP the weight maybe negative

Input: edge weighted directed graph $G = (V, E, W)$, $s \in V$
Output: $d[u], u \in V \setminus \{s\}$: length of shortest path from s to u.

- Dijkstra algorithm will fail. Exercise: give an example.

Negative cycle



- A negative cycle is a cycle which has a negative total weight.
- Taking a negative cycle ∞ times, we get $-\infty$ distance from A to D.
- A simple path is a path that does not contain a cycle, every edge cannot be used twice.
- If we restrict the shortest path to be simple, we get a distance 3 from A to D.

DP approach

- We can apply dynamic programming to solve this problem.
- Define our subproblem as $DP[i][v]$ which is asking the shortest path from source to v containing **at most** i edges.
- We have the recursive structure below:

$$DP[i][v] = \begin{cases} 0 & i = 0, v = s \\ \infty & i = 0, v \neq s \\ \min \begin{cases} DP[i-1][v] \\ \min\limits_{u:(u,v)\in E} (DP[i-1][u] + w(u,v)) \end{cases} & i > 0 \end{cases}$$

DP SSSP algorithm

DP SSSP

Initialize $DP[0][s] \leftarrow 0$ and $DP[0][v] \leftarrow \infty$ for any $v \in V \setminus \{s\}$
for $i \leftarrow 1$ to $n - 1$ **do**
    Copy $DP[i-1][*] \rightarrow DP[i][*]$
    for each $(u, v) \in E$ **do**
        if $DP[i-1][u] + w(u, v) < DP[i][v]$ **then**
            $DP[i][v] \leftarrow DP[i-1][u] + w(u, v)$
return $(DP[n-1][v])_{v \in V}$

- Observe that $DP[i][*]$ only depends on $DP[i-1][*]$. We can just use 2 rows.
- In fact, through further optimization we can just use 1 row.
- The issue is that the change of $DP[i][*]$ in the ith iteration might affect the other updates in the same iteration. However this change will take effect in the next iteration anyways.

DP SSSP algorithm

DP SSSP

1: Initialize $DP[0][s] \leftarrow 0$ and $DP[0][v] \leftarrow \infty$ for any $v \in V \setminus \{s\}$
2: **for** $i \leftarrow 1$ to $n-1$ **do**
3: Copy $DP[i-1][*] \rightarrow DP[i][*]$
4: **for** each $(u, v) \in E$ **do**
5: **if** $DP[i-1][u] + w(u, v) < DP[i][v]$ **then**
6: $DP[i][v] \leftarrow DP[i-1][u] + w(u, v)$
7: **return** $(DP[n-1][v])_{v \in V}$

- We know that for a graph with n vertices, if a path is longer than $n-1$, it must contain a cycle. So we just need n iterations at most to cover all cases.
- If by the end of n iterations we still see changes we know that there must exist a negative cycle. Because if there exists one, the weight will keep decreasing.

Bellman-Ford algorithm

- We can reduce the dimension of the DP table to an array. Here is the final updated Bellman-Ford algorithm.
- Once we have no changes in the DP array we can terminate the algorithm. Running time is $O(nm)$.

Bellman-Ford SSSP

```
Initialize DP[s] ← 0 and DP[v] ← ∞ for any v ∈ V \ {s}
for i ← 1 to n do
    updated ← false
    for each (u, v) ∈ E do
        if DP[u] + w(u, v) < DP[v] then
            DP[v] ← DP[u] + w(u, v), π[v] ← u
            updated ← true
    if not updated then
        return DP
Output: "Negative cycle exists"
```