COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

Greedy Algorithms

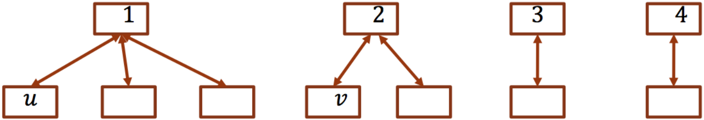
Algorithm

`Kruskal's Algorithm(G)`
- $S \leftarrow E$; $T \leftarrow \{\}$
- While the edge set T does not connect all the vertices
    - Let e be the minimum weight edge in the set S
    - If e does not create a cycle in T
        - $T \leftarrow T \cup \{e\}$
    - $S \leftarrow S \setminus \{e\}$

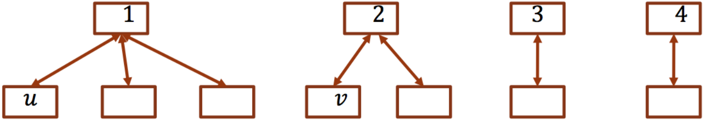
Algorithm

`Kruskal's Algorithm(G)`
- $S \leftarrow E$; $T \leftarrow \{\}$
- While the edge set T does not connect all the vertices
    - *//Note that $G' = (V, T)$ contains dicsonnected components*
    - Let e be the minimum weight edge in the set S
    - ~~If e does not create a cycle in T~~
    - If u and v are in different components of G'
        - $T \leftarrow T \cup \{e\}$
    - $S \leftarrow S \setminus \{e\}$

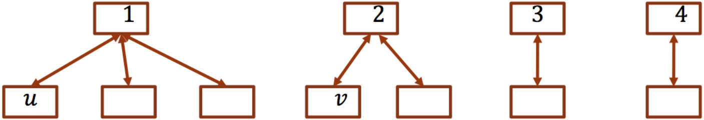
- Union-Find: Used for storing partition of a set of elements. The following two operations are supported:
  1. *Find*(v): Find the partition to which the element v belongs.
  2. *Union*(u, v): Merge the partition to which u belongs with the partition to which v belongs.
- Consider the following data structure.



- Suppose we start from a full partition (i.e., each partition contains one element).
- How much time does the following operation take:
  - *Find*(v):
  - *Union*(u, v):



- Suppose we start from a full partition (i.e., each partition contains one element).
- How much time does the following operation take:
  - $Find(v)$: $O(1)$
  - $Union(u, v)$:



- Suppose we start from a full partition (i.e., each partition contains one element).
- How much time does the following operation take:
  - $Find(v)$: $O(1)$
  - $Union(u, v)$:
    - Claim: Performing k union operations takes $O(k \log k)$ time in the worst case when starting from a full partition.
    - Proof sketch: For any element u, every time its pointer needs to be changed, the size of the partition that it belongs to at least doubles in size. This means that the pointer for u cannot change more than $O(\log k)$ times.

- Kruskal's algorithm using Union-Find.

Algorithm

`Kruskal's Algorithm(G)`

- $S \leftarrow E$; $T \leftarrow \{\}$
- While the edge set T does not connect all the vertices
  - *//Note that* $G' = (V, T)$ *contains dicsonnected components*
  - Let e be the minimum weight edge in the set S
  - ~~If e does not create a cycle in T~~
  - ~~If u and v are in different components of G'~~
  - If $(Find(u) \neq Find(v))$
    - $T \leftarrow T \cup \{e\}$
    - $Union(u, v)$
  - $S \leftarrow S \setminus \{e\}$

- What is the running time of the above algorithm?

- Kruskal's algorithm using Union-Find.

Algorithm

Kruskal's Algorithm(G)
- $S \leftarrow E$; $T \leftarrow \{\}$
- While the edge set T does not connect all the vertices
  - *//Note that $G' = (V, T)$ contains dicsonnected components*
  - Let e be the minimum weight edge in the set S
  - ~~If e does not create a cycle in T~~
  - ~~If u and v are in different components of G'~~
  - If $(Find(u) \neq Find(v))$
    - $T \leftarrow T \cup \{e\}$
    - $Union(u, v)$
  - $S \leftarrow S \setminus \{e\}$

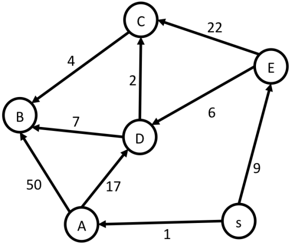
- What is the running time of the above algorithm? $O(|E| \cdot \log |V|)$

- Path length: Let $G = (V, E)$ be a weighted directed graph. Given a path in G, the length of a path is defined to be the sum of lengths of the edges in the path.
- Shortest path: The shortest path from u to v is the path with minimum length.

- Path length: Let $G = (V, E)$ be a weighted directed graph. Given a path in G, the length of a path is defined to be the sum of lengths of the edges in the path.
- Shortest path: The shortest path from u to v is the path with minimum length.

Problem

Single source shortest path: Given a weighted, directed graph $G = (V, E)$ with positive edge weights and a source vertex s, find the shortest path from s to all other vertices in the graph.



Problem

Single source shortest path: Given a weighted, directed graph $G = (V, E)$ with positive edge weights and a source vertex s, find the shortest path from s to all other vertices in the graph.

- Claim 1: Shortest path is a *simple* path.

Problem

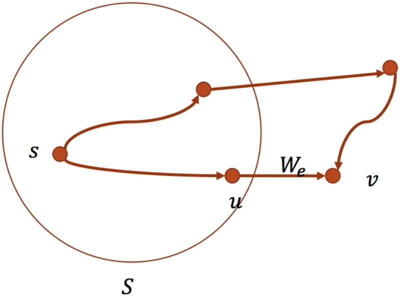
Single source shortest path: Given a weighted, directed graph $G = (V, E)$ with positive edge weights and a source vertex s, find the shortest path from s to all other vertices in the graph.

- Claim 1: Shortest path is a *simple* path.
- Claim 2: For any vertex $x \in V$, let $d(x)$ denote the length of the shortest path from s to vertex x. Let S be any subset of vertices containing s. Let $e = (u, v)$ be an edge such that:
  1. $u \in S$, $v \in V \setminus S$ *(that is, (u, v) is a cut edge)*,
  2. $(d(u) + W_e)$ is the least among all such cut edges.

  Then $d(v) = d(u) + W_e$.

- Claim 2: For any vertex $x \in V$, let $d(x)$ denote the length of the shortest path from s to vertex x. Let S be any subset of vertices containing s. Let $e = (u, v)$ be an edge such that:
  1. $u \in S$, $v \in V \setminus S$ *(that is, (u, v) is a cut edge)*,
  2. $(d(u) + W_e)$ is the least among all such cut edges.

  Then $d(v) = d(u) + W_e$.



- Claim 2: For any vertex $x \in V$, let $d(x)$ denote the length of the shortest path from s to vertex x. Let S be any subset of vertices containing s. Let $e = (u, v)$ be an edge such that:
  1. $u \in S$, $v \in V \setminus S$ *(that is, (u, v) is a cut edge)*,
  2. $(d(u) + W_e)$ is the least among all such cut edges.

  Then $d(v) = d(u) + W_e$.

Algorithm

`Dijkstra's Algorithm`(G, s)
- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

- Claim 2: For any vertex $x \in V$, let $d(x)$ denote the length of the shortest path from s to vertex x. Let S be any subset of vertices containing s. Let $e = (u, v)$ be an edge such that:
  1. $u \in S$, $v \in V \setminus S$ *(that is, (u, v) is a cut edge)*,
  2. $(d(u) + W_e)$ is the least among all such cut edges.

  Then $d(v) = d(u) + W_e$.

Algorithm

`Dijkstra's Algorithm`(G, s)
- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

- What is the running time of the above algorithm?

- Claim 2: For any vertex $x \in V$, let $d(x)$ denote the length of the shortest path from s to vertex x. Let S be any subset of vertices containing s. Let $e = (u, v)$ be an edge such that:
  1. $u \in S$, $v \in V \setminus S$ *(that is, (u, v) is a cut edge)*,
  2. $(d(u) + W_e)$ is the least among all such cut edges.

  Then $d(v) = d(u) + W_e$.

Algorithm

`Dijkstra's Algorithm`(G, s)
- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

- What is the running time of the above algorithm?
  - Same as that of the Prim's algorithm. $O(|E| \cdot \log |V|)$.

- Claim 2: Let S be a subset of vertices containing s such that we know the shortest path length $d(u)$ from s to any vertex in $u \in S$. Let $e = (u, v)$ be an edge such that
  1. $u \in S$, $v \in V \setminus S$,
  2. $(d(u) + W_e)$ is the least among all such cut edges.

  Then $d(v) = d(u) + W_e$.

Algorithm

`Dijkstra's Algorithm`(G, s)
- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

- What is the running time of the above algorithm?
  - Same as that of the Prim's algorithm. $O(|E| \cdot \log |V|)$.

Algorithm

`Dijkstra's Algorithm`(G, s)

- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

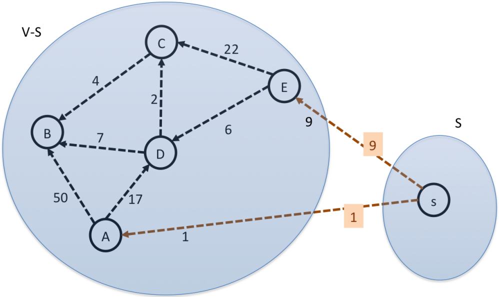


Figure: $d(s) = 0$

Algorithm

`Dijkstra's Algorithm`(G, s)

- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

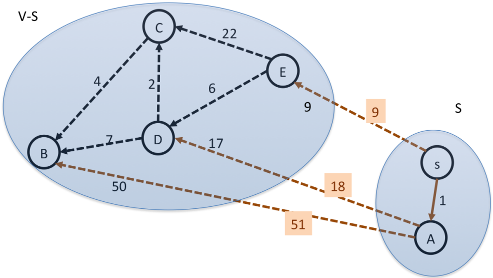


Figure: $d(s) = 0; d(A) = 1$

Algorithm

`Dijkstra's Algorithm`(G, s)

- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

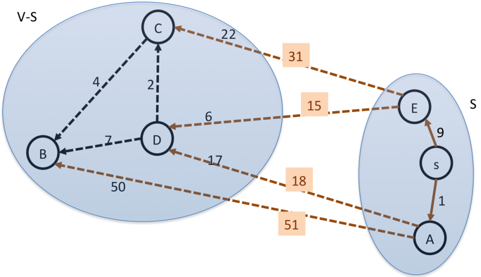


Figure: $d(s) = 0; d(A) = 1; d(E) = 9$

Algorithm

Dijkstra's Algorithm(G, s)
- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

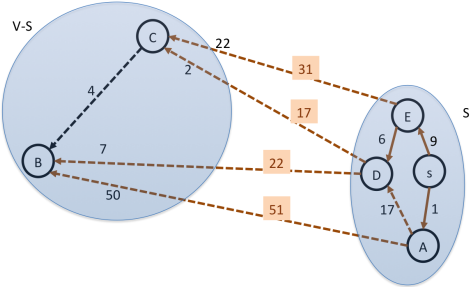


Figure: $d(s) = 0; d(A) = 1; d(E) = 9; d(D) = 15$

Algorithm

Dijkstra's Algorithm(G, s)
- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

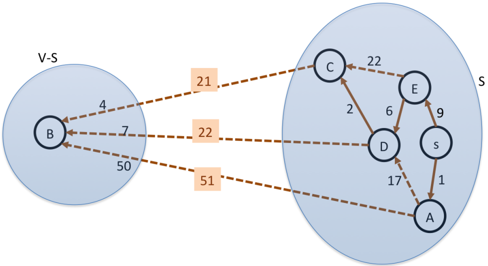


Figure: $d(s) = 0; d(A) = 1; d(E) = 9; d(D) = 15; d(C) = 17$

Algorithm

Dijkstra's Algorithm(G, s)
- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

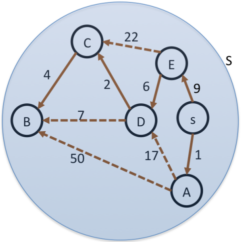


Figure: $d(s) = 0; d(A) = 1; d(E) = 9; d(D) = 15; d(C) = 17; d(B) = 21$

Algorithm

`Dijkstra's Algorithm`(G, s)

- $S \leftarrow \{s\}$
- $d(s) \leftarrow 0$
- While S does not contain all vertices in G
  - Let $e = (u, v)$ be a cut edge across $(S, V \setminus S)$ with minimum value of $d(u) + W_e$
  - $d(v) \leftarrow d(u) + W_e$
  - $S \leftarrow S \cup \{v\}$

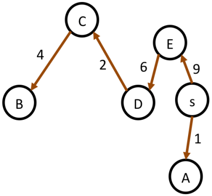


Figure: The algorithm also implicitly produces a *shortest path tree* that gives the shortest paths from s to all vertices.

End