CSC006P1M: Design and Analysis of Algorithms *Lecture* 07 *(Greedy Method-I)*

Sumit Kumar Pandey

August 31, 2022

Greedy Method

- The greedy method for solving optimization problems follows from the philosophy of greedy approach which tries to maximize (minimize) short-term gain and hopes for the best without regard to long-term consequences.
- Algorithm based on the greedy method are usually simple, easy to code and efficient.
- Unfortunately, just as in the real life, in the theory of algorithms also, the greedy method applied to a problem often leads to less than optimal results.
- However, there are important problems where the greedy method does yield optimal results, such as finding a shortest path between two vertices in a weighted graph or determining a minimum spanning tree (MST) in a weighted graph.

Minimum Spanning Tree

Problem

Given an undirected connected weighted graph $G = (V, E)$, find a spanning tree T of G of minimum cost.

Spanning Tree

A spanning tree T of an undirected graph G is a subgraph that is a tree which includes all vertices of a graph.

Minimum Spanning Tree

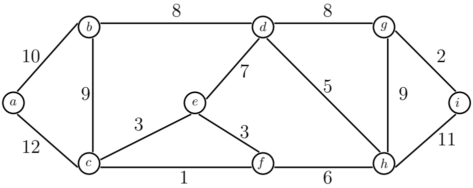


Figure: An Undirected Connected Graph

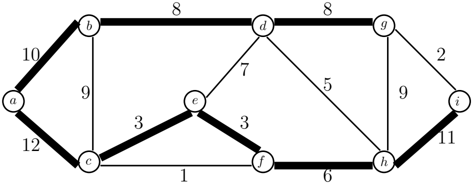


Figure: One Spanning Tree - (1)

Cost of the tree = Sum of all weights in the tree = 61.

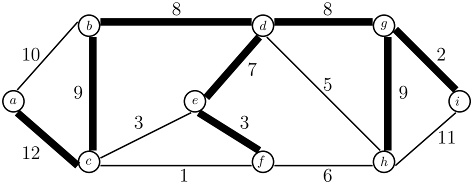


Figure: One Spanning Tree - (2)

Cost of the tree = Sum of all weights in the tree = 58.

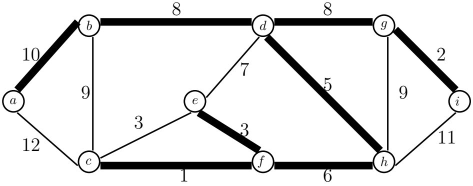


Figure: One Spanning Tree - (3)

Cost of the tree = Sum of all weights in the tree = 43.

Greedy Method:

- Choose any vertex as a root vertex.

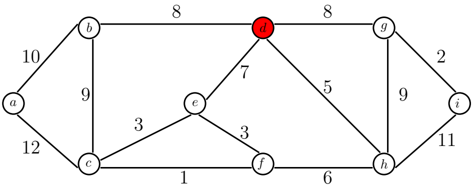
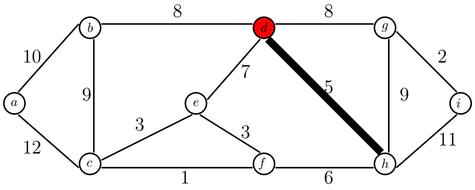


Figure: Root vertex is the vertex (d).

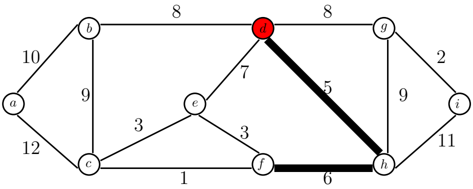
Greedy Method:

- Find the edge having minimum weight and adjacent to the vertex *d*. In case of tie, choose any.



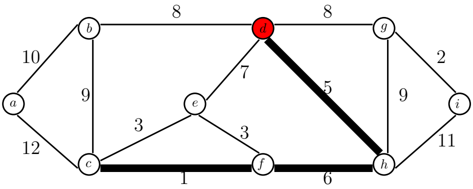
Greedy Method:

- Find the edge having minimum weight among all edges adjacent to both vertices d and h. If that edge forms a cycle, discard that edge and choose the minimum among the rest of edges adjacent to both vertices d and h. Repeat that process, until you find an edge which does not form a cycle. In case of tie, choose any.



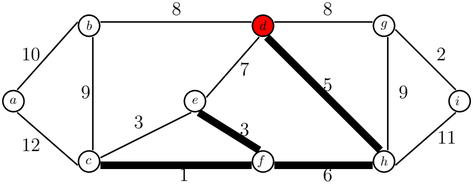
Greedy Method:

- Find the edge having minimum weight among all edges adjacent to all vertices d, h and f. If that edge forms a cycle, discard that edge and choose the minimum among the rest of edges adjacent to all vertices d, h and f. Repeat that process, until you find an edge which does not form a cycle. In case of tie, choose any.



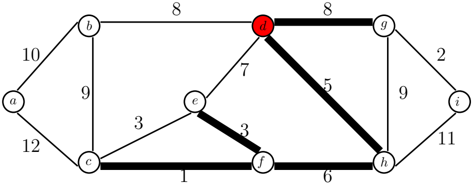
Greedy Method:

- Find the edge having minimum weight among all edges adjacent to all vertices d, h, f and c. If that edge forms a cycle, discard that edge and choose the minimum among the rest of edges adjacent to all vertices d, h, f and c. Repeat that process, until you find an edge which does not form a cycle. In case of tie, choose any.



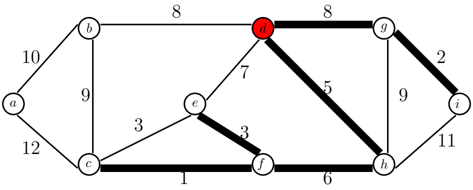
Greedy Method:

- Find the edge having minimum weight among all edges adjacent to all vertices d, h, f, c and e. If that edge forms a cycle, discard that edge and choose the minimum among the rest of edges adjacent to all vertices d, h, f, c and e. Repeat that process, until you find an edge which does not form a cycle. In case of tie, choose any.



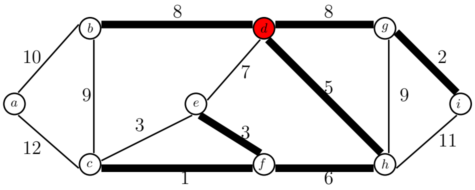
Greedy Method:

- Find the edge having minimum weight among all edges adjacent to all vertices d, h, f, c, e and g. If that edge forms a cycle, discard that edge and choose the minimum among the rest of edges adjacent to all vertices d, h, f, c, e and g. Repeat that process, until you find an edge which does not form a cycle. In case of tie, choose any.



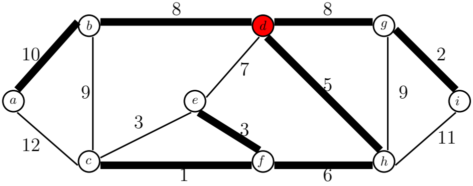
Greedy Method:

- Find the edge having minimum weight among all edges adjacent to all vertices d, h, f, c, e, g and i. If that edge forms a cycle, discard that edge and choose the minimum among the rest of edges adjacent to all vertices d, h, f, c, e, g and i. Repeat that process, until you find an edge which does not form a cycle. In case of tie, choose any.



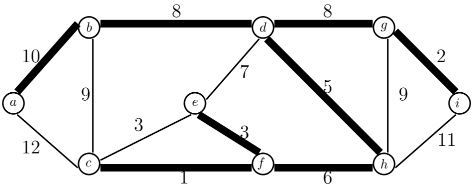
Greedy Method:

- Find the edge having minimum weight among all edges adjacent to all vertices d, h, f, c, e, g, i and b. If that edge forms a cycle, discard that edge and choose the minimum among the rest of edges adjacent to all vertices d, h, f, c, e, g, i and b. Repeat that process, until you find an edge which does not form a cycle. In case of tie, choose any.



Minimum Spanning Tree

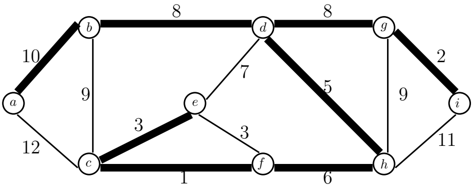
Final Minimum Spanning Tree:



Cost of the tree = Sum of all weights in the tree = 43.

Minimum Spanning Tree

Another Minimum Spanning Tree:



Cost of the tree = Sum of all weights in the tree = 43.

Properties of a Tree

Assumption: The graph is undirected.

Tree

A tree is an acyclic connected graph.

A tree with n vertices has $n - 1$ edges.

A connected graph with n vertices and $n - 1$ edges is a tree.

Properties of a Tree

Adding one edge in a tree produces exactly one cycle.

Any two vertices of a tree are connected by exactly one path.

Let T be a tree and let C be the cycle produced by adding any edge e in T. The removal of any edge e' from C results in a tree T'. If $e' = e$, $T' = T$.

There will be at least one MST in a weighted connected graph.

Proof of the algorithm (A slightly different variation)

Start with the edge with the least weight.

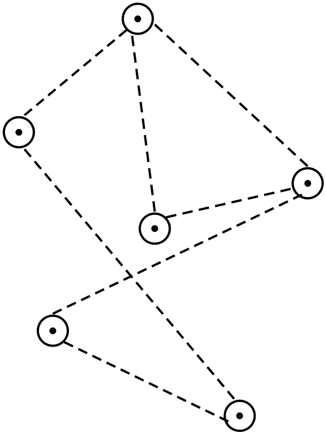
- Claim: the minimum-cost edge must be included in the MST. If it is not included, then adding it to the MST would create a cycle; removing any other edge from this cycle creates a tree again, but with smaller cost, which is a contradiction to the minimality of the MST.
- Assume after k number of iterations, the created graph T_k is a subgraph of some MST T.
- At $k+1^{\text{st}}$ iteration: Divide the vertices in two sets: S_1 and S_2.
    - S_1 contains those vertices which are in T_k.
    - S_2 contains the rest.

  If the new edge in $k+1^{\text{st}}$ iteration contains end vertices from S_1, it will create a cycle which is not allowed. So, the new edge must have one end point in S_1 and another in S_2.

$At\ k+1^{st}$ iteration

- Let E_k be set of edges connecting vertices from S_1 to S_2.
- Claim: The edge with the minimum weight (assuming unique) in E_k belongs to the MST T. Let (u, w) be that edge where $u \in S_1$ and $v \in S_2$.
    - Proof by contradiction: Assume (u, w) is not in T.
    - Since T is a tree, there exist a unique path from u to w.
    - Since $u \in S_1$ and $w \in S_2$, there must be at least one edge (x, y) in this path that connects a vertex in S_1 to a vertex in S_2.
    - We assumed that the weight of (x, y) is higher than the weight of (u, w).
    - Now, add (u, w) in T and remove (x, y) to obtain a spanning tree with a lower cost, which is a contradiction.

Explanation With Diagrams

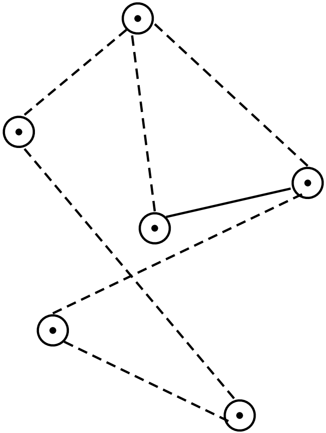


$G = (V, E)$ be the undirected connected weighted graph.

Start with the edge with the least weight.

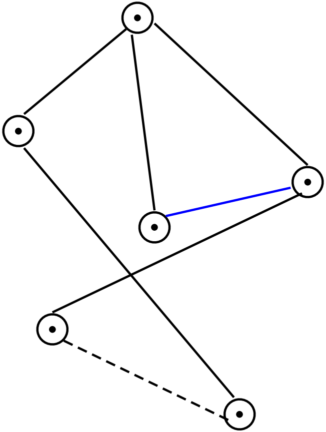
- Claim: the minimum-cost edge must be included in the MST. If it is not included, then adding it to the MST would create a cycle; removing any other edge from this cycle creates a tree again, but with smaller cost, which is a contradiction to the minimality of the MST.

Explanation With Diagrams



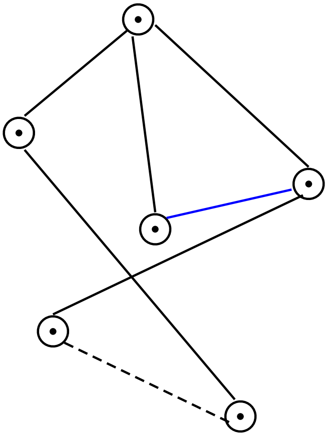
Let the bold edge be an edge with the least weight.

Explanation With Diagrams



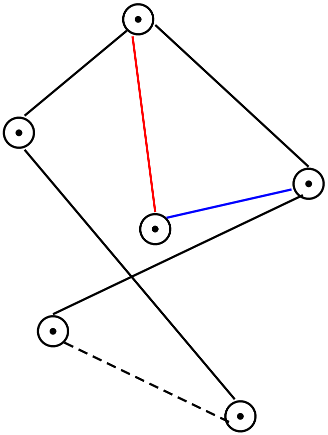
The black edges constitute a final MST. The blue edge is an edge with the least weight.

Explanation With Diagrams



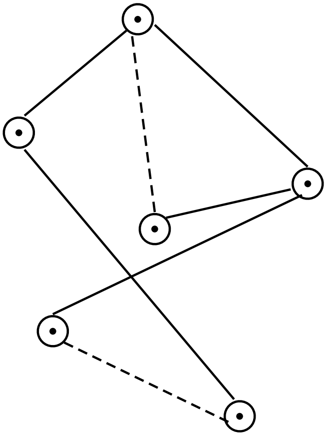
Adding blue edge in the tree forms a cycle.

Explanation With Diagrams



Add blue edge in the tree and remove red edge whose weight is higher than that of blue edge.

Explanation With Diagrams



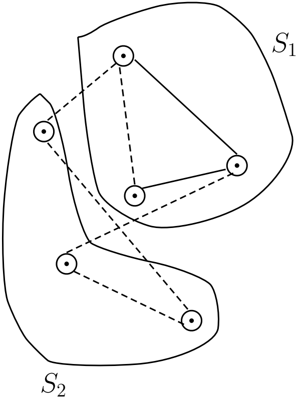
A new tree with a smaller cost, a contradiction.

Explanation With Diagrams

Start with the edge with the least weight.

- Claim: the minimum-cost edge must be included in the MST. If it is not included, then adding it to the MST would create a cycle; removing any other edge from this cycle creates a tree again, but with smaller cost, which is a contradiction to the minimality of the MST.
- Assume after k number of iterations, the created graph T_k is a subgraph of some MST T.
- At $k + 1^{\text{st}}$ iteration: Divide the vertices in two sets: S_1 and S_2.
    - S_1 contains those vertices which are in T_k.
    - S_2 contains the rest.

Explanation With Diagrams



Two sets S_1 and S_2.

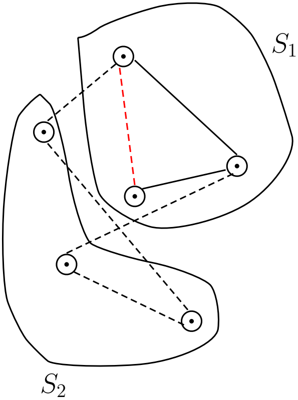
Explanation With Diagrams

Start with the edge with the least weight.

- Claim: the minimum-cost edge must be included in the MST. If it is not included, then adding it to the MST would create a cycle; removing any other edge from this cycle creates a tree again, but with smaller cost, which is a contradiction to the minimality of the MST.
- Assume after k number of iterations, the created graph T_k is a subgraph of some MST T.
- At $k+1^{\text{st}}$ iteration: Divide the vertices in two sets: S_1 and S_2.
    - S_1 contains those vertices which are in T_k.
    - S_2 contains the rest.

  If the new edge in $k+1^{\text{st}}$ iteration contains end vertices from S_1, it will create a cycle which is not allowed. So, the new edge must have one end point in S_1 and another in S_2.

Explanation With Diagrams

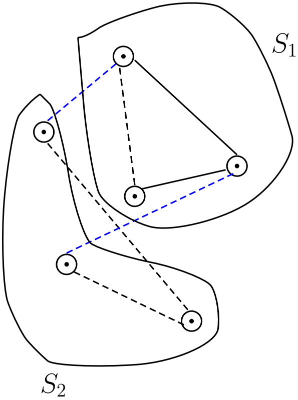


The red dotted edge produces a cycle because the end vertices are in S_1.

Explanation With Diagrams

- Let E_k be set of edges connecting vertices from S_1 to S_2.

Explanation With Diagrams

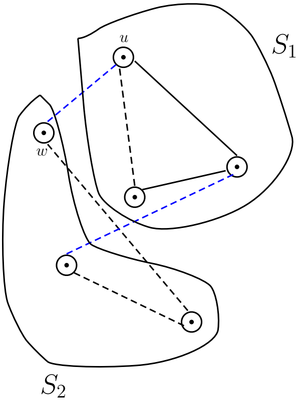


The blue dotted edges are in E_k.

Explanation With Diagrams

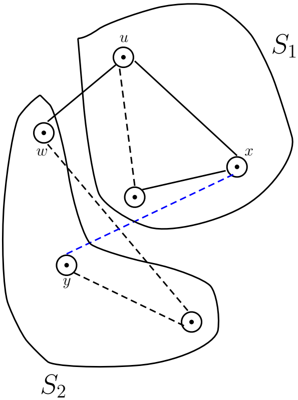
- Let E_k be the set of edges connecting vertices from S_1 to S_2.
- Claim: The edge with the minimum weight (assuming unique) in E_k belongs to the MST T. Let (u, w) be that edge where $u \in S_1$ and $w \in S_2$.

Explanation With Diagrams



Let (u, w) be the edge with the minimum weight in E_k.

Explanation With Diagrams

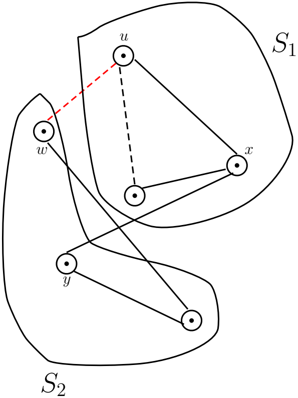


Claim: (u, w) must be in some MST T.

Explanation With Diagrams

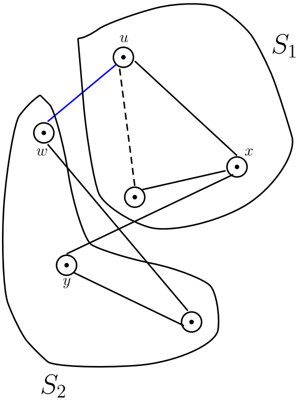
- Let E_k be set of edges connecting vertices from S_1 to S_2.
- Claim: The edge with the minimum weight (assuming unique) in E_k belongs to the MST T. Let (u, w) be that edge where $u \in S_1$ and $v \in S_2$.
    - Proof by contradiction: Assume (u, w) is not in T.
    - Since T is a tree, there exist a unique path from u to w.
    - Since $u \in S_1$ and $w \in S_2$, there must be at least one edge (x, y) in this path that connects a vertex in S_1 to a vertex in S_2.
    - We assumed that the weight of (x, y) is higher than the weight of (u, w).
    - Now, add (u, w) in T and remove (x, y) to obtain a spanning tree with a lower cost, which is a contradiction.

Explanation With Diagrams



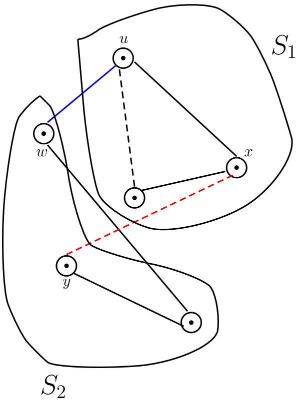
Consider a path from u to w in MST T. Red dotted edge is not in T. There exists an edge (x, y) such that $x \in S_1$ and $y \in S_2$.

Explanation With Diagrams



Add (u, w) in T. It creates a cycle.

Explanation With Diagrams



Removing an edge (x, y) from the cycle creates a tree T' with a smaller cost than that of T.

Thank You