CS2040

Tutorial 9: Minimum Spanning Tree

Nicholas **Russell** Saerang (russellsaerang@u.nus.edu)

:D

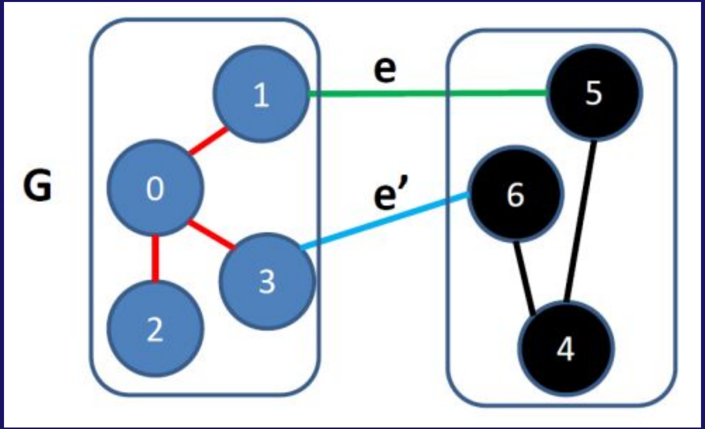
MST

Minimum Spanning Tree (MST)

A tree that spans (covers) every vertex in G with the minimum possible total weight of edges in the spanning tree.

Property:

1. Cycle Property: For any cycle C in graph G(V, E), if the weight of an edge e is larger than every other edge in C, e cannot be included in the MST of G(V, E).
2. Cut Property: For any cut of the graph, if the weight of an edge e in the cut-set is strictly smaller than the weights of other edges of the cut set, then e belongs to all MSTs of the graph.



Prim's Algorithm

1. Start with one vertex, put in set S.
2. In every step, add edge e (that connects S to the outer set) with smallest weight to MST.
3. Let's say e connects node a to b where a is in S, and b is outside S. Add b to S, repeat step 2 until all vertices are in set S.

Use priority queue to select the lowest weight edge, and boolean array to indicate whether a vertex has been visited. Use adjacency list.

Total time complexity: O(E log V)

Kruskal’s Algorithm

1. Sort the edges in increasing order of weight. Initially all n vertices are in disjoint sets.
2. In every step, add edge e with smallest weight to MST if edge e connects a and b where a and b are previously in different sets (merge set in UFDS). Merge set that contains a with set that contains b.
3. Repeat step 2 until all vertices are in the same set.

Use an edge list for sorting edge based on weight.
Use UFDS to check isSameSet(i,j) or to merge(i,j).

Total time complexity: O(E log V)

01

TRUE OR FALSE?

Question 1a

The MST is always a connected, undirected graph.

Answer:
True. The MST is a spanning tree, and by definition a spanning tree will connect all vertices in the graph.

Question 1b

The MST will always have V - 1 edges.

Answer:
True. The MST is a spanning tree, and does not have a cycle. It cannot have more than V - 1 edges. Since all vertices must be connected, it cannot have less than V - 1 edges.

Question 1c

For a graph with unique edge weights, the edge with the largest weight in any cycle of the graph can be included in the MST.

Answer:
False. The edge with the largest weight in any cycle of the graph will always be excluded for unique edge weights.

Question 1d

For a graph with two disjoint sets of vertices A and B (vertices in A are not in B and vice versa), and another vertex x not inside both the sets, the combined MST of A ∪ x and B ∪ x is a MST of the original graph.

Question 1d

Answer:
False. This is not true in all cases. An example is seen in the diagram below. If A = {0,1}, B = {2,3} and some vertex X. The edge (1, 2) should be included in the MST, but the stated method will give a non-minimal spanning tree that does not include (1, 2).

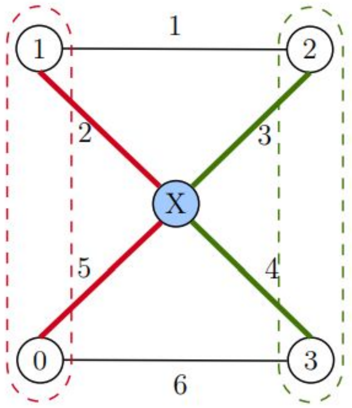


Figure 1: Wrong MST

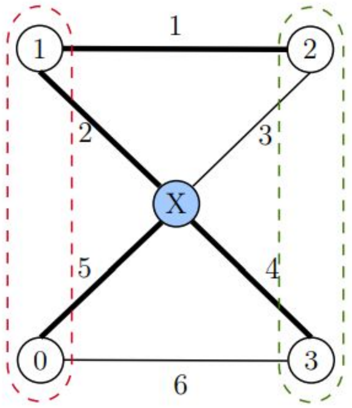
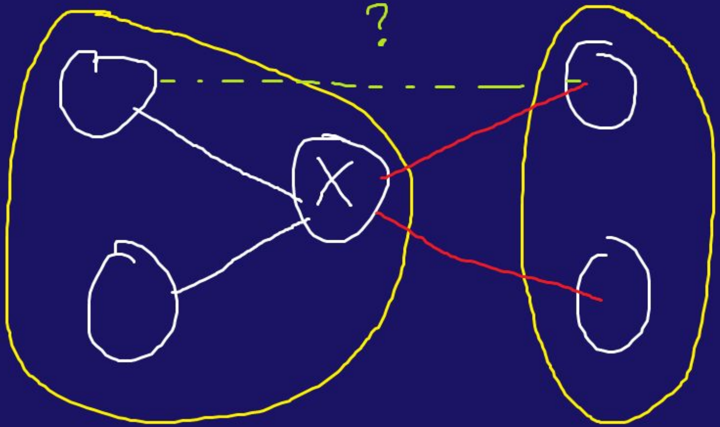


Figure 2: Correct MST

Question 1d

Alternatively, we can consider the cut property. If we consider the two sets of vertices A ∪ x and B, and their cut-set, the smallest weighted edge in the cut-set that connects A ∪ x and B must be inside the MST. This may not necessarily be an edge connecting x to B if there is an edge that directly connects A to B with a lower weight.



02

LET'S ADD SOME NEW STUFF!

Question 2a

Given a graph G with V vertices and E edges, and the unique Minimum Spanning Tree (MST) of G (i.e. G has only 1 MST), give an algorithm (including the time complexity) to update the MST if

A new edge (A, B) is to be inserted into G

Insert (A, B) into the original graph G and re-run Prim's or Kruskal's. Will run in O((E + 1) log V) = O(E log V).

Can we do better? **Above is not the best solution!**

Question 2a

Insert (A,B) into the original graph G and re-run Prim's or Kruskal's. Will run in O((E + 1) log V) = O(E log V).

Can we do better? **Above is not the best solution!**

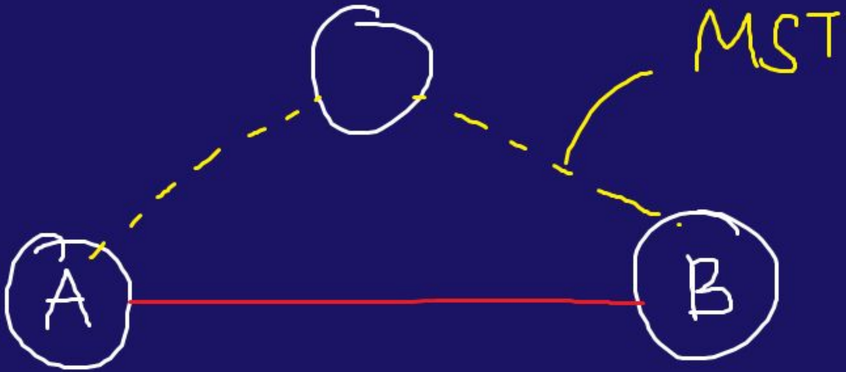
Insert the edge (A,B) into the MST itself and then re-run Prim's or Kruskal's on (this MST) + (1 edge).

- MST is a tree so there will be V - 1 edges.
- Time complexity will be O(V log V).

Can we do better? **Above is also not the best solution!**

Question 2a

Before insertion of this edge (A, B) into the MST, there is only a unique path from vertex A to vertex B in the MST. After the insertion of edge (A, B), we have a cycle which consists of the edges of the path from A and B and the new edge (A, B). In order to update the MST, we only need to find the largest edge in this cycle and remove it. This will ensure that the resultant graph will remain the MST for the new graph G ∪ (A, B).



Question 2a

Simply run DFS from A in the original MST to find the unique path from A to B. This allows us to find the edge X with the largest weight in the path from A to B.

- If the weight of (A, B) is larger than X, then the MST does not change.
- Otherwise, we remove X and insert (A,B) instead into the MST to have a better MST.

This algorithm runs in O(V) since DFS is O(V) on a tree and removing X and inserting (A, B) can also be done in O(V) time. (e.g. we use a (sorted) EdgeList, we can find X with an O(V) loop and we can insert (A, B) in the correct position also in O(V)).

Question 2b

Given a graph G with V vertices and E edges, and the unique Minimum Spanning Tree (MST) of G (i.e. G has only 1 MST), give an algorithm (including the time complexity) to update the MST if

A new vertex Y, along with a set of edges connecting Y to the rest of the graph, is inserted to G

Take the original graph G with the new vertex Y and its edges and re-run Prim's or Kruskal's which will take O(E log V) time.

Can we do better? **Above is not the best solution!**

Question 2b

However, we can observe that in the resulting MST, the edges can only consist of edges from the MST of the original graph and the new edges connected to vertex Y.

If any other edge can be found in the resulting MST, then the MST of the original graph would not be unique. Thus, we run Prim's or Kruskal's on the original MST together with vertex Y and its edges.

We note that when adding a new vertex Y, there will be at most V edges that connect Y to the original V vertices in the graph. Thus, there are a total of V + 1 vertices and V - 1 + V = 2V - 1 edges. The time complexity is thus O(V log V).

Question 2b

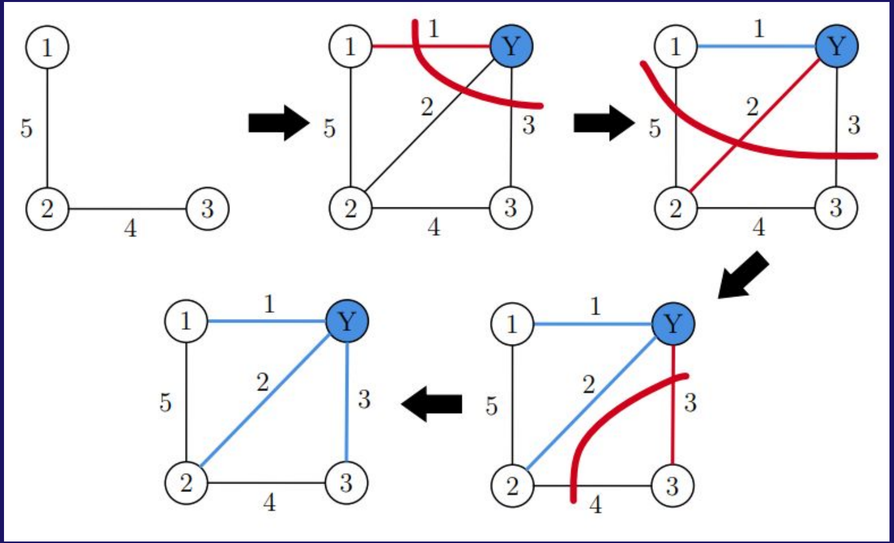
Did you think:
We will always only take 1 edge from the edges connecting to Y and all V - 1 edges from the original MST to form the updated MST.

Wrong, next slide will explain why.
It’s a common misconception! (and temptation :D)

Question 2b

An example of a repeated application of the cut property after the addition of vertex Y.

- The first graph = the original MST
- Red curve = the current cut of the graph we are considering
- Red edge = the edge in the cut set with the lowest cost, that is to be added into the MST
- Edges in blue = previously chosen edges.



03

CLEARLY HAS MST FLAVOR

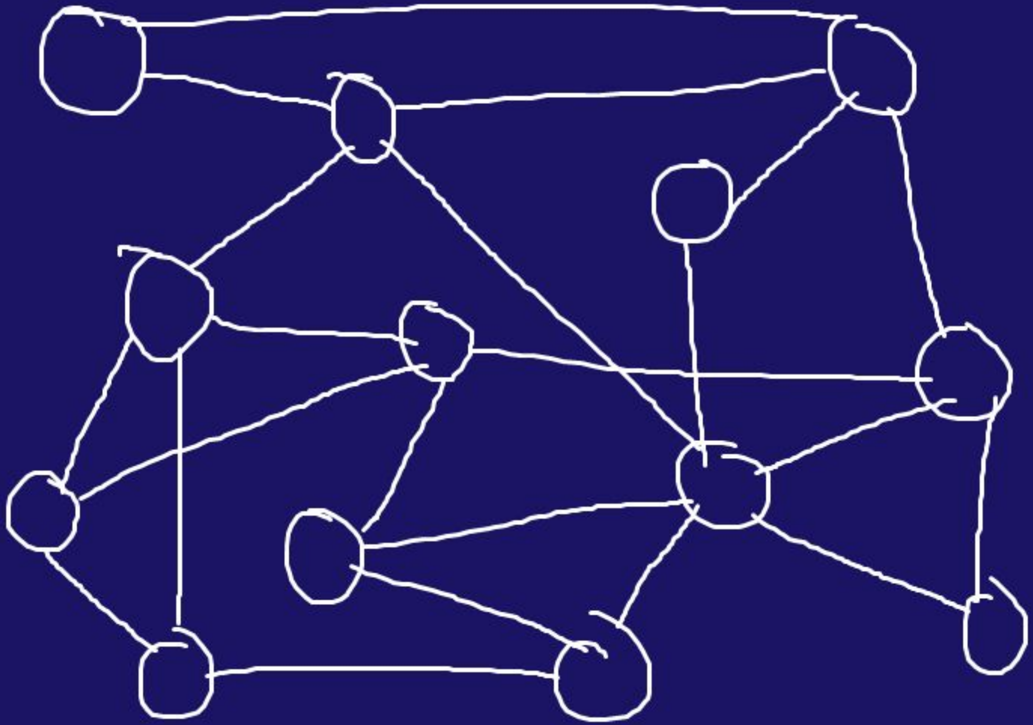
Question 3

Abridged problem description:
A company want to connect offices with high speed fibre optics, but the MST cost of the whole network is exceeding the budget b, so they decide to group the government offices into k groups and link up the offices in each group, but not offices between groups to save on the cost. This effectively creates k intra-nets instead of one big intra-net.

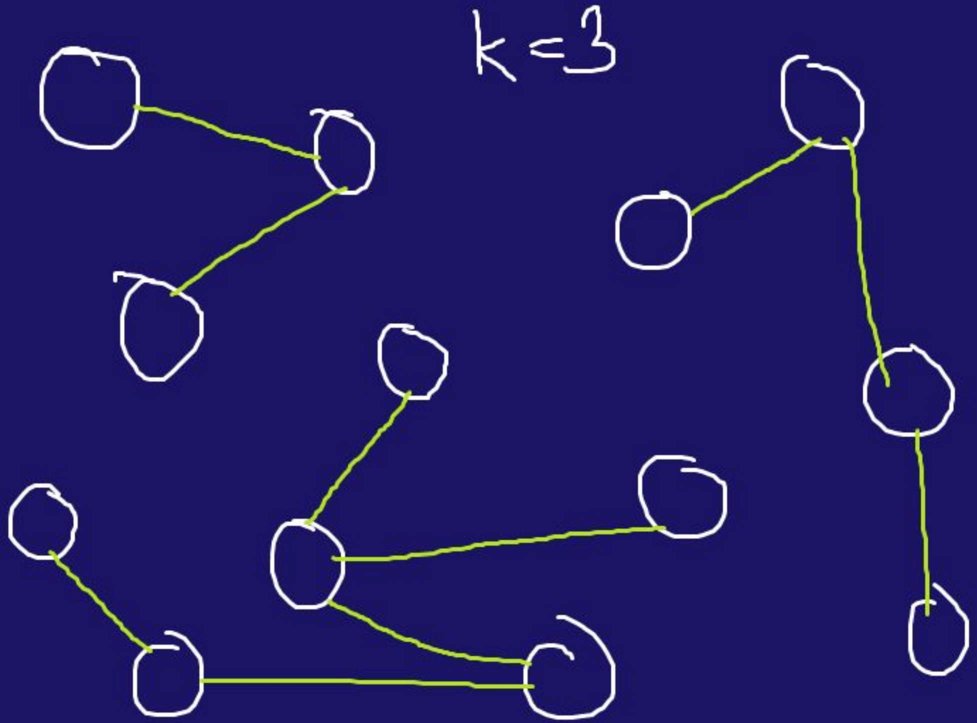
Given V government offices, the cost of linking E pairs of government offices, and a budget b, help the company design a program which will tell them what is the smallest value of k (so as to minimize the number of intra-nets). The program also needs to output the government offices in each of the k groups and how they should be linked such that the total cost of all selected links is within budget b.

The program should model the problem as a graph. It should also run in O(E log V) time.

Example office graph



Add one more edge then > b



Question 3

Answer:
Run Kruskal’s which sorts E edges from cheapest to most expensive (Prim’s algorithm is not suitable to solve this problem, why?), keep track of sum of the weights of the edges picked so far. Once an edge that is going to be added will cause the sum to exceed the given budget b, we stop the algorithm. Then report k = the number of disjoint sets currently in the UFDS.

We can do an O(V) pass to trace the p[i] array of the UFDS to list down members of each groups. How the offices are linked is simply listing down all the edges that are picked.

As we don’t do anything other than potentially stopping Kruskal’s earlier, this is at worst O(E log V), same as the standard Kruskal’s algorithm.

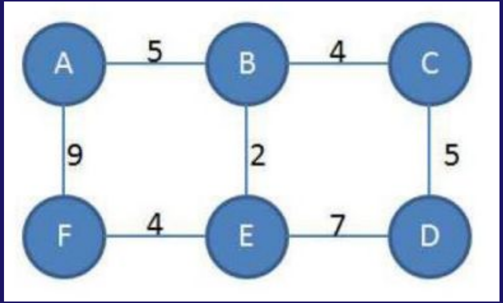
04

THE MST PROBLEM IS NOT OBVIOUS

Question 4

(AY11/12 S1 Finals: Vehicle Monitoring System)
Given n junctions and m roads, we want to place security cameras on the road (cost equals to edge weight) such that each circuit/loop has at least one camera. Output the minimum cost to do so.

Example below, the three possible racing circuits are {A,B,E,F,A}, {B,C,D,E,B}, and {A,B,C,D,E,F,A}. We need at least two cameras: One located at road B-E with cost 2 (hundreds SGD) and another one at road B-C (or road E-F) with cost 4 (hundreds SGD), with total cost of C = 6 (hundreds SGD). Therefore, the output is C = 6.



Question 4

Simply run Kruskal’s but sort edges in non-increasing (downwards) order to obtain the Maximum Spanning Tree (MaxST). Edges not taken by Kruskal’s for the MaxST will be the edges where we have to place the cameras, and take the sum of their weights.

The time complexity is simply O(m log n), the same as Kruskal’s time complexity.

But Russell, the solution is so short yet so obscure!
Why u do dis to me :’(

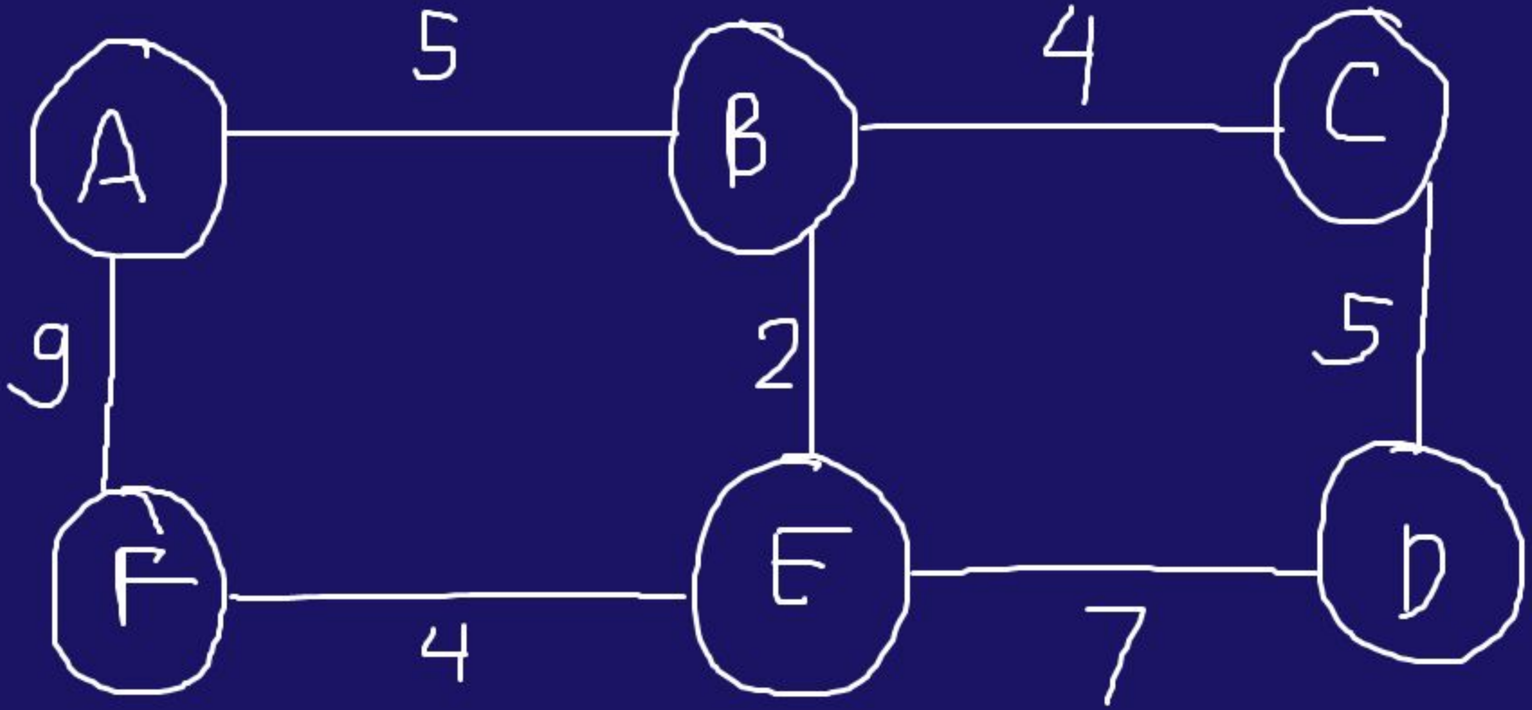
Cycle Property!

Recall the cycle property:
For any cycle C in graph G(V, E), if the weight of an edge e is larger than every other edge in C, e cannot be included in the MST of G(V, E).

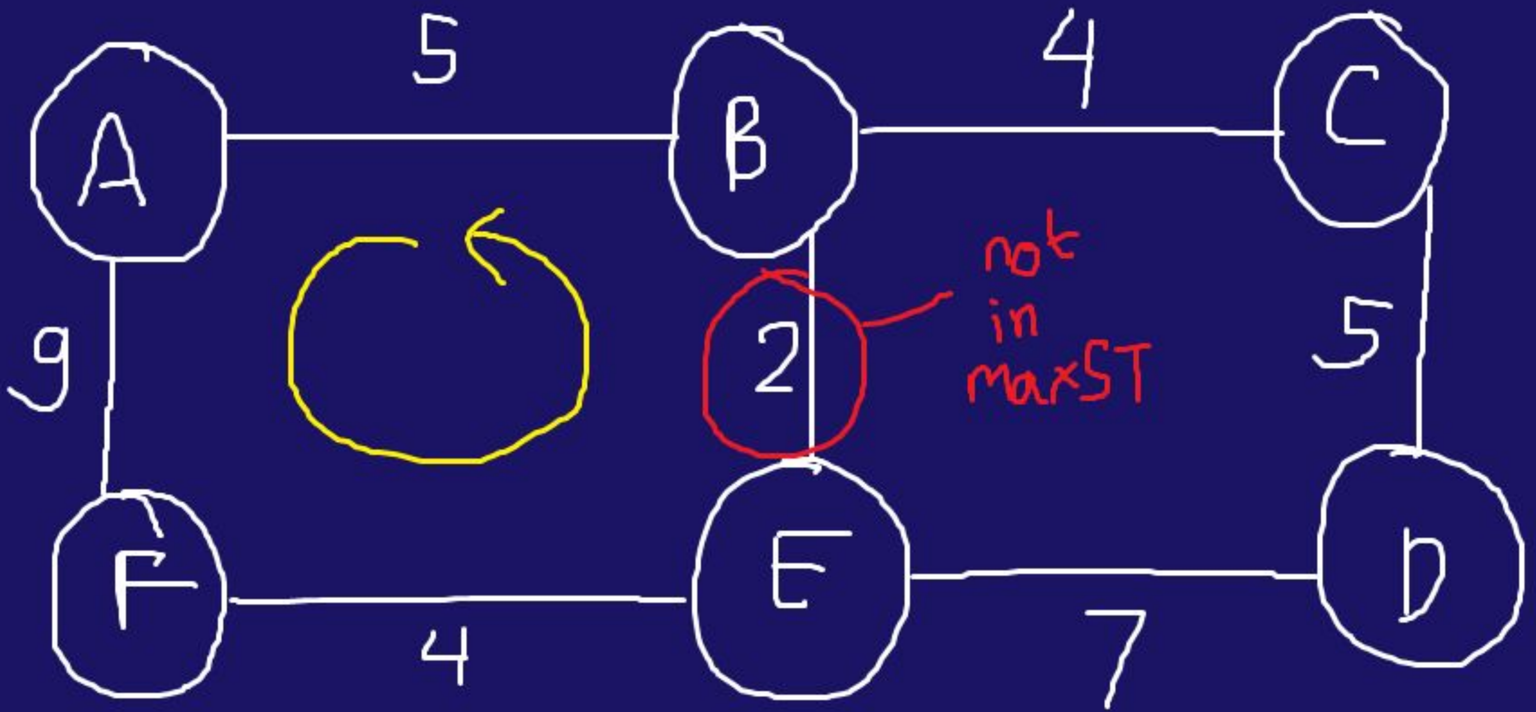
Equivalent to this?
For any cycle C in graph G(V, E), if the weight of an edge e is smaller than every other edge in C, e cannot be included in the MaxST of G(V, E).

This means the smallest weighted edge on every single cycle must not be part of the MaxST, and for every cycle there will be at least one edge not a part of the MaxST, so we can place the cameras on those edges :)

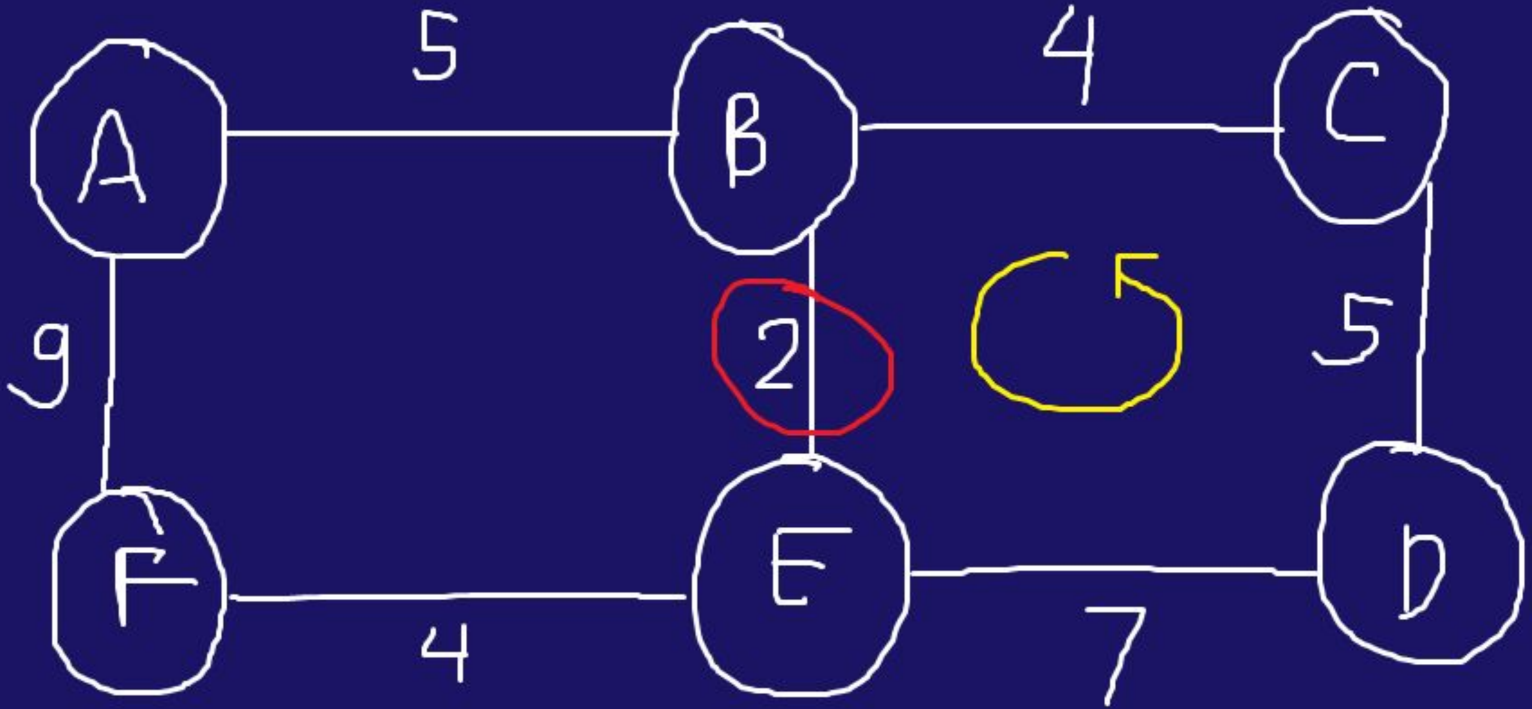
Cycle Property!



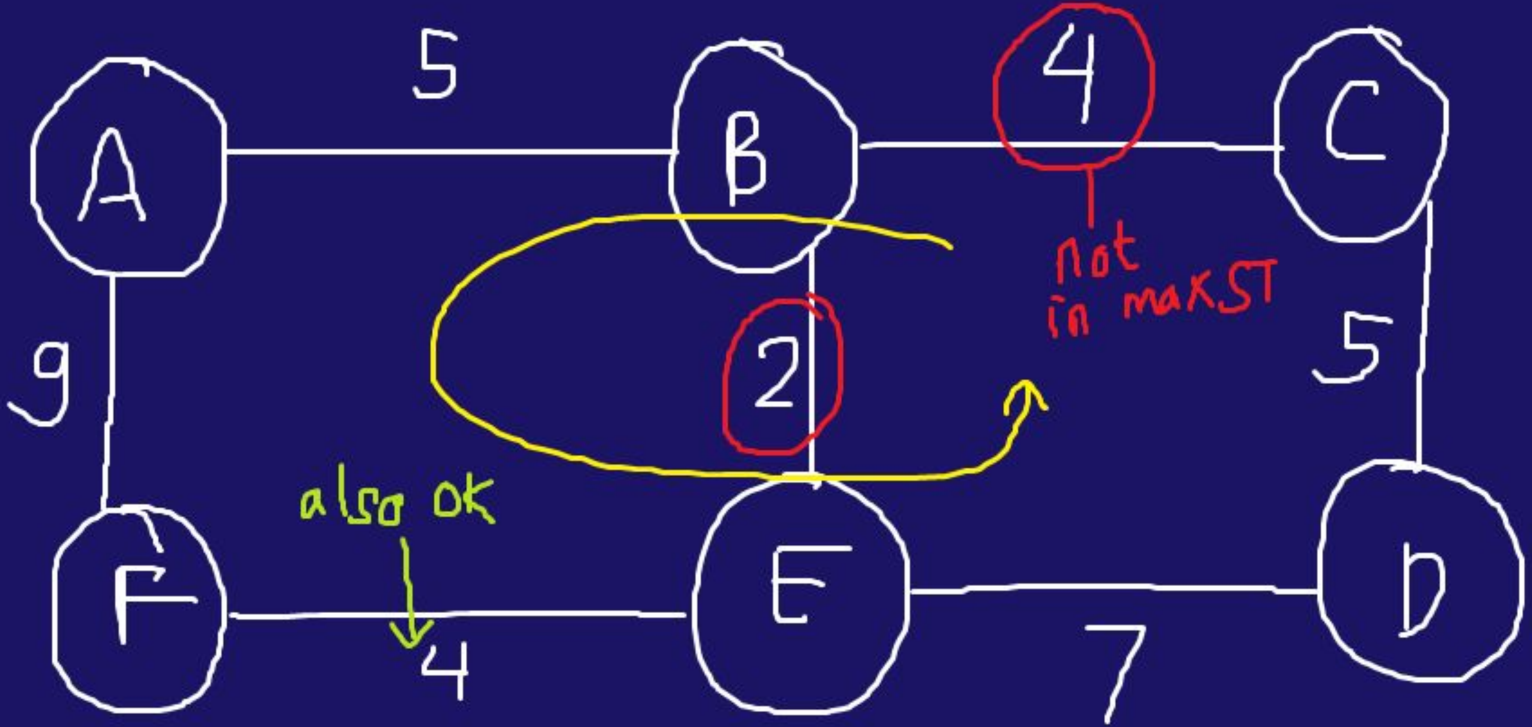
Cycle Property!



Cycle Property!



Cycle Property!



05

IS THIS MST

Question 5

(ICPC 2013 Indonesia)
Power plants, cities, towns and all other important sites can be represented as a graph where each node represents a site and each edge which connects two different sites represents a cable transferring electricity between the two sites in both directions. You may assume that all sites are connected through some cables and for each pair of sites there is at most one cable connecting them.

The government has a plan to calculate the minimum total cost to maintain only necessary cables such that **all sites are connected to at least one power plant**, except for the power plants (which are already connected to themselves). A site is considered connected to a power plant if and only if **there is a path which consists of only maintained cables from the site to a power plant**.

Question 5

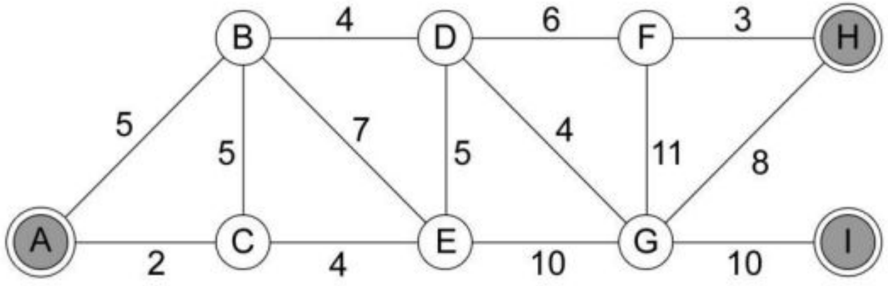


Figure 3: An example of power plants, sites and cables

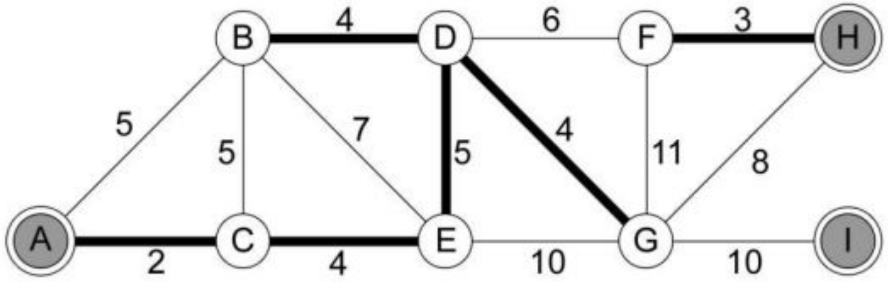


Figure 4: An example of the edges chosen to minimise total cost of cables to build

Question 5

Answer:
This is multi-source MST. Simply run Prim’s algo but during initialization, enqueue the priority queue with not just one vertex as source but all vertices representing power plants as source vertices.

In this way, each power plant vertex will grow an MST to a subset of the vertices in the graph (those cities and towns using that power plant), and all the MSTs will cover all vertices in the graph.

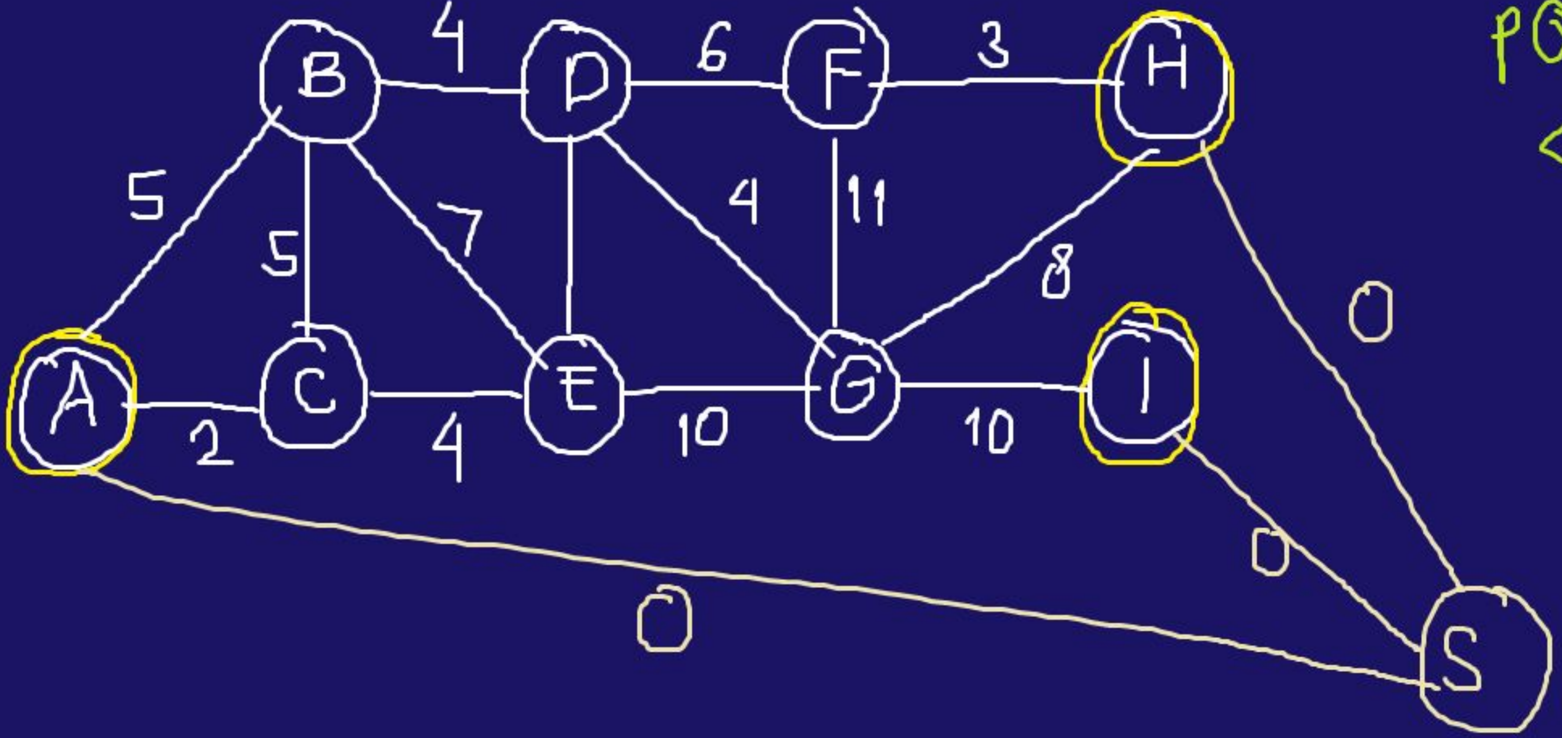
These MSTs will be disjoint since vertices already added to some MST will not be added to another MST (checking the taken array during Prim’s).

Question 5

Alternative answer:
Create a special/imaginary node s which has an edge of weight 0 to every vertex representing power plants. Then run Prim’s by initializing the priority queue with s. This ensures that the MST created will include the edges of weight 0 to each power plant. Removing those edges will give you the MSTs associated with each power plant.

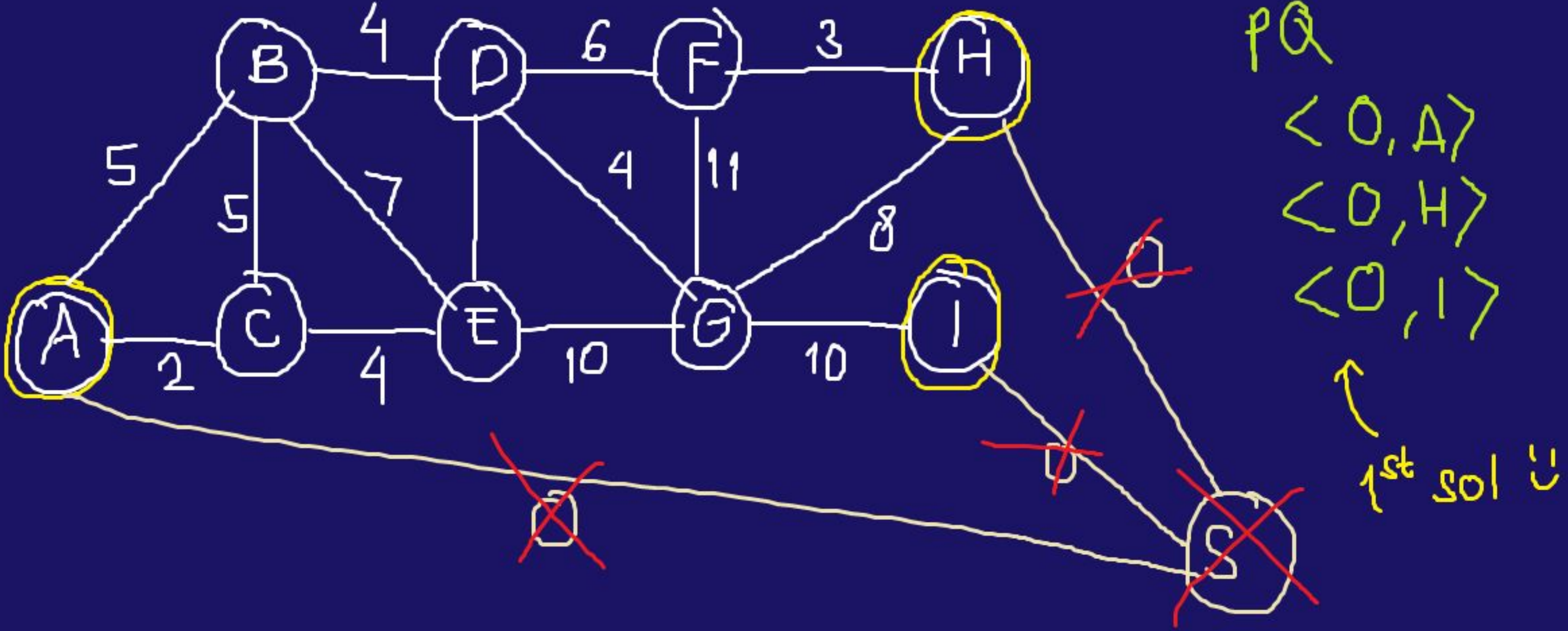
Why is this equivalent to the previous solution?

Question 5



PQ
<0, S>

Question 5



THE END!