Data Structures
Minimum Spanning tree

Cain Susko

Queen's University
School of Computing

March 30, 2022

Time Complexity

The time complexity for an MST with n elements is:

Create	$O(1)$
Find	$O(\log n)$ or $O(1)$
Union	$O(\log n)$ or $O(1)$

the last 2 are $O(1)$ if the targets are already found. (ie. already found root a and b for Union)

Prim's Algorithm

This is an algorithm to grow a MST in successive steps:

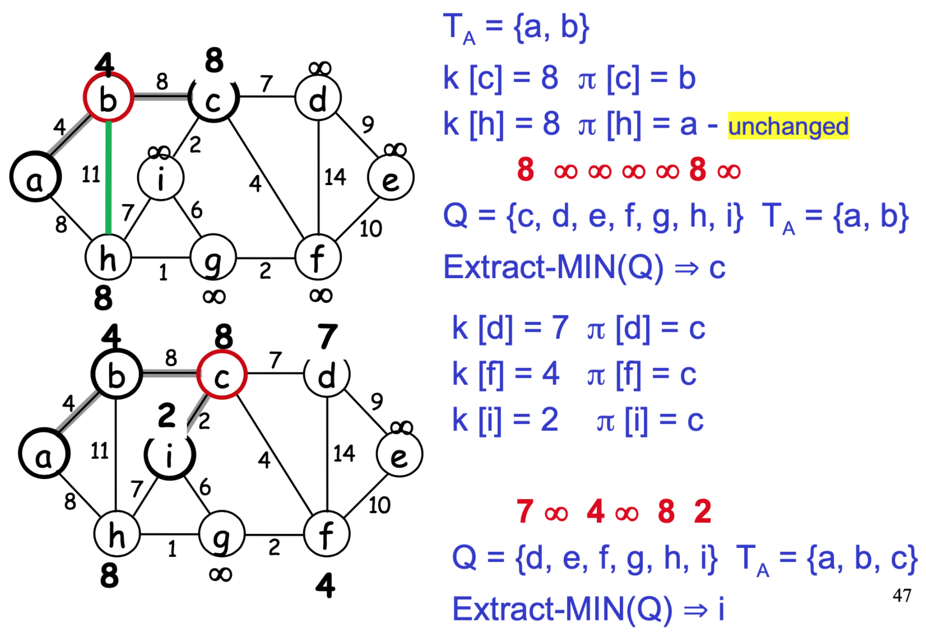
1. given the set S of the n edges we are choosing
2. randomly choose a vertex $v \in T$ and the rest of the vertices are in the set Q
3. while the size of S is less than $n - 1$:
   (a) let e be the edge with the lowest weight which connects v to any vertex in Q (ie. $e = (x, y)$ where x is in T and y is in Q)
   (b) add e to S
   (c) add y to T
   (d) remove y from Q
4. return S

Note: this is a 'greedy'/'local' algorithm for taking a graph and converting it into a MST.

Finding Lightest Edge a key part of Prim's algorithm is finding the lightest edge in a Tree. To quickly do this, we are given a list/dictionary K which stores a priority-vertex pair. $K[v]$ stores the 'shortest distance' between v and the current tree. The algorithm is as follows:

1. After adding a new node to the tree T we update the weights of all nodes adjacent to it
2. $K[v] = \infty$ if v is not adjacent to any vertices in T

Consider the following example of Prim's algorithm where we are given a graph, and it is gradually converted into MST:



T_A = {a, b}

k [c] = 8 π [c] = b

k [h] = 8 π [h] = a - unchanged

8 ∞ ∞ ∞ ∞ 8 ∞

Q = {c, d, e, f, g, h, i} T_A = {a, b}

Extract-MIN(Q) ⇒ c

k [d] = 7 π [d] = c

k [f] = 4 π [f] = c

k [i] = 2 π [i] = c

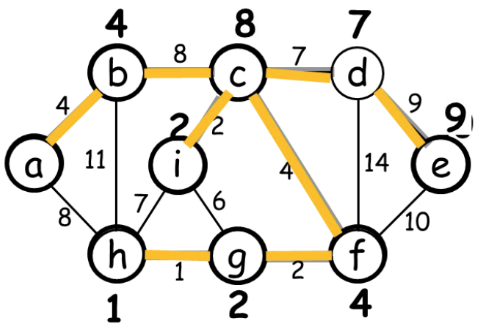
7 ∞ 4 ∞ 8 2

Q = {d, e, f, g, h, i} T_A = {a, b, c}

Extract-MIN(Q) ⇒ i

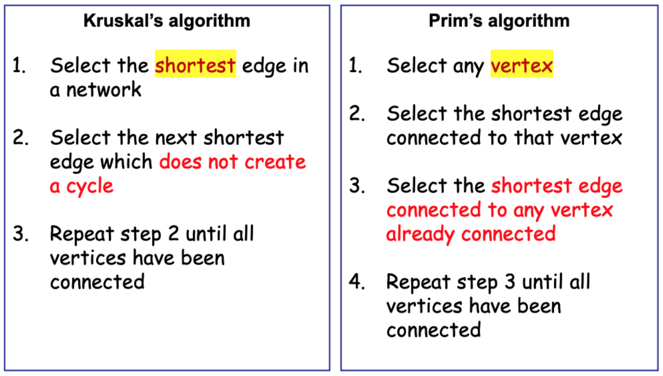
47

This is just an example of 2 iterations of the algorithm, going from b to c, and choosing i as the next vertex with the lightest edge.
The final result for this graph would be:



Prim's v. Kruskal's

For either algorithm that is used to make a MST, there are pro's and cons.



Kruskal's algorithm

1. Select the shortest edge in a network
2. Select the next shortest edge which does not create a cycle
3. Repeat step 2 until all vertices have been connected

Prim's algorithm

1. Select any vertex
2. Select the shortest edge connected to that vertex
3. Select the shortest edge connected to any vertex already connected
4. Repeat step 3 until all vertices have been connected

It is also good to note that both algorithms will always give the same solutions with the same length, even if the order in which the vertices are chosen is different. Occasionally, the algorithms will choose different edges but this is ony when either option is the same length.