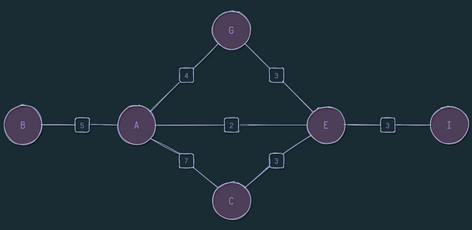
Puntos de la ciudad → vértices del grafo
Cañerías posibles que unen distintos puntos → Aristas
Precio de renovar la cañería de punto X a punto Y → Peso de la arista

Para obtener qué cañerías necesitan arreglarse, necesito el árbol de tendido mínimo, éste va a conectar todos los puntos con la suma mínima de aristas

Esto se puede resolver tanto con el algoritmo de Prim como con el de Kruskal, ambos tienen la misma complejidad → O(E log E) → O(E log V)

Resolución con Prim



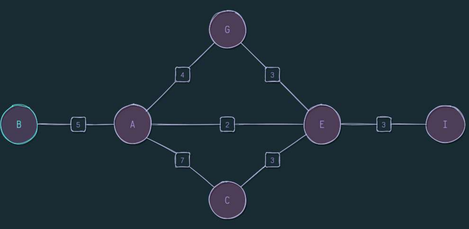
Prim:
Se empieza por un vértice aleatorio
Se mete en un heap todas las aristas (las de vértices no visitados)
Saco una arista del heap (cmp → peso mínimmo), y si el destino no está visitado la agrego al árbol
Se vuelve al 2° paso

Seguimiento

Empiezo por un vértice cualquiera → B
Lo agrego a visitados y dado que A no fue visitado agrego esa arista

Visitados: B

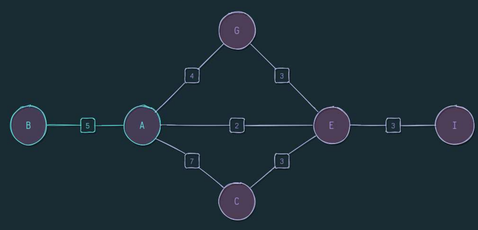
Heap:
(BA: 5)



Desencolo del heap → (BA: 5)
Vértice → A
Agrego a A a visitados, agrego la arista (BA: 5) al árbol, encolo las aristas de A con C, E y G (con B no porque se encuentra en visitados)

Visitados: B - A

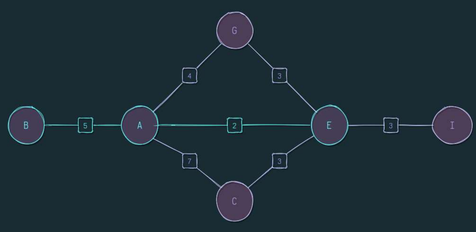
Heap:
(AC: 7) (AE: 2)
(AG: 4)



Desencolo del heap → (AE: 2)
Vértice → E
Agrego a E a visitados, agrego la arista (AE: 2) al árbol, encolo las aristas de E con C, G, e I (con A no porque se encuentra en visitados)

Visitados: B - A - E

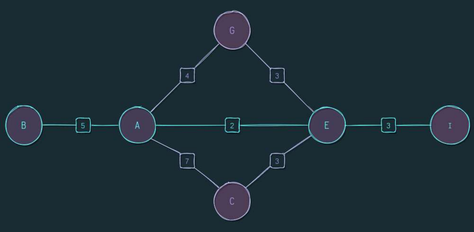
Heap:
(AC: 7) (AG: 4)
(EG: 3) (EC: 3)
(EI: 3)



Desencolo del heap → (EI: 3)
Vértice → I
Agrego a I a visitados, agrego la arista (EI: 3) al árbol, no encolo la arista de I (porque E se encuentra en visitados)

Visitados: B - A - E - I

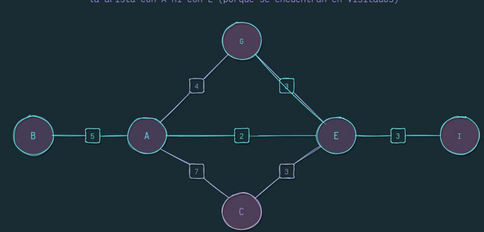
Heap:
(AC: 7) (AG: 4)
(EG: 3) (EC: 3)



Desencolo del heap → (EG: 3)
Vértice → G
Agrego a G a visitados, agrego la arista (EG: 3) al árbol, no encolo la arista con A ni con E (porque se encuentran en visitados)

Visitados: B - A - E - I - G

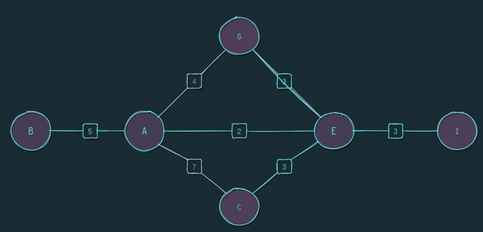
Heap:
(AC: 7) (AG: 4)
(EC: 3)



Desencolo del heap → (EC: 3)
Vértice → C
Agrego a C a visitados, agrego la arista (EC: 3) al árbol, no encolo ninguna arista de C porque sus adyacentes se encuentran visitados

Visitados: B - A - E - I - G - C

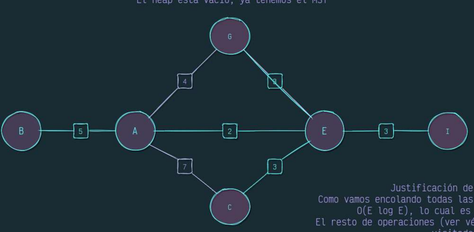
Heap:
(AC: 7) (AG: 4)



Desencolo del heap → (AG: 4)
Vértice → G, ya está visitado, no hay nada por hacer
Desencolo del heap → (AC: 7)
Vértice → C, ya está visitado, no hay nada por hacer
El heap está vacío, ya tenemos el MST

Visitados: B - A - E - I - G - C

Heap:



Costo mínimo de cañerías: $16

Justificación del orden para prim:
Como vamos encolando todas las aristas del grafo en un heap es O(E log E), lo cual es equivalente a O(E log V)
El resto de operaciones (ver vértice, ver visitados, actualizar visitados) es O(1)