MINIMUM SPANNING TREES (MST) & SHORTEST PATHS (SP)

Tuesday, October 23, 2018 5:08 PM

- brute / exhaustive
- transform
- decrease
- divide
- greedy

Applications of GREEDY TECHNIQUE TO graph problems

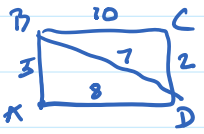
MINIMUM SPANNING TREE

INPUT: a weighted graph $G = (V, E, W)$

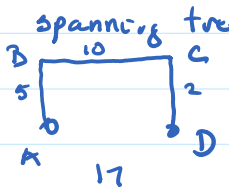
OUTPUT: a spanning tree of G with minimum total weight

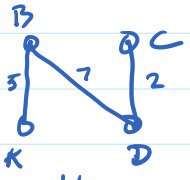
* spanning tree of a graph G is a subgraph T that is connected & acyclic and contains all of the original vertices.

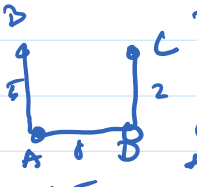
Ex: G =

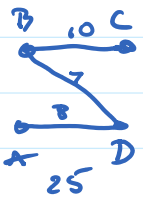


spanning trees









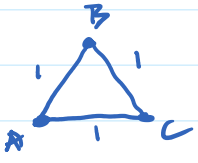
In general, a graph has multiple spanning trees. We want the "lightest" one

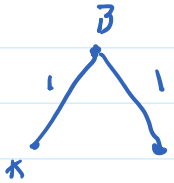
Since all vertices must be included in a spanning tree, MST boils down to selecting n-1 edges (if G has n vertices). This is good candidate for Greedy Technique

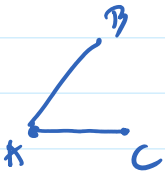
Pick the lightest edge

WORKS!!

Ex:

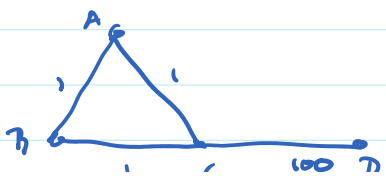


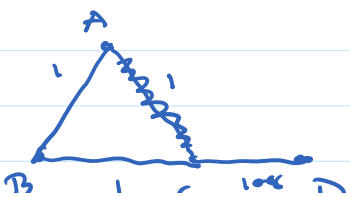




works

Ex:



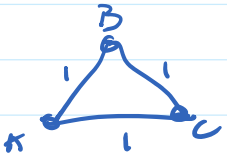


doesn't work

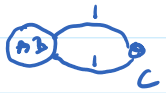
CORRECT GREEDY STRATEGY : pick lightest edge unless it creates a cycle with already picked edges.

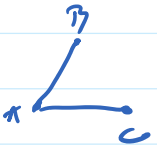
```
KRUSKAL ( V, E, w)
{
    if |V| == 1
        return { }
    let {a,b} be the lightest edge
    {
        compress {a,b} into a single vertex ; new graph is V', E'
        return  {a,b} ∪ KRUSKAL (V', E', w)
    }
}
```

Ex :



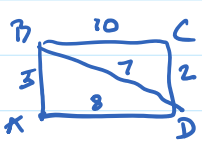
{a,b} ∪ {c,a} ∪ { }

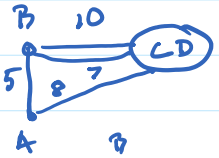


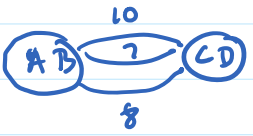


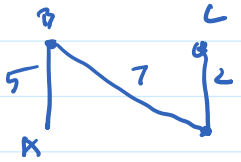
Ex :

{c,d} {a,b} {b,d} { }

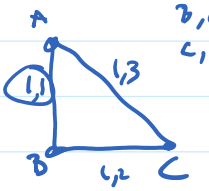








A,B = 1
B,C = 2
C,A = 3



CLAIM : There is a MST T' that includes the lightest edge e_0.

PROOF : Let T be an MST. If T includes e_0 then we are done.
If not, $T \cup \{e_0\}$ has a cycle containing e_0

Removing any other edge e on this cycle yields a tree T' no heavier than so T' is another MST that includes e_0.

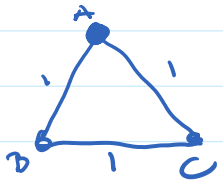
Compressing e_0 into a vertex yields a MST for the new graph.

MORE EFFICIENT IMPLEMENTATION

```
KRUSKAL (V,E,w)
{
    sort E in increasing weight order        Θ(m lg m)
    T = { }
    for each e in E                          Θ(m( n ))
        if T ∪ {e} is acyclic
            T = T ∪ {e}
    return T
}
```

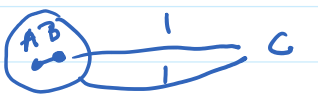
$\Theta(m \lg m + mn)$

ANOTHER MST SOLUTION BY PRIM



Select a special vertex A

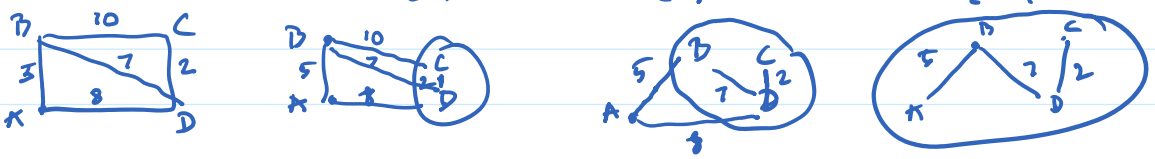
There is a MST that includes the lightest edge incident to A



Ex:

Select D

Pick {c,d} Pick {b,d} Pick {a,b}




```
Prim (V, E, w, s)
{
    if |V| == 1
        return { }
    let {s,a} be the lightest edge incident to s; compress {s,a} into (sa); (V', E')  new graph
    return {s,a} ∪ Prim (V', E', w', (sa))
}
```

CLAIM: There is an MST that contains the lightest edge e_0 incident to s, s any vertex in V.

claim: There is an MST that contains the lightest edge e_0 incident to s, s any vertex in V.

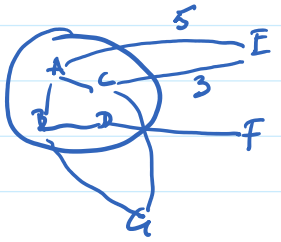
Proof:

Let T be an MST. If T includes e_0 then we are done. $e_0 = \{s,a\}$

Otherwise $T \cup \{e_0\}$ has a cycle with another edge incident to s $\{s,b\}$

Since $\{s,b\}$ is no lighter than $\{s,a\}$, $T - \{s,b\} \cup \{s,a\}$ is another MST that includes e_0

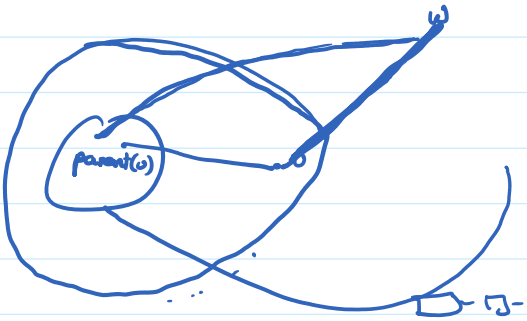
MORE EFFICIENT IMPLEMENTATION

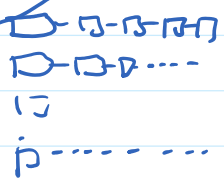


For each vertex A not yet compressed,
keep the lightest edge between A and a compressed vertex
Maintain these values on a heap

```
Prim (V, E, W, s)
{
   T = { }
   for each v ∈ V
   {   key (v) = ∞ ; parent (v) = null;
   }
   key (s) = 0;
Θ(n)  Q = Make_Heap (V)
      while (!Q.empty())
Θ(n)  {  u = Q.extract_min(); lg n
         T = T ∪ {(u, parent(u)}
         for each v ∈ Adj[u] and v ∈ Q
              if w(v,u) < key (v)
                 { key (v) = w(v,u) ; parent[v] = u;
                 ; Q.decrease_key(v, key(v)); } → m lg n
      }
}
```

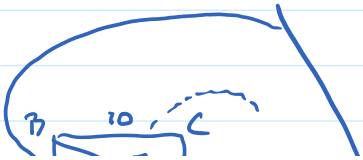
Running Time: $\Theta(n) + \Theta(n \lg n) + \Theta(m \lg n)$
$= \Theta((n+m)\lg n) = \Theta(m \lg n)$

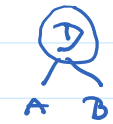


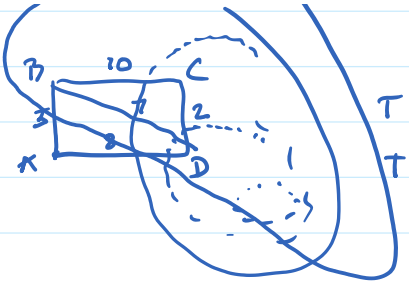


$s = D$

	A	B	C	D
key	~~∞~~ ~~5~~	~~∞~~ 7	~~∞~~ 2	0
parent	~~D~~ B	D	D	—



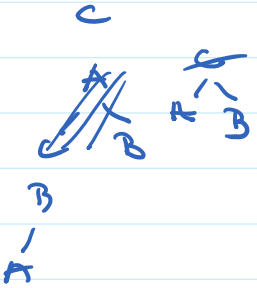




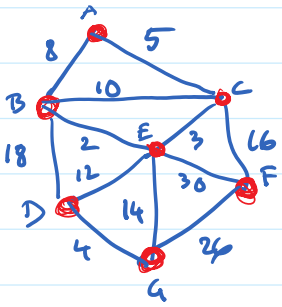
T = { }

T = { (-, D)

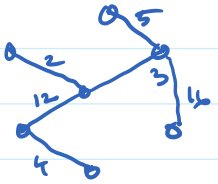
2, 7, 5



Ex: Graph



Kruskal



Prim