COMP20003 Workshop Week 12

1	MST & Greedy Algorithm for building MST
2	Prim's Algorithm
3	Kruskal's Algorithm
4	`2b|!2b` : Past exams OR ass2

MST & Greedy Algorithm

Task: give a connected, weighted graph `G= (V, E ,w)`, find a MST of it.

What's a spanning tree? What's a MST? Can `G` have more than one MST? Which algorithms?

Greedy algorithm:

A myopic policy of always taking the "best" bite in each step.

In many cases it's the best policy!

Dijkstra's algorithm is greedy.

Main idea of Dijkstra's Algorithm (shortest part from s to all other nodes):

```
init dist[s]= 0, dist[u]= for other nodes u
H= set of nodes u and dist[u]
while ( H not empty )
   remove u that have smallest dist from H
   output: dist[u] as shortest path for u
   use knowledge from u to update remaining elements in H
}
```

MST

Task: give a connected, weighted graph `G= (V, E ,w)`, find a MST for G.

What's a spanning tree? What's a MST? Can `G` have more than one MST?

Which algorithms? Complexity = ?

Which algorithm is better for:

- dense graphs
- sparse graphs

MST & Greedy Algorithm

Greedy algorithm can be used for the MST task, for example:

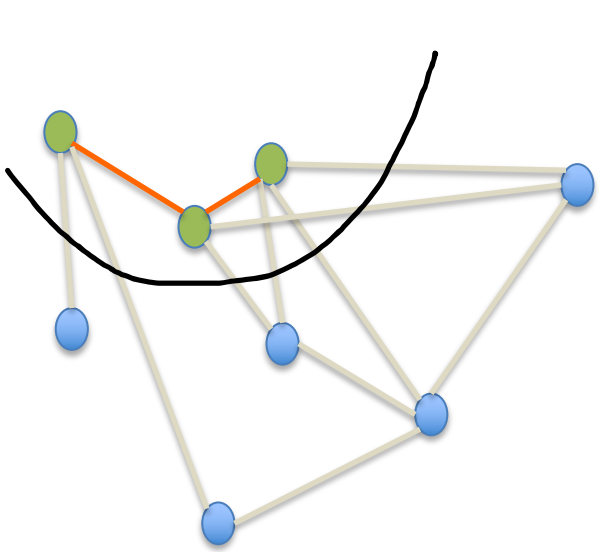
Prim: MST is set of vertices	Kruskal: MST is set of edges

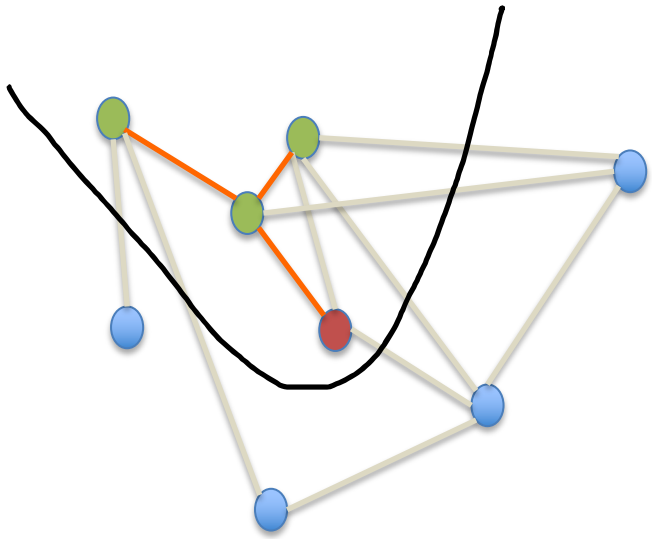
`T= any vertex`
`while ( |T| < |V|):`
`  add to T the vertex that`
`  has least distance to T`
 	
Note: distance between node u and set T is defined as the minimal distance between u and any member of T
	`T= EMPTY SET OF edges`
	`while ( |T| < |V|-1):`
	`  add to T the lightest edge`
	`  that doesn't make cycle in T`

Prim's Algorithm

- Consider a randomly-chosen vertex as the so-far MST.
- At each stage we, expand the MST-so-far by adding a vertex to that tree (which one? the one that is closest to the so-far MST).

Sounds familiar?





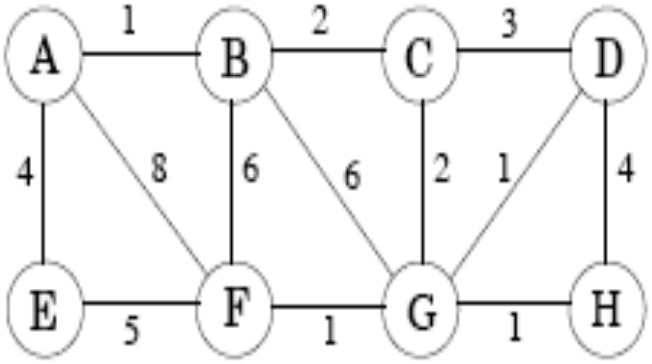
Prim's algorithm: operates vertex-by-vertex

Purpose: Find MST of (G= (V,E), w)

```
set cost[u]=∞, prev[u]=nil for all u
set X= empty, s= any vertex, cost[s]=0
H= makeheap(V)
while (H ≠ ∅):
    u= deleteMin(H)
  add u to X
    for each v that (u,v)∈E & v∈H:
        if (cost[v] > w(u,v)):
          cost[v]= w(u,v)
          prev[v]= u
          discrease cost[v] in H
```

Example

Suppose we want to find the minimum spanning tree of the following graph.



(a) Run Prim's algorithm; whenever there is a choice of nodes, always use alphabetic ordering (e.g., start from node *A*). Draw a table showing the intermediate values of the `cost` array.

Kruskal's algorithm

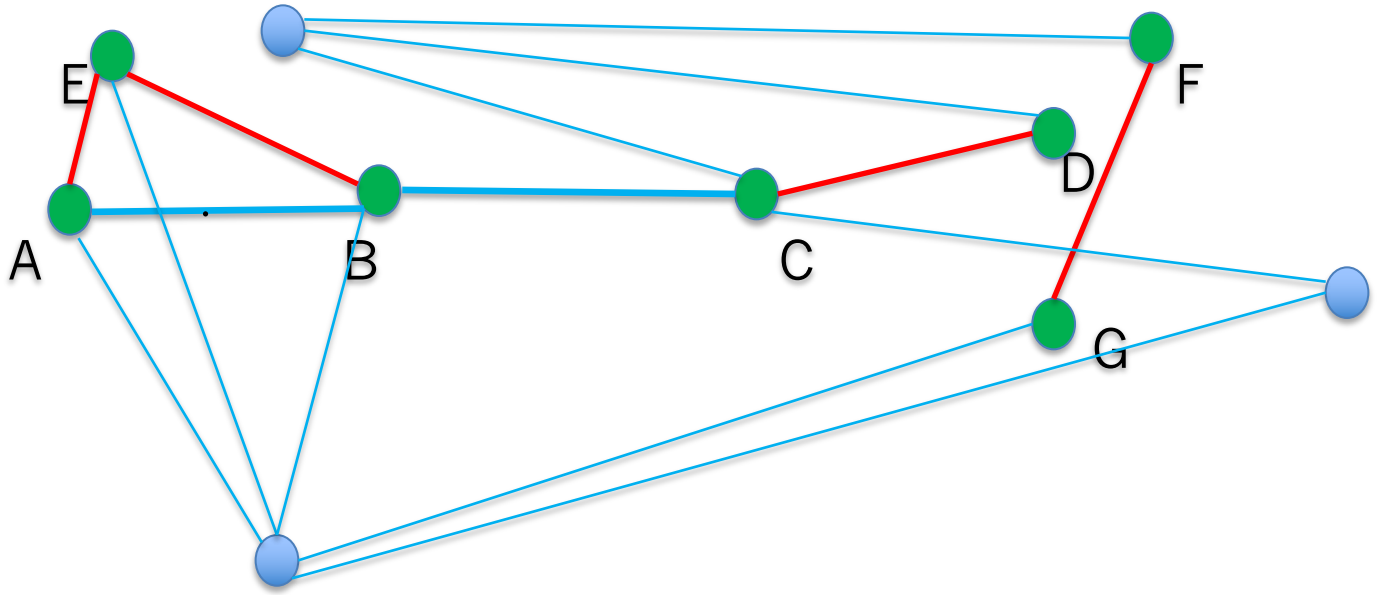
Purpose: Find MST of G= (V,E,w)

Prim's algorithm: processing node-by-node, ie adding a new node to MST at each step.

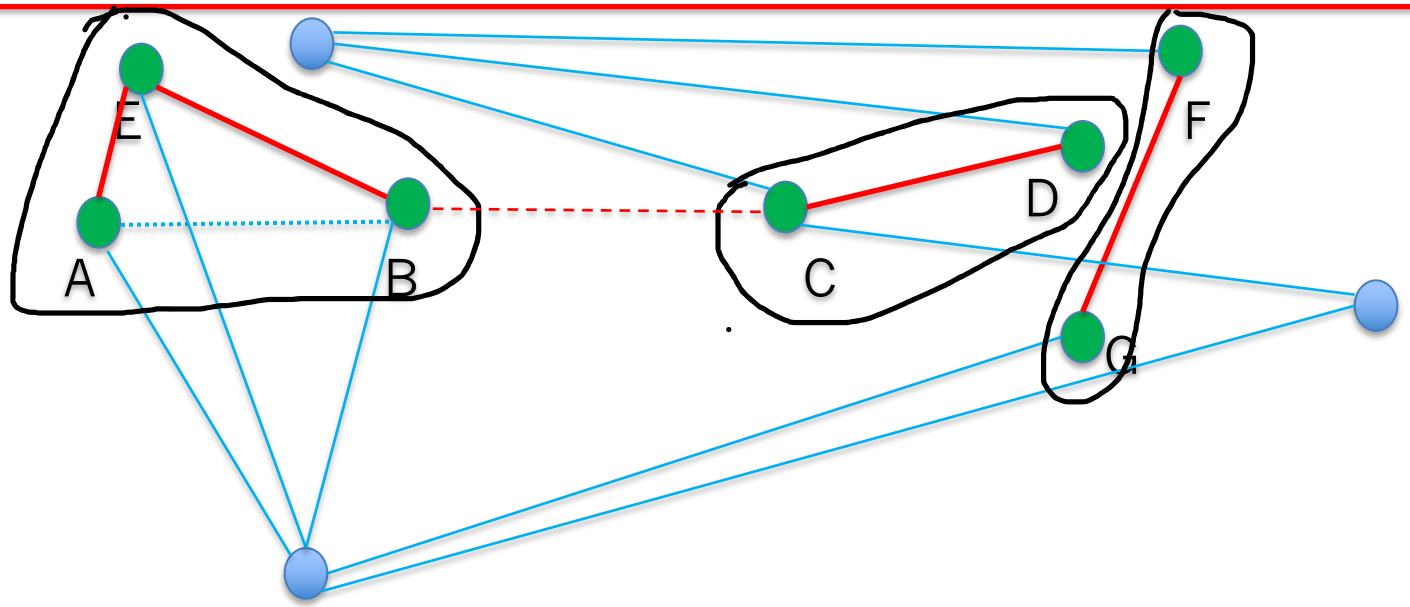
Kruskal's algorithm: operates edge-by-edge.

```
0  set MST-so-far to empty
3 for each (u,v), in increasing order of weight:
4     if  ( (u,v) does not form a cycle in MST-so-far):
5         add edge (u,v) to MST
```

How do we implement?



Suppose that the above 7 green vertices and 4 red edges form our MST-so-far. The next lightest are AB (should be rejected), then CD. *How can we recognize that inclusion of AB would create a cycle in the MST-so-far?*



How can we recognize that inclusion of AB would create a cycle in the MST-so-far?

We consider each green connected component a set of nodes.

(A,B) should not be added because A and B belong to the same set.

After adding (B,C) the sets ABE and CD are joined into ABCDE.

Needed: - Operation Find(u) : the set node u belongs to

- Operation Union(u,v) : join the disjoin sets of u and v into a single set

- an ID for each set ?

Kruskal's algorithm

Implement step 4 using disjoint set:

Make $|V|$ disjoint subsets, each contains a single node of V.

If we decide to add (u,v) into the MST, then we join the respective subsets together.

If u and v belong to a same subset, we can no more add (u,v) to the MST.

The algorithm stops when we have V-1 edges in MST

We an implement subsets as trees (note: not the same as MST). Operations:

`makeset(u)`- return a tree that contains single u

`find(u)` – return the root (means, ID) of the tree that contains u;

`union(u,v)` – joins trees containing u and v.

Kruskal's algorithm

Purpose: generate MST with Efficient checking for cycle in finding MST

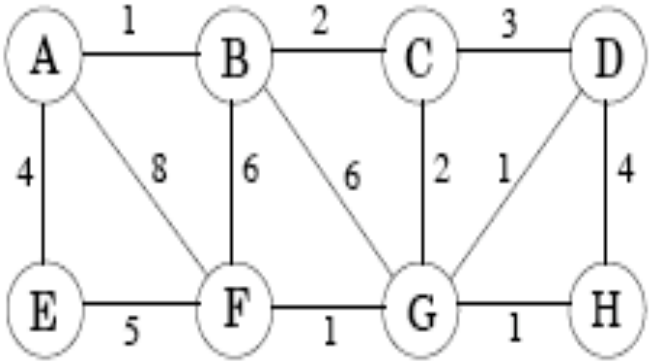
```
0 set X= empty. X is the MST-so-far
1 E1 = E, but sorted in increasing order of weights
2 for each u: makeset(u) (build single-element set {u})
3 for each edge (u,v) in E1 (in increasing order of weight):
4   if  (find(u) ≠ find(v) ):
   //if  (u & v do not belong to a same set):
5     add edge (u,v) to X
6   union(u,v)
```

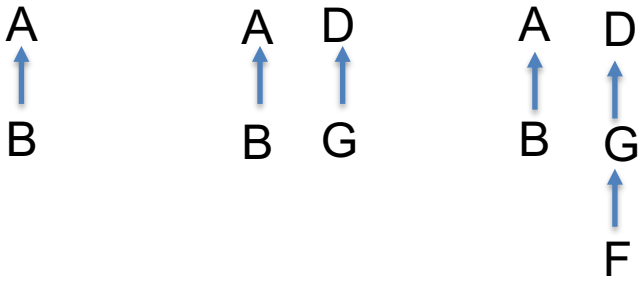
Watch online lecture for how to actually implement `makeset(u)`, `find(u)`, `union(u,v)`

→ We can implement with O(V+ElogV) for steps 3-6

→ The cost of KA is dominated by ElogE of the sorting phase in step 0

Example (Kruskal's)





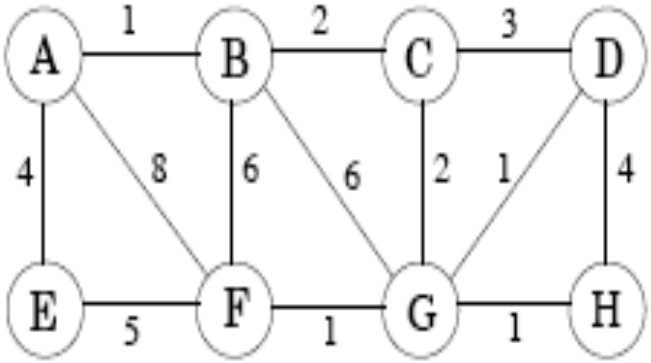
Edges in MST	A	B	C	D	E	F	G	H
	0	1	2	3	4	5	6	7
A---B	0	0	2	3	4	5	6	7
D---G	0	0	2	3	4	5	3	7
G---F	0	0	2	**3**	4	**6 ?**	3	7

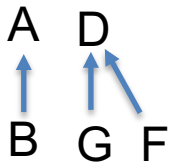
Note: number in table represents tree ID, if a tree ID is the same as node ID, then the node is the root of its tree

Rules:

1) join smaller tree to bigger tree (*weighted*, or *join-by-rank* optimization)

2) when joins, joins to the root



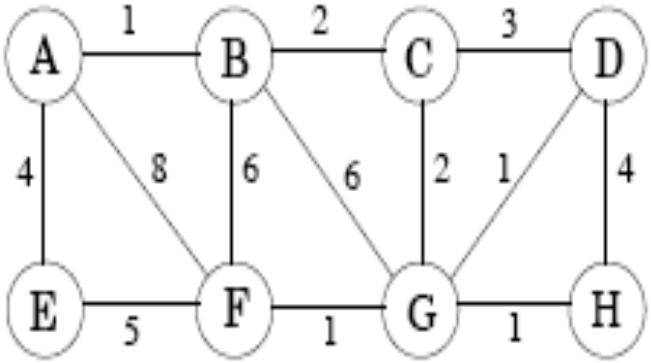


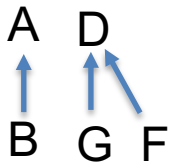
Edges in MST	A	B	C	D	E	F	G	H
	0	1	2	3	4	5	6	7
A---B	0	0	2	3	4	5	6	7
D---G	0	0	2	3	4	5	3	7
G---F	0	0	2	3	4	6 ?	3	7
G---F	0	0	2	3	4	**3**	3	7

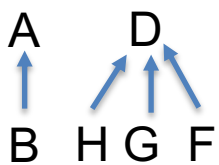
Rules:

1) join smaller tree to bigger tree (*weighted*, or *join-by-rank* optimization)

2) when joins, joins to the root





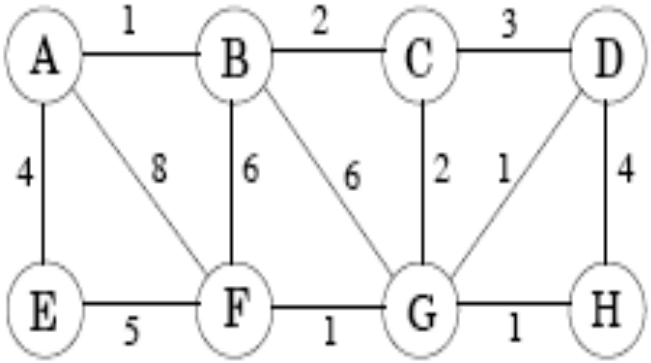


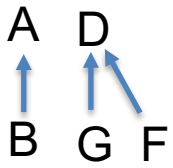
Edges in MST	A	B	C	D	E	F	G	H
	0	1	2	3	4	5	6	7
A---B	0	0	2	3	4	5	6	7
D---G	0	0	2	3	4	5	3	7
G---F	0	0	2	3	4	**3**	3	7
G---H	0	0	2	3	4	3	3	3

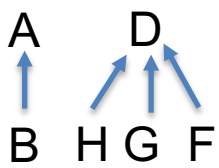
Rules:

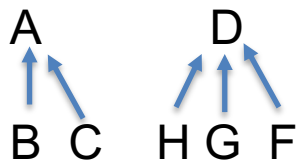
1) join smaller tree to bigger tree (*weighted*, or *join-by-rank* optimization)

2) when joins, joins to the root



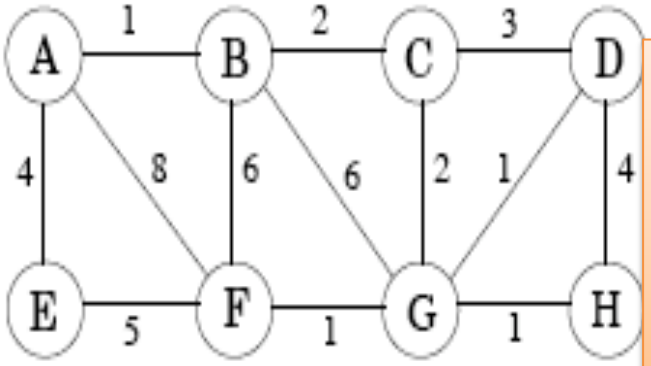




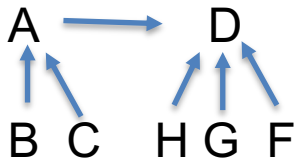


Edges in MST	A	B	C	D	E	F	G	H
	0	1	2	3	4	5	6	7
A---B	0	0	2	3	4	5	6	7
D---G	0	0	2	3	4	5	3	7
G---F	0	0	2	3	4	**3**	3	7
G---H	0	0	2	3	4	3	3	3
B---C	0	0	0	3	4	3	3	3
C---D	3	0	0	3	4	3	3	3

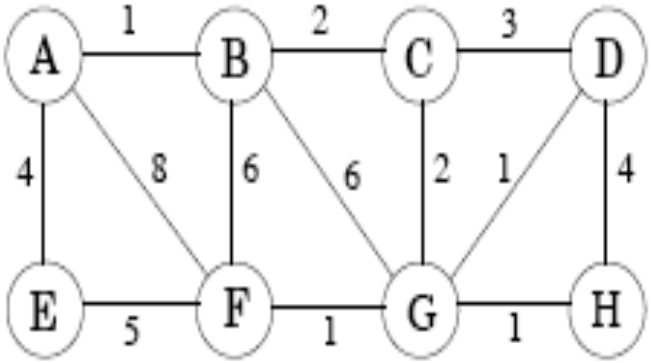
Rules:
3) Path compression



- Now, the pathlen of B,C is >1
- In the future, if we call find(B) we will have find(B)= D, and at that time we will make B point directly to D. That is called path compression.



Edges in MST	A	B	C	D	E	F	G	H
	0	1	2	3	4	5	6	7
A---B	0	0	2	3	4	5	6	7
D---G	0	0	2	3	4	5	3	7
G---F	0	0	2	3	4	**3**	3	7
G---H	0	0	2	3	4	3	3	3
B---C	0	0	0	3	4	3	3	3
C---D	3	0	0	3	4	3	3	3



After adding A---E to the MST-so-far, the latter has enough V-1 egdes, and we stop. Note that in our tree-array, there is only a single tree at this stage.

Edges in MST	A	B	C	D	E	F	G	H
	0	1	2	3	4	5	6	7
A---B	0	0	2	3	4	5	6	7
D---G	0	0	2	3	4	5	3	7
G---F	0	0	2	3	4	**3**	3	7
G---H	0	0	2	3	4	3	3	3
B---C	0	0	0	3	4	3	3	3
C---D	0	0	0	3	4	3	3	3
A---E	3	0	0	3	3	3	3	3

Justify the Complexity

	Prim	Kruskal
General	(E+V) log V	E log E
Dense Graph	E log V	E log E
V<<E, Prim's is faster		
Sparse Graph	V log V	V log V
Kruskal's is faster because of the data structures		

Lab Time:

- Finish ass2, or

- go through some questions, especially short questions, in sample exam paper

- give questions to the in-lazy-mode Anh

Minimal requirements for Report

Minimal content:

- a table, exactly as described in the spec, and please just include everything in a *single table*.
- a graph that plots the number of pegs left as a function of the number of pegs initially in the board (for example, one curve or bar group for one budget level). Again, I'd like to see a single graph for this one.
- a short discussion on how the budgets affect solution quality in the last 4 layouts.

Note: In our workshops I said that I expect a kind of nice report which implies the inclusion of (in addition to the above content) introduction, experiment descriptions, other discussion, and conclusion. These additional parts are no longer required. That is to agree with @1226 in Piazza FAQ!

Report size: should be no more than 2 pages, including tables/figures.

What if you already wrote a nice report?

Obviously, you are welcome to submit that report and I will enjoy reading it. But you should check to make sure that you include the 3 components listed above.