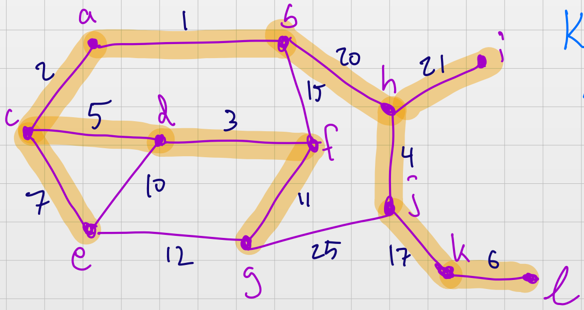
CS 5112

Algorithms for Applications

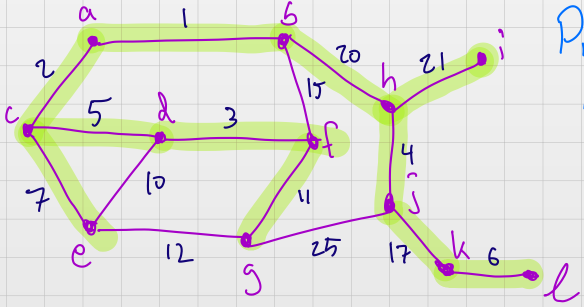
Minimal Spanning Trees



Kruskal's Algorithm

Assuption: Distinct edge weights

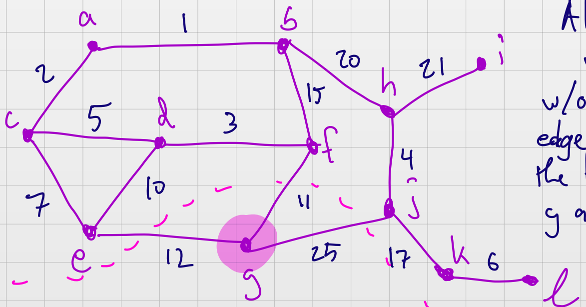
Prim's Algorithm



Assuption: Distinct edge weights

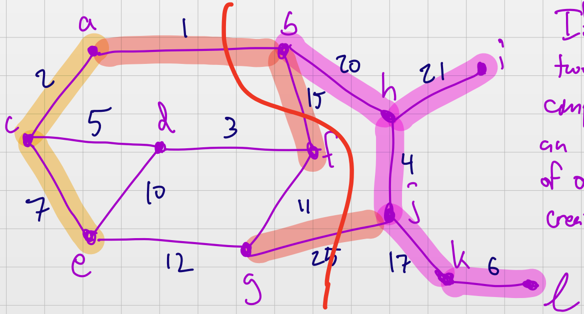
WTS: Output is correct.

1. Lowest weight ("minimal")
2. Tree
   2a. Connected ✓
   2b. No cycles ✓
3. Spanning - Every node is connected ✓



Algorithm won't stop w/o adding an edge to g, since the first edge to g added can't create a cycle.

Assuption: Distinct edge weights



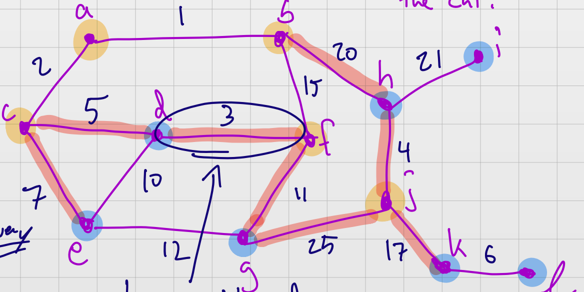
If there are two disconnected components, adding an edge on the cut of one of them can't create a cycle.

$\Rightarrow$ algo hasn't finished

Assuption: Distinct edge weights

Cut Property

For any $S \subseteq V$, every MST contains the lowest weight edge in the cut.



⇓

exist a list of edges that have to be in every MST

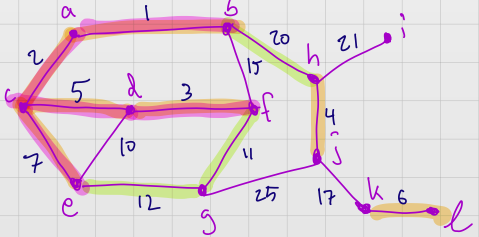
lowest weight edge in the cut

Use the Cut Property (Kruskal's)

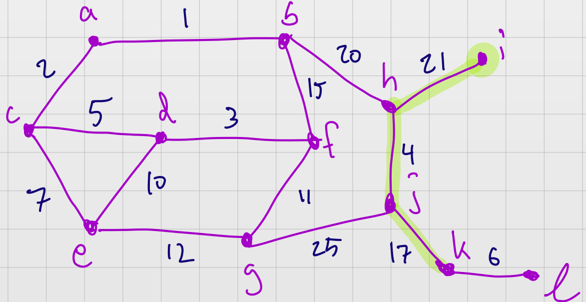
Suppose Kruskal's algorithm adds an edge $e=(v,w)$. Right before we add e, look at the set S of all nodes with a path to v.

In this example $e=(f,g)$

S is the pink highlight



Use the Cut Property (Prim's)

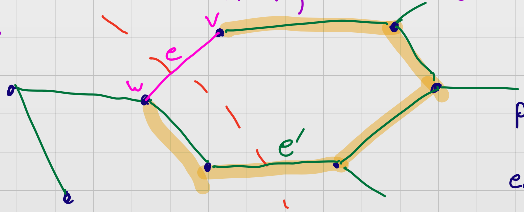


Cut Property

Let e be the lowest weight edge in the cut of S.
Let T be any spanning tree.

WTS: If $e \notin T$, then T is not minimal.

Idea is to exchange e for e'.

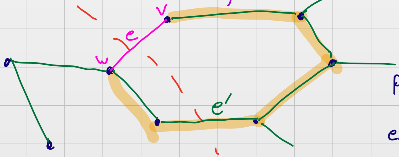


There must be a path from v to w in T. Pick a edge on the path in the cut,

Let e be the lowest weight edge in the cut of S.
Let T be any spanning tree.

WTS: If $e \notin T$, then T is not minimal.

Idea is to exchange e for e'.



There must be a path from v to w in T. Pick a edge on the path in the cut,

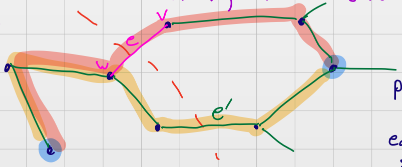
Let T' be T with e' removed and replaced with e.

1. lower weight ✓
2. tree
3. spanning

Let e be the lowest weight edge in the cut of S.
Let T be any spanning tree.

WTS: If $e \notin T$, then T is not minimal.

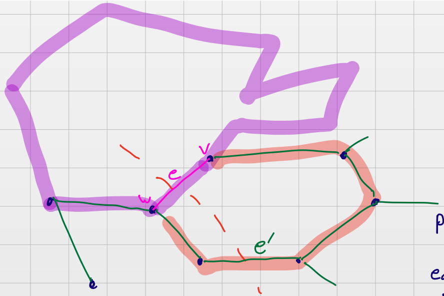
Idea is to exchange e for e'.



There must be a path from v to w in T. Pick a edge on the path in the cut.

Tree: 1. Connected Can create a path in T' using the path from v to w in T and e.

2. No cycles



There must be a path from v to w in T. Pick a edge on the path in the cut.

Tree: 1. Connected — Can create a path in T' using the path from v to w in T and e.

2. No cycles — Any cycle in T' must contain e. So a similar argument gives a cycle in T.