CSC 226

Algorithms and Data Structures: II

More MST

Tianming Wei
twei@uvic.ca
ECS 466

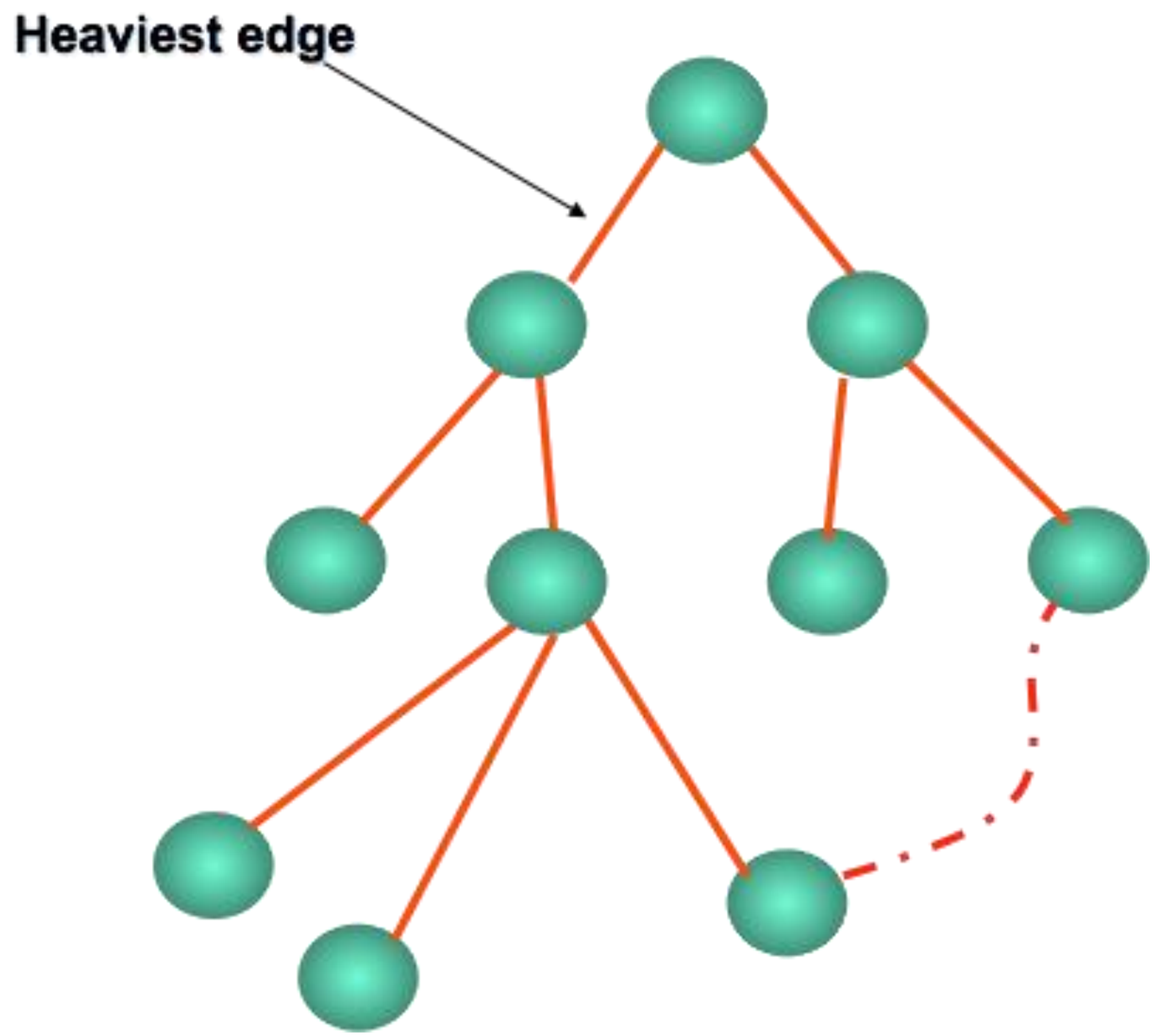
Two basic properties for minimum spanning trees

- Cycle property
- Cut property

Cycle property

- Let C be any cycle in weighted graph G with distinct edge weights. Let e be the heaviest edge in the cycle.
- Then the minimum spanning tree for G does not contain e.

Cycle property

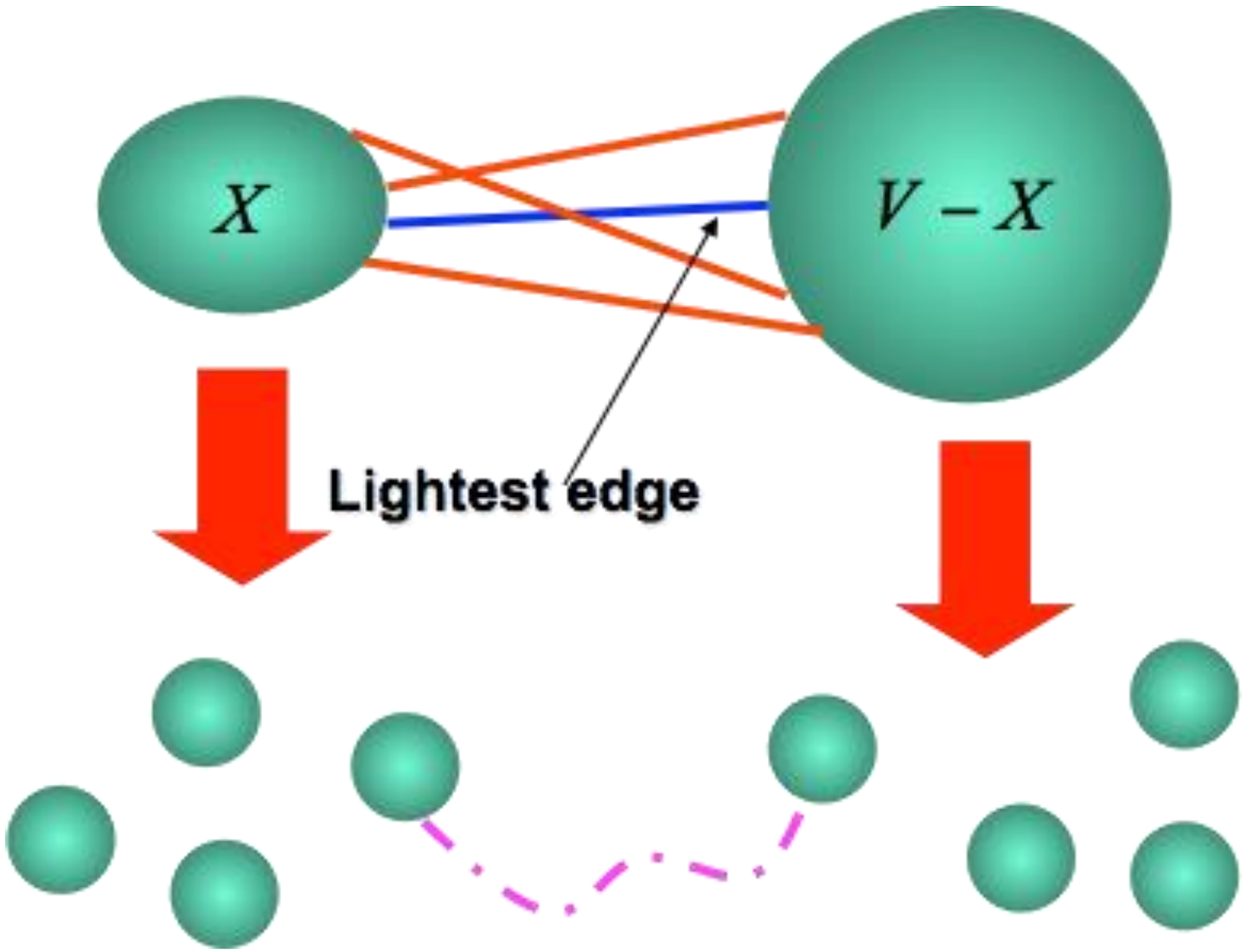


Cut Property

- Let V' be any proper subset of vertices in weighted graph $G = (V, E)$, and let e be the lightest edge that has exactly one endpoint in V'

- Then the minimum spanning tree T for G contains e.

Cut Property



More Practice Questions

1. Let T be a minimum spanning tree of a graph G, and let L be the sorted list of the edge weights of T . Show that for any other minimum spanning tree T' of G, the list L is also the sorted list of edge weights of T'.

2. Given a graph G and a minimum spanning tree T , suppose that we decrease the weight of one of the edges in T . Show that T is still a minimum spanning tree for G.

More Practice Questions

- More formally, let T be a minimum spanning tree for G with edge weights given by weight function w. Choose one edge $(x, y) \in T$ and a positive number *k*, and define the weight function *w'*

$$w'(u, v) = \begin{cases} w(u, v) & \text{if } (u, v) \neq (x, y) , \\ w(x, y) - k & \text{if } (u, v) = (x, y) . \end{cases}$$

- Show that T is a minimum spanning tree for G with edge weights given by *w'*.

More Practice Questions

3. Give an algorithm for finding the minimum spanning tree in Kruskal's algorithm can return different spanning trees for the same input graph G, depending on how it breaks ties when the edges are sorted into order. Show that for each minimum spanning tree T of G, there is a way to sort the edges of G in Kruskal's algorithm so that the algorithm returns T.

More Practice Questions

A ***bottleneck spanning tree*** T of an undirected graph G is a spanning tree of G whose largest edge weight is minimum over all spanning trees of G. We say that the value of the bottleneck spanning tree is the weight of the maximum-weight edge in T.

a. Argue that a minimum spanning tree is a bottleneck spanning tree.

Part (a) shows that finding a bottleneck spanning tree is no harder than finding a minimum spanning tree. In the remaining parts, we will show how to find a bottleneck spanning tree in linear time.

b. Give a linear-time algorithm that given a graph G and an integer b, determines whether the value of the bottleneck spanning tree is at most b.

a. To see that every minimum spanning tree is also a bottleneck spanning tree. Suppose that T is a minimum spanning tree. Suppose there is some edge in it (u, v) that has a weight that's greater than the weight of the bottleneck spanning tree. Then, let V_1 be the subset of vertices of V that are reachable from u in T, without going though v. Define V_2 symmetrically. Then, consider the cut that separates V_1 from V_2. The only edge that we could add across this cut is the one of minimum weight, so we know that there are no edge across this cut of weight less than $w(u, v)$. However, we have that there is a bottleneck spanning tree with less than that weight. This is a contradiction because a bottleneck spanning tree, since it is a spanning tree, must have an edge across this cut.

b. To do this, we first process the entire graph, and remove any edges that have weight greater than b. If the remaining graph is selected, we can just arbitrarily select any tree in it, and it will be a bottleneck spanning tree of weight at most b. Testing connectivity of a graph can be done in linear time by running a breadth first search and then making sure that no vertices remain white at the end.