-------flashes-------------

The blog aims to achieve whats going on in my mind. I try to put down everything I think, my questions, my answers, my frustrations whatever. When you are thinking about something and suddenly something strikes and the whole scenario changes, the blog is an attempt to catch that flash.

About Me

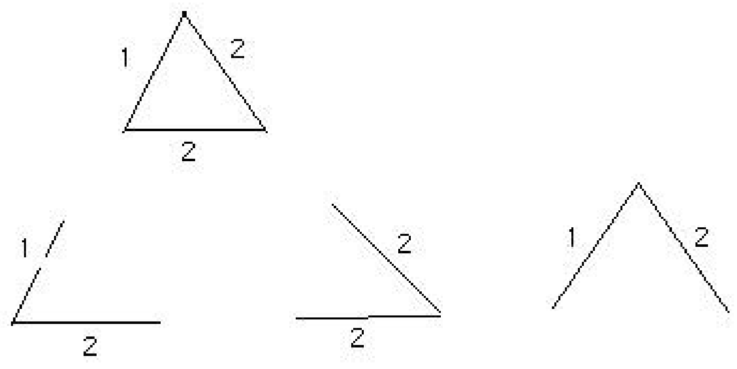
Goldy Blank

View my complete profile

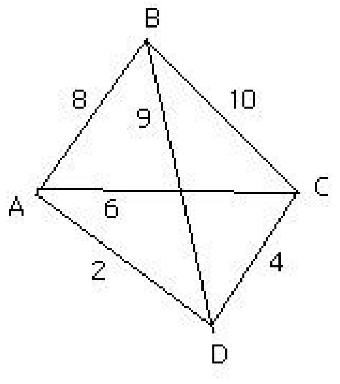
Saturday, June 5, 2010

Everything about Bottleneck Spanning Tree

In this post we will take a detail look on BottleNeck Spanning Tree (BNST). But first, lets understand the definition of BNST.
Suppose we have a graph G(V, E). Now take out all the spanning trees of G. In all the Spanning trees, the maximum weight edge is called the bottleneck edge [because this edge is some sort of bottleneck for us. The ST cost can be reduced if we reduce this edge]. Of all the spanning trees, pick the tree whose bottleneck weight is such that its the smallest of all other bottleneck weights of other trees, i.e there exist no other spanning tree which has its bottleneck value lesser than the bottleneck value of the tree you picked up.
Lets take an example. Consider the graph below. Also shown are its spanning trees.



The bottleneck edge value is 2 in each case and if I pick any tree, there exist no other tree having its bottleneck value less than 2. Hence all three spanning trees are also BNST [NOTE: the middle spanning tree is a BNST but not a MST, the other two are].
Another example can be as shown.



Consider all its spanning trees. We can notice that spanning trees can have either of AB, BD or BC edge to include the B vertex(or more than one). So 8,9,10 are the heaviest edge that one of the spanning trees can contain and among all the spanning trees, there is no spanning tree whose maximum edge weight is less than 8. So pick any spanning tree with AB edge in it and it will be a BNST. Note that BNST is only concerned with its maximum edge weight.

Problem 1: Is an MST a BNST also? Is a BNST an MST also?
Solution: There are two ways to proving this.

First, by contradiction. Lets say an MST is not a BNST i.e for graph G(V, E), lets say there is an MST, T and a BNST, T'. Assume the maximum edge in T is e and that of T' is e'.
W(e)>W(e') because T' is a BNST and its by definition of BNST that all other spanning trees have their maximum edge weight greater than that of T'. That means if I add edge e to T', it will form a cycle and the cycle will have the maximum edge as e. By the red rule, the edge e cannot belong to any MST hence its contradicting. So the MST is same as BNST.

Second, Instead of proving MST=>BNST, lets prove that ~BNST=>~MST. Assume graph G has a spanning tree T that is not a BNST. Let e be the maximum weight edge in T. Lets say e connects two trees T1 and T2. Since T is not a BNST, there exists another edge e' which also connects T1 and T2 and W(e')
T=T1+T2+e
T'=T1+T2+e', where T' is the BNST.

Sum of tree edge weights, S(T)>S(T'), so T cannot be the MST. Now T is not a BNST, implies it cannot be the MST as well. Hence proved.

The second part of the question can be proved using the counter example explained above.

Problem-2: Give an algorithm to create a BNST of a given graph G(V, E).
Solution: The solution is very simple. For this, starting with the largest edge, remove edges from G one by one. If removal of any edge disconnects G, then keep that edge and continue the process with the next edge in sequence. Continue this until all vertices are covered. Now we

have a spanning tree which is a BNST. Note that it is an exact opposite of Kruskal's algorithm to find the MST of G. Moreover any MST finding algorithm is also fine because MST is a BNST as well.
Another method is, find the median edge weight and remove all edges with weight more than the median. If the remaining graph is connected recursively repeat the process. If the graph is not connected then there are connected components in the G. Shrink these components into single vertices with edges coming out is similar to original graph and reconstruct till G is connected again. Repeat the whole process till result is achieved. It is O(|E|+|V|log|V|).

Problem 3: Give a Linear time algorithm to determine if a graph G(V,E) contains a BNST with its maximum edge <= b, where b is a given constant.
Solution: We remove all the edges with weights > b from G. Then we check is G is still connected. If yes, then such a BNST is possible, else not. We can use DFS to check connectedness of G which is linear.

References:
http://pages.cpsc.ucalgary.ca/~dcatalin/413/t4.pdf
http://www.fongboy.com/classes/cs180/hw8-2.pdf
http://www.cse.nd.edu/courses/cse413/www/hw7_sol.pdf
http://www.cs.dartmouth.edu/~ac/Teach/CS105-Winter05/Homeworks/hw2-sol.pdf

Stumble it!

Posted by Goldy Blank at 5:35 AM

6 comments:

Blog SuffocatedByRules February 7, 2013 at 1:47 AM

Hello there! Are you mostly an often online visitor or maybe you are more into face to face communication?

Reply

Anonymous September 14, 2013 at 9:06 PM

Thanks man, this really helped me understand this tricky thing;
"A MST is necessarily a MBST (provable by the cut property), but a MBST is not necessarily a MST."
It is simple but somehow I was not able to figure out, your blog definitely helped me.

Reply

The_Blue_Jaguar September 18, 2013 at 2:51 AM

Brilliant post bro. Cormen problems solved:D

Reply

Goldy Blank September 24, 2013 at 9:52 PM

Glad I can help.

Reply

Goldy Blank September 24, 2013 at 9:52 PM

This comment has been removed by the author.

Reply

Timothy Swan March 30, 2014 at 5:08 PM

Goldy Blank,
Nice information! I think the BNST can be found in O(|E|+|V|) since selection for median can be done in linear time and the recurrence for checking whether an edge is the bottleneck divides the problem size by two each time. (i.e. as opposed to merge sort which divides the problem size by two but also doubles the number of problems, keeping the problem size the same) I was assigned homework where we were required to determing minimum bottleneck weight in linear time.
https://courses.engr.illinois.edu/cs473/hw/hw_07.pdf

Reply

Enter your comment...

Comment as: Google Accou ▼

Publish Preview

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)