Subject: Finding the new extreme value in extractBinomial.
Posted by MattM on Wed, 29 Mar 2017 02:20:48 GMT

I don't understand how we are supposed to find the new extreme value after removing the extreme node in extractBinomial.

It seems we should loop through the rootlist from "degree 0 to degree log(size of b) / log(2)." However, in this looping, I think we need to use the heap's compare function to determine the most extreme rootlist value. Say the heap only has one subtree in the rootlist with degree 2, then there is a NULL pointer in the 0 and 1 place in the rootlist. The "comparator function has to be implemented so that it treats a null value as more extreme than any other value." So, the most extreme value would be set to an empty spot in the rootlist.

I suppose one way to circumvent this possible issue would be to ignore spots in the rootlist that are NULL. Am I missing something here, or is this how it should be done?

Subject: Re: Finding the new extreme value in extractBinomial.
Posted by SSinischo on Wed, 29 Mar 2017 02:27:19 GMT

I never even ended up having to calculate log(b->size)/log(2), I just looped through every subtree in the rootlist. I think it turns out to be the maximum size of the rootlist anyway. If there is no subtree, I skip over that index.

Also, when it says "comparator function has to be implemented so that it treats a null value as more extreme than any other value.", value actually means key. The key of this binomial heap is the distance of a vertex. The comparator treats vertex->distance as an extreme value if it is NULL, not the vertex itself.

- 

Subject: Re: Finding the new extreme value in extractBinomial.
Posted by MattM on Wed, 29 Mar 2017 03:10:24 GMT

Thanks!

Subject: Re: Finding the new extreme value in extractBinomial.
Posted by lusth on Fri, 31 Mar 2017 13:47:55 GMT

MattM wrote on Tue, 28 March 2017 21:20 I don't understand how we are supposed to find the new extreme value after removing the extreme node in extractBinomial.
It seems would should loop through the rootlist from "degree 0 to degree log(size of b) / log(2)." However, in this looping, I think we need to use the heap’s compare function to determine the most extreme rootlist value. Say the heap only has one subtree in the rootlist with degree 2, then there is a NULL pointer in the 0 and 1 place in the rootlist. The "comparator function has to be implemented so that it treats a null value as more extreme than any other value." So, the most extreme value would be set to an empty spot in the rootlist.

I suppose one way to circumvent this possible issue would be to ignore spots in the rootlist that are NULL. Am I missing something here, or is this how it should be done?

The size of the binomial heap is the number of nodes, not the number of subtrees.

It is permissible to loop through the rootlist using the size of the rootlist as the limit (although using the log calculation in the presence of deletions is slightly more efficient in a practical sense).