Subject: The Linked List Union Function  
Posted by bmbaker1 on Wed, 16 Nov 2016 22:54:29 GMT  
View Forum Message <> Reply to Message

In the pseudocode, for the binheap union() function, it says:
Quote: merge the root list of the donor into the root list of b (via linked-list union)
Where exactly do we merge the two lists? Do we append the the donor's root list in front of b's root list?
Do we append the donor's root list to the back of b's? Is there a specific spot in the middle of either root list where the two need to be merged?

P.S. - This is my pre apology for the influx of, possibly, silly questions that I ask over the next few weeks... There will be MANY... Sorry...

Subject: Re: The Linked List Union Function  
Posted by btlindow on Thu, 17 Nov 2016 01:40:29 GMT  
View Forum Message <> Reply to Message

I don't believe that order matters, but if you incorporated some sort of numerical ordered insert into your linked-list, kudos. It might even make a ladder part of the project easier and more efficient (printing out the traversal).

My merge took two linked list objects, I repeatedly called removeNode from one linked list until it was empty and inserted that removed Node into the other. I believe my insert places the Node to the back of the linked list.

Subject: Re: The Linked List Union Function  
Posted by bmbaker1 on Thu, 17 Nov 2016 02:28:56 GMT  
View Forum Message <> Reply to Message

Thank you! I believe that I was focussing too much on the 'merge' part, thinking that everything needed to be merged in a certain order.

Subject: Re: The Linked List Union Function  
Posted by lusth on Thu, 17 Nov 2016 13:16:07 GMT  
View Forum Message <> Reply to Message

btlindow wrote on Wed, 16 November 2016 19:40 I don't believe that order matters, but if you incorporated some sort of numerical ordered insert into your linked-list, kudos. It might even make a ladder part of the project easier and more efficient (printing out the traversal).

My merge took two linked list objects, I repeatedly called removeNode from one linked list until it was empty and inserted that removed Node into the other. I believe my insert places the Node to the back of the linked list.
This is very slow (linear). You can do this task in constant time. If you have non-empty lists A and B with head and tail pointers:

A->tail->next = B->head;
B->head->prev = A->tail;
A->tail = B->tail
A->size = A->size + B->size;
B->head = 0;
B->tail = 0;
B->size = 0;

Subject: Re: The Linked List Union Function
Posted by sestephens on Sat, 19 Nov 2016 21:43:46 GMT
View Forum Message <> Reply to Message

You've linked the list in the middle, but the new head and tail don't point to each other. Shouldn't the code be as follows?

A->tail->next = B->head;
B->head->prev = A->tail;
B->tail->next = A->head;  //my addition
A->head->prev = B->tail;  //my addition
A->tail = B->tail
A->size = A->size + B->size;
B->head = 0;
B->tail = 0;
B->size = 0;

Or are my two additions unnecessary?

Subject: Re: The Linked List Union Function
Posted by rokohler on Mon, 21 Nov 2016 15:38:20 GMT
View Forum Message <> Reply to Message

Your additions are unnecessary for a non-circular linked list merge which Dr. Lusth is showing above. If you are using a circular linked list you would not need a tail pointer but your additions would be necessary.