I am confused as to how the type casting and using the comparator function as a parameter is supposed to work.

Following the pseudo-code, my newBinHeap() function definition looks like this:
binheap *newBinHeap(int (*)(void *, void *));
while my newBinHeap() implementation looks like this:
binheap *newBinHeap(int comparator(void *a, void *b)){ }
I understand that I am passing a function into newBinHeap as a parameter, but I do not understand how to set the comparator field in my binheap struct to the given comparator. Inside of my binheap struct, I have
int comparator;
so in the newBinHeap() function, I try:
b->comparator = comparator;
which produces a type cast error.
then I tried:
b->comparator(a, b);
but the variables a and b are not recognized...

Can anyone explain what is actually going on when I send a function as a parameter? How do I set a variable to the return value of the given comparator function? Am I thinking about this completely wrong?

Your binheap constructor implementation should be:

binheap *newBinHeap(int (*)(void *, void *)) {...}

Your binheap struct should have the field:

int (*)(void *,void *);

The assignment in the constructor would be:

b->comparator = comparator;

The types have to match everywhere, just like any other type.
Ah! That makes an incredible amount of sense... And just to be clear, after newBinHeap() is
called and ran, will the value of comparator in the struct be and integer(1, 0, or -1) depending on
what the return value of the given comparator function is?

Subject: Re: Confusion Regarding Comparator Setting in binheap
Posted by lusth on Tue, 15 Nov 2016 01:50:21 GMT

No, the value of the comparator stored in the struct is a function. Calling that function will yield an
integer value:

// x and y are nodes with generic value pointers
int result = b->comparator(x->value,y->value);

Subject: Re: Confusion Regarding Comparator Setting in binheap
Posted by bmbaker1 on Tue, 15 Nov 2016 02:15:12 GMT

Understood. Thank you!

Subject: Re: Confusion Regarding Comparator Setting in binheap
Posted by BentHam on Fri, 25 Nov 2016 03:02:40 GMT

I have a question regarding implementation of the comparator function. I know that we are to
return 0 in the event that the vertex values are tied; in that case, the tie is broken by the
(semi-arbitrary?) vertex ID numbers, which, if I understand correctly, are assigned as the vertices
are read in from the input file.

Knowing the vertex ID value within the scope of vertexComparator, would it make sense to simply
deal with the tie case inside it? Or does the tie-breaker need to be settled within the binHeap call
to vertexComparator? For example,

int vertexComparator(void* a, void* b) {
    (...typecast...)
    if (...is smaller...)    return -1;
    else if (...is bigger...)    return 1;
    else {    // instead of return 0

return (a->ID < b->ID)? -1 : 1;
}
?
Or should that case be handled upon returning 0 to a higher function?

Subject: Re: Confusion Regarding Comparator Setting in binheap
Posted by btlindow on Fri, 25 Nov 2016 03:41:38 GMT
View Forum Message <> Reply to Message

I did all of my tie breaking inside of my comparator. Here is Dr. Lusth's recent post about how to handle tie breaking:

compare key values
  if they are equal
    compare predecessor numbers
  if they are equal
    compare vertex numbers