.DSA 2 Program 2 Organization

Objective: create (binary) heap class that acts as a priority queue

Each node = {unique string ID, int. key, optional pointer (void)}

* Int key can be negative

Declare heap

* Capacity (max # of items) passed to constructor (never allowed to grow larger for assignment)

Provided with useHeap.cpp & Makefile (only change = flags)

To-do:

1. Correct hash.cpp file
2. Fill in missing heap.cpp & heap.h files

Heap class must include:

* Constructor: accepts integer rep. capacity of binary heap
* insert (public member func): inserts new item into heap
* deleteMin (public member func): removes & returns item w/ lowest key from heap
* setKey (public member func): provides both increaseKey & decreaseKey functions
  + use hash table
* remove (public member func): delete item w/ spec. ID from heap
  + use hash table
* additional private data members & member functions
  + int -> capacity
  + int -> current size of heap
  + vector -> contains node objects
  + hash table (construct hash table to be large enough to not rehash based on heap capacity)
    - items only lazily deleted from hash table (still take up space)

heap.h must include:

* declaration of heap class (including public/private members/functions)

heap.cpp must include:

* implementation of heap class

Order of Operations:

Take in capacity input -> construct heap

* set vector size = capacity
* initializes hash table

5 options

* if option 1: insert new item [insert(string id, int key, void \*pv)]
  + insert into vector starting from [1] and to the right
  + take in string ID & integer key -> stores it as node in vector
    - a node = {string ID, int key, optional pointer}
  + use string as hashvalue to insert string and location of node into hashtable
  + apply percolate up to compare inserted node against parent(s) in heap
    - If swaps need to occur, switch string id and int key in heap & update location of node associated with string in hashtable using setpointer (location of string in hashtable never needs to move bc just used as lookup)
      * Setpointer() used to update pointer values in ht
* if option 2: set key of spec. item [setKey(string id, int key)]
  + look up id string in hashtable, find location of node holding string in heap (use ht function getpointer), & change key value to input key
  + check if percolation needs to happen: if less than parent, percolate up, else if greater than children, percolate down
* if option 3: delete spec. item [remove(str ID, int \*pkey, void \*ppdata)]
  + use id to look up in hashtable, mark as isdeleted (lazy deletion done thru ht remove function), and find node with id in heap (getpointer) and fully deleted node in heap
  + treat like deletemin – insert most recent node and percolate up/down
  + if \*pkey provided, write key of node being deleted to address of key
  + if \*ppdata provided, write associated void pointer to address
* if option 4: deleteMin(string \*pID, int \*pKey, void \*ppData)
  + don’t need key because always choosing minimum
  + look up id of minimum node in hashtable and mark isdeleted then deleted in heap
  + bring latest inserted node to root then percolate up
  + write deleted values to addresses
* if option 5: Quit
  + quits program

Q: Why do we need the hash table?

* Access to an item in a vector is linear, but its constant in hash table (for working w/ specific items)

Q: How should I insert items into hash table?

* Use string ID to find location, then store location of the entire node containing string ID in heap

Q: How to percolate down for vector form? (applies to insert)

* Compare w/ children (left first) & switch if larger up to leaf node

Q: How to percolate up for vector form? (applies to remove)

* Compare w/ parent and smaller if smaller than parent up to root