CS375 Design and Analysis of Algorithms

Fall 2020

***Programming Assignment* 2**

Assigned: October 2, 2020
Due: Midnight Thursday, October 15, 2020

Prelude

Reality TV Game Shows such as Survivor, American Idol, Master Chef, and more, are often structured similarly to one another: contestants compete weekly, and after each competition, one or more contestants are eliminated from the show until a single contestant remains and is crowned the winner. Your job for Program 2 is to implement a data structure that can be used to simulate the progress of an elimination game. In particular, you will implement an extended min-heap to implement a priority queue.

Contestants lose and gain points for their performance in the game, thereby moving them (a) further down the heap, away from the root and toward safety, or (b) up toward the root, where they risk elimination. Periodically, the currently worst performing contestant is removed from the game by taking her or him off the top of the heap. In our case, unlike most shows, contestants may enter the game after it begins.

Your extended min-heap implementation should support the following operations, as described below:

findContestant	insertContestant	eliminateWeakest
earnPoints	losePoints	crownWinner
showContestant	showHandles	showLocation

Goal

The goal of the assignment is to implement a min-priority queue based on an "extended heap" data structure. Such a data structure supports a priority queue when the priority of any element in the queue may change. The elements in the extended heap have the following fields:

1. An id that uniquely identifies a contestant.
2. A contestant's point total.

The asymptotic running time to find an element with id k in a regular heap containing n elements is $O(n)$. Your extended heap should be able to find a contestant with id k in $O(1)$ time. Once a contestant node is found, its score can be changed in $O(\lg n)$ time. You may assume that each contestant receives a unique integer id between 1 and n, and that the maximum size n of the extended heap is given. To support *find()* in $O(1)$ time, add a *handle array*---an additional array that tracks the location of all the items in the extended heap. A contestant whose id is k resides in heap location *handle[k]*. When id k is not in the extended heap, the handle should contain a value that cannot be a legal index into the heap. Note that when a contestant's location in the heap changes, your program must update the handle array accordingly. For this assignment, indexing for all arrays starts with 1 (leave *heap*[0] and *handle*[0] unused).

Input: Your program should take two command line arguments:
1. The name of an input file.
2. The name of your output file.

Each input file is for the creation and manipulation of a separate extended heap. The first line of the input file will contain the maximum size of the heap n. The rest of the lines of the input file will contain a sequence of operations, one per line. Operations are listed and described below.

Output: The output file should contain for each operation a copy of the operation and the output generated in response. Your output file must be formatted EXACTLY (spelling, space, case, etc.) as required, for automated testing of your program. Your program should support the following operations. To implement them, you should find the corresponding extended heap operation:

1. **findContestant** <k>
   Find the element with id = <k> in *O*(1) running time.
   The output should be one of the following, as appropriate:
   a. "Contestant <k> is in the extended heap with score <s>."
   b. "Contestant <k> is not in the extended heap."
2. **insertContestant** <k> <s>
   Insert a new contestant with id = <k> and score = <s> into the extended heap.
   The output should be one of the following, as appropriate:
   a. "Contestant <k> inserted with initial score <s>."
   b. "Contestant <k> could not be inserted because the extended heap is full."
   c. "Contestant <k> is already in the extended heap: cannot insert."
3. **eliminateWeakest**
   Remove the contestant with the lowest current score..
   The output for this operation should be one of the following:
   a. "Contestant <k> with current lowest score <s> eliminated."
   b. "No contestant can be eliminated since the extended heap is empty."
4. **earnPoints** <k> <p>
   Add <p> points to the contestant with id = <k>.
   The output should be one of the following:
   a. "Contestant <k>'s score increased by <p> points to <s>."
   b. "Contestant <k> is not in the extended heap."
5. **losePoints** <k> <p>
   Take away <p> points from the contestant with id = <k>.
   The output should be one of the following:
   a. "Contestant <k>'s score decreased by <p> points to <s>."
   b. "Contestant <k> is not in the extended heap."
6. **showContestants**
   Print the contents of the extended heap.
   The output should be a sequential list of the contestants currently in the extended heap. For each contestant, print in a single line:
   "Contestant <k> in extended heap location <i> with score <s>."
7. **showHandles**
   Print the contents of the handle array in order, one value per line, formatted as follows:
   "Contestant <k> stored in extended heap location <j>."
   Your program should produce n lines of output. If there is no Contestant <k>, you should print
   "There is no Contestant <k> in the extended heap: handle[<k>] = -1."
8. **showLocation** <k>
   Show the location of the contestant with id <k> in the heap, as follows:
   "Contestant <k> stored in extended heap location <j>."
   If there is no Contestant <k>, you should print:
   "There is no Contestant <k> in the extended heap: handle[<k>] = -1."
9. **crownWinner**
   Remove all contestants from the extended heap, in order, until only one remains. Then print:
   "Contestant <k> wins with score <s>!"

Find sample input and output in Blackboard.

Submission Instructions

You may write the code using C, C++, or Java. **Your program must compile on** remote.cs.binghamton.edu**. No exceptions.** It should be purely a command line program. NO GUI will be accepted.

Please submit a *.tar.gz* file to Blackboard.

The zip file should be named (lower case) as follows:

```
<last name>_<first initial>_p2.tar.gz
```

When the file is unzipped it should contain a directory with the same base name as the zip file.

The directory should contain the following files:

1. File(s) with the source code for the program and possibly a makefile.
2. A read-me file named *readme.txt* which should contain:
   Line 1: C, or C++, or Java
   Line 2+: Either a comma-delimited list of files, which will be compiled and linked with the appropriate compiler (gcc, g++, or javac) and then executed (make sure that for Java the first file has the main method) or the single word "make" which will execute the makefile in the unzipped directory. The makefile will need to produce an executable called "submission" or a "Submission" class in the java case, which will be interpreted/executed. The remainder of the file should describe any assumptions that limit the applicability of the program, any known bugs, and the honesty statement.

Plagiarism Policy

All your code will be checked for similarity to other submissions using Moss. Programmers tend to reproduce the same code that they have seen before. So you are advised not to look at each other's code. Please review the course's plagiarism policy.