Homework

Do not Google for a solution and copy it – if you need to watch a video or look at another solution, then do that...close it...and then program it yourself. It is important to try, to make mistakes, and then debug them.

READ Chapters 4, 7, 8 of "grokking algorithms" – try all exercises in 8, but do NOT turn in! (yes, you already read 4 and 7 – but it is worth it)

WATCH the videos on Quicksort – try to really understand the in-place swapping versions of the algorithm

Homework (30 points)

for bool/true/false you will need `#include <stdbool.h>`
see https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.bpxbd00/stdboolh.htm

#1 Program a simple[1] MAX-heap package – use an array like in slides 101/102 – `d` is a data integer. Assume the heap is a global int array that you do not need to pass to these functions:

- `bool isEmpty()` – returns true if the heap is empty, false otherwise
- `bool isFull()` – returns true if the heap is full, false otherwise
- `void push(d)` – insert value `d` into the heap and re-balance the heap – if the heap is full, print an error msg
- `int pop()` – returns `d` from top of heap (the largest value in the heap) and re-balance the heap – if heap is empty, print an error msg

You will need

- a global index variable pointing to the next open slot of the heap
- 3 functions to find the index for the parent, left child, and right child of a particular element
- a function `isNull(i)` to check if node `i` is NULL (hint: how does it compare with the "next open slot of the heap")
- a recursive `rebalanceUp(i)` function to say that `i` is the location of a new node that needs to be compared with its parent to find its right place in the heap – remember you are building a MAX-heap so the bigger number goes up
- a recursive `reblanceDown(i)` function to say that `i` is the location of a new node that needs to be compared with its children to find its right place in the heap – remember to swap with the LARGER of the two children to maintain MAX-heap

Test your package of functions with your own test cases – do NOT use any HEAP or TREE libraries.

Turn in your program and screenshot(s) of at least two test runs into Canvas

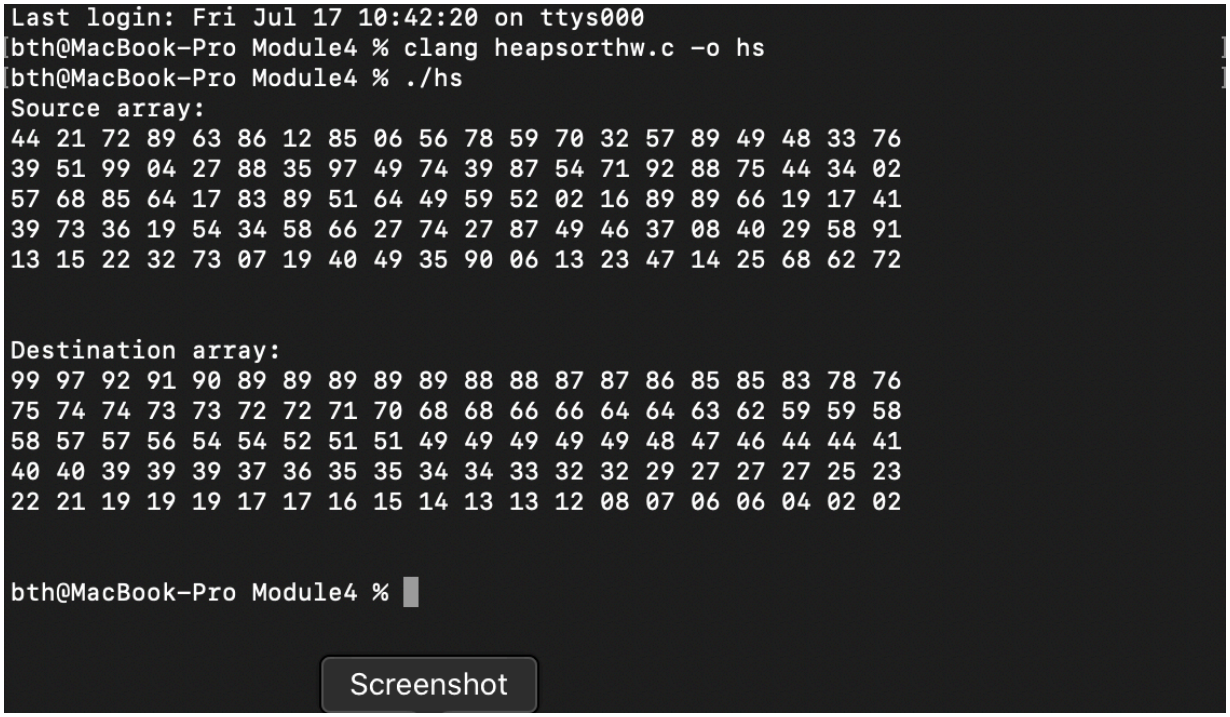
[1] It is "simple" because data=priority, so no changing priorities

Homework (30 points)

#2 Using the package of functions you built in #1 write a heap sort program:

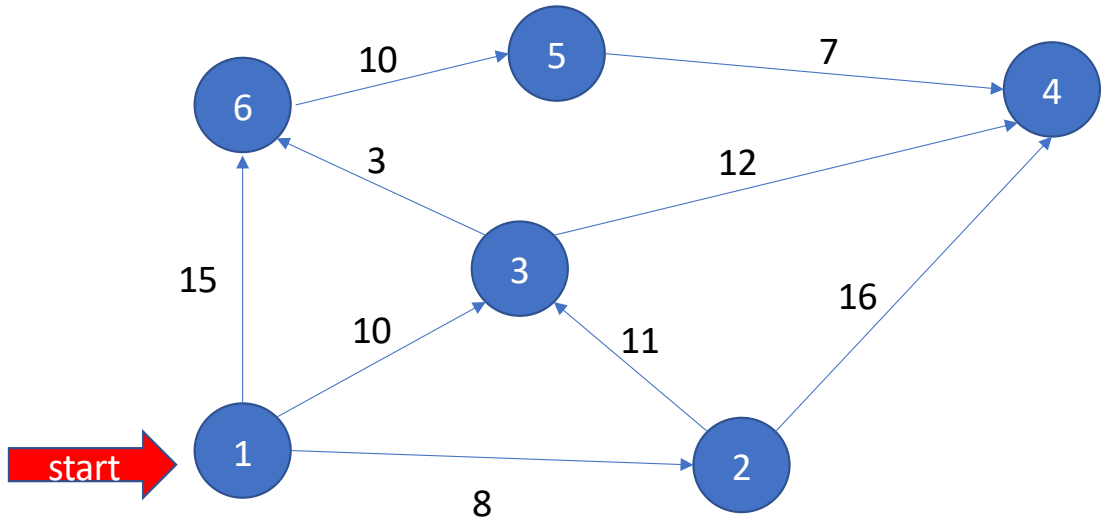
- generate 100 random numbers – print them out
- insert the 100 numbers into your heap
- pull them out again – the list will be in descending order
- print them out as the sorted list

Upload the program and a screen shot of your run to Canvas.


Last login: Fri Jul 17 10:42:20 on ttys000
bth@MacBook-Pro Module4 % clang heapsorthw.c -o hs
bth@MacBook-Pro Module4 % ./hs
Source array:
44 21 72 89 63 86 12 85 06 56 78 59 70 32 57 89 49 48 33 76
39 51 99 04 27 88 35 97 49 74 39 87 54 71 92 88 75 44 34 02
57 68 85 64 17 83 89 51 64 49 59 52 02 16 89 89 66 19 17 41
39 73 36 19 54 34 58 66 27 74 27 87 49 46 37 08 40 29 58 91
13 15 22 32 73 07 19 40 49 35 90 06 13 23 47 14 25 68 62 72
Destination array:
99 97 92 91 90 89 89 89 89 89 88 88 87 87 86 85 85 83 78 76
75 74 74 73 73 72 72 71 70 68 68 66 66 64 64 63 62 59 59 58
58 57 57 56 54 54 52 51 51 49 49 49 49 49 48 47 46 44 44 41
40 40 39 39 39 37 36 35 35 34 34 33 32 32 29 27 27 27 25 23
22 21 19 19 19 17 17 16 15 14 13 13 12 08 07 06 06 04 02 02
bth@MacBook-Pro Module4 %
Screenshot

Homework – Extra Credit (10 points)

#3 Using Dijkstra's algorithm, write down the shortest paths in this graph from the start node to each other nodes



Homework – due 6pm PT next Tuesday

Please upload to Canvas your C program and screen shot of you executing the program.

For this homework assignment, you may (if you choose) work in teams of 2.

Each student is expected to work WITH the other student while writing and testing (via ZOOM or whatever collaboration tool you like). Do not split the task between the two of you.

Each student should upload programs/screen shot + pdf of the Dijkstra algorithm results

The comment at the top of the programs should say if it was done by a team and who the two members of the team were (so that the TAs do not have to grade the same homework twice).