Assignment No. 2: Analysis & Comparison of Bottom-up and Top-down Build Heap Approaches

Allocated time: 2 hours

Implementation

You are required to implement **correctly** and **efficiently** two methods for building a heap, namely the *bottom-up* and the *top-down* strategies.

You may find any necessary information and pseudo-code in your course notes, or in the book[1]:

- *Bottom-up*: section 6.3 (Building a heap)
- *Heapsort*: chapter 6.4 (The heapsort algorithm)
- *Top-down*: section 6.5 (Priority queues) and problem 6-1 (Building a heap using insertion)

Thresholds

Threshold	Requirements
5	Implement and exemplify correctness of bottom-up build heap procedure
6	Implement and exemplify correctness of heapsort
7	Implement and exemplify correctness of top-down build heap procedure
9	Comparative analysis of the two build heap methods in the average case
10	Comparative analysis of the two build heap methods in the worst case

Evaluation

! Before you start to work on the algorithms evaluation code, make sure you have a correct algorithm! You will have to prove your algorithm(s) work on a small-sized input.

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms*

1. You are required to compare the two build heap procedures in the **average** case. Remember that for the **average** case you have to repeat the measurements m times (m=5) and report their average; also for the **average** case, make sure you always use the **same** input sequence for the two methods – to make the comparison fair.

2. This is how the analysis should be performed:
   - vary the dimension of the input array (*n*) between [100…10000], with an increment of maximum 500 (we suggest 100).
   - for each dimension, generate the appropriate input sequence for the method; run the method, counting the operations (assignments and comparisons, may be counted together for this assignment).

   ! Only the assignments and comparisons performed on the input structure and its corresponding auxiliary variables matter.

   Generate a chart which compares the two methods under the total number of operations, in the **average** case. If one of the curves cannot be visualized correctly because the other has a larger growth rate, place that curve on a separate chart as well. Name your chart and the curves on it appropriately.

3. Interpret the chart and write your observations in the header (block comments) section at the beginning of your main .cpp file.

4. Prepare a demo for each algorithm implemented.

5. For this laboratory only the demo for heapsort should be presented. The analysis is not needed.

6. We do not accept assignments without code indentation and with code not organized in functions (for example where the entire code is in the main function).

7. ***The points from the requirements correspond to a correct and complete solution, quality of interpretation from the block comment and the correct answer to the questions from the teacher.***