Weekplan: Priority Queues and Heaps

The 02105+02326 DTU Algorithms Team

Reading

*Introduction to Algorithms*, Cormen, Rivest, Leisersons and Stein (CLRS): Chapter 6 + Appendix B.5

Exercises

1. **Heap Properties and Run by Hand**  
   Solve the following exercises.

1.1 [w] Which of the following trees are heaps?

![Trees](image)

1.2 [w] Which of the following arrays are heaps? Index 0 is not used and is therefore marked with $-$

   \[ A = [-, 9, 7, 8, 3, 4] \quad B = [-, 12, 4, 7, 1, 2, 10] \quad C = [-, 5, 7, 8, 3] \]

1.3 [w] Let \( S = 4, 8, 11, 5, 21, \ast, 2, \ast \) be a sequence of operations where a number corresponds to an insertion of that number and \( \ast \) corresponds to an \text{EXTRACTMAX} operation. Starting with an empty heap \( H \), show how \( H \) looks after each operation in \( S \).

1.4 Is a sorted array a heap?

1.5 Where can the minimum element be found in a (max-)heap?

1.6 [BSc] Show that \text{INSERT}, \text{EXTRACTMAX} and \text{INCREASEKEY} maintains the heap property.

1.7 [\*] CLRS 6.5-9.

2. **Priority Politics**  
The Kakistocratic Party want you to help them implement their "Fresh Air"-policy. Design a registry of all citizens and their income such that one can efficiently find those with the lowest income and deport them. Specifically, the system must support the following operations.

- \text{INSERT}(c, i): insert a person with social number \( c \) and a yearly income \( i \) in the system.
- \text{DEPORTLOWESTINCOME}(): Remove and return the person with the lowest income.

Design an efficient solution for the system.
3 Priority Queue Operations We now want to extend the set of operations on priority queues. We are interested in the following operations.

- **REMOVELARGEST(m)**: remove the m largest elements in the priority queue.
- **DELETE(x)**: remove the element x from the priority queue.
- **FUSION(x, y)**: remove x and y from the priority queue and add the element z with key x.key + y.key.
- **FINDLARGEST(x)**: return the elements in the priority queue with key ≥ x.
- **EXTRACTMIN**: Remove and return the element with the lowest key.

We want to support these operations efficiently, while maintaining the complexities of the standard operations. Let n be the number of elements in the priority queue. Solve the following exercises.

3.1 Extend the priority queue to support REMOVELARGEST(m) in $O(m \log n)$ time.

3.2 Extend the priority queue to support DELETE and FUSION in $O(\log n)$ time.

3.3 [*] Extend the priority queue to support FINDLARGEST in $O(m)$ time, where m is the number of elements with key ≥ x.

3.4 [*] Extend the priority queue to support EXTRACTMIN in $O(\log n)$ time.

4 Satellite Data Let $A[0..n]$ be a heap represented as an array. Each element x in the heap is represented by an index i og the key stored in $A[i]$. It is often useful to store some extra information (called satellite data) associated with an element (for instance if we want to store persons in a heap the satellite data could be age, gender, height, weight, etc). Show how to support access to satellite data in $O(1)$ time only given the index i while still maintaining the running times for the standard heap operations.

5 Heap Properties Let T be a complete binary tree of height h. Solve the following exercises.

5.1 Show the number of nodes in T is $n = 2^{h+1} - 1$. Hint: we know $n = 1 + 2 + 4 + \cdots + 2^h$. Multiply the sum by 2 and subtract the sum.

5.2 [BSc] Show that the sum, $S = n/4 \cdot 1 + n/8 \cdot 2 + n/16 \cdot 3 + n/32 \cdot 4 + \cdots = \Theta(n)$. Hint: Calculate $S - S/2$.

6 Implementation of Heaps We are interested in implementing a priority queue using a heap represented by an array. Solve the following exercises.

6.1 [*] Implement the INSERT and EXTRACTMAX operations.

7 Sums Let $A[0..n-1]$ be an array of integers. We are interested in the following operations on A.

- **CHANGE(i, x)**: set $A[i] = x$.

Solve the following exercises.

7.1 [w] Give a data structure that supports SUM in $O(1)$ time and uses $O(n^2)$ space.

7.2 [*] Give a data structure that supports SUM in $O(1)$ time and uses $O(n)$ space.

7.3 [**] Give a data structure that supports both SUM and CHANGE in $O(\log n)$ time and uses $O(n)$ space.