Study Note

This week we are going to introduce two fundamental data types, address the challenges of developing algorithms and data structures that can serve as the basis of efficient implementations, and try to convince you that such implementations enable solution of a broad range of applications problems that could not be solved without them.

Lecture: Priority Queues. We introduce the priority queue data type and an efficient implementation using the *binary heap* data structure. This implementation also leads to an efficient sorting algorithm known as *heapsort*. We conclude with an applications of priority queues where we simulate the motion of $\mathrm{N}N$ particles subject to the laws of elastic collision.

Lecture: Elementary Symbol Tables. We define an API for *symbol tables* (also known as *associative arrays*) and describe two elementary implementations using a sorted array (binary search) and an unordered list (sequential search). When the keys are Comparable, we define an extended API that includes the additional methods min, max floor, ceiling, rank, and select. To develop an efficient implementation of this API, we study the *binary search tree* data structure and analyze its performance.

Exercise. (*Sorry, we are still waiting for Coursera to migrate the exercises from the old platform*.) Drill exercises on the lecture material.

Programming Assignment: 8-Puzzle. Your programming assignment is to implement the famous A* search algorithm to solve a combinatorial problem, and to substantially speed it up with an efficient priority queue implementation.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested Readings. Section 2.4, 3.1, and 3.2 in *Algorithms, 4th edition*.