Algorithms Example Sheet 3: Core Questions

Binary Search Trees (BSTs)

Exercise 3.C.1 [Core BST operations]

(a) Define *binary search trees.*

(b) Write pseudocode for each of the following operations, argue about their correctness and state their time complexity:

1. `max()`
2. `min()`
3. `find(x)`
4. `insert(x)`
5. (+) `delete(x)`
6. `successor()`
7. `predecessor()`

(c) What is the worst-case height for a BST of n items?

(d) Give an example of a BST that is not a min-heap.

(e) (optional) Attempt **[2012P1Q6]**.

(f) Your friend argues that they have an $\mathcal{O}(n \log\log n)$ time algorithm based on comparisons for creating a BST. Is this possible?

2-3-4 trees / Red-Black Trees

Recommended reading/useful links:

- Chapters 3.2 and 3.3 in Sedgewick's Algorithms
- Chapter 13 from CLRS
- Chapter 3.3 in Okasaki's "Purely functional data structures"
- 2-3-4 trees visualisation.
- RBT visualisation.

Exercise 3.C.2 [2-3-4 Trees]

(a) Define the *2-3-4 trees.*

(b) Explain how search works in a 2-3-4 tree.

(c) Describe how a 2-3-4 tree in insertions.

(d) Draw the 2-3 tree that results when you insert the keys `Y L P M X H C R A E S` in that order into an initially empty tree.

(e) Draw all the structurally different 2-3-4 trees of size 1 to 6.

Exercise 3.C.3 [Red Black Trees]

(a) Define the five properties of *red-black trees.*

(b) Define the mapping between 2-3-4 trees and RBTs.

(c) Argue that this mapping is one-to-one (bijective).

(d) Why did the lecturer chose to present 2-3-4 trees before RBTs?

(e) Show that a RBT with n internal nodes has height at most $2\log_2(n+1)$.

(f) What are the smallest and largest possible number of nodes of a red-black tree of height h, where the height is the length in edges of the longest path from root to leaf?

(g) Let us define a *relaxed red-black tree* as a binary search tree that satisfies all red-black properties except that the root may be either red or black. Consider a relaxed red-black tree T whose root is red. If we color the root of T black but make no other changes to T , is the resulting tree a red-black tree? **[CLRS 13.1.3]**

(h) Describe a red-black tree on n keys that realizes the largest possible ratio of red internal nodes to black internal nodes. What is this ratio? What tree has the smallest possible ratio, and what is the ratio? **[CLRS 13.1.7]**

(i) (optional) Attempt **[2015P1Q7]**.

1 B-trees

Exercise 3.C.4 [B-trees]

(a) Define *B trees* and *B+ trees.*

(b) Describe how insertion and deletion work.

(c) Using a soft pencil, a large piece of paper and an eraser, draw a B-tree with $t = 2$, initially empty, and insert into it the following values in order: $63; 16; 51; 77; 61; 43; 57; 12; 44; 72; 45; 34; 20; 7; 93; 29$.

i. How many times did you insert into a node that still had room?

ii. How many node splits did you perform? What is the depth of the final tree?

iii. What is the ratio of free space to total space in the final tree?

Hash tables

Exercise 3.C.5 [Collisions]

(a) What is the *collision problem*?

(b) Describe two methods of solving the collision problem.

(c) Make a hash table with 8 slots and insert into it the following values: $15; 23; 12; 20; 19; 8; 7; 17; 10; 11$. Use the hash function $h(k) = (k \bmod 10) \bmod 8$ and, of course, resolve collisions by chaining.

(d) Non-trivial Imagine redoing the exercise above but resolving collisions by open addressing. When you go back to the table to retrieve a certain element, if you land on a non-empty location, how can you tell whether you arrived at the location for the desired key or on one occupied by the overspill from another one? (**Hint:** *describe precisely the low level structure of each entry in the table.*)

(e) How can you handle deletions from an open addressing table? What are the problems of the obvious naïve approach?

Exercise 3.C.6 [Multipicative hash functions] Given a table with m entries, what is the desired property of a and b, for the hash function $h(x) = ax + b \pmod m$?

Exercise 3.C.7 [Length of longest chain] (optional) Consider an array of length m (for $m = 10, 10^2, 10^3, 10^4, 10^6$) and choose m values chosen uniformly at random in $0, \dots, m-1$. Count the number of items for the index that receives most entries.

Exercise 3.C.8 [Linear Probing]

(a) Describe *linear probing*.

(b) What is *primary clustering*?

Exercise 3.C.9 [Quadratic Probing]

(a) Describe *quadratic probing*.

(b) Why is the choice of the constants important?

(c) What is *secondary clustering*?

Exercise 3.C.10 [Double hashing] Describe *double hashing*.

Exercise 3.C.11 [Deletions] How can you handle deletions in the aforementioned hashing approaches? How do these compare?

Priority queues

Exercise 3.C.12 [Priority queues]

(a) What are the operations of a binary heap?

(b) State with justification the time complexity for the operations of a priority queue implemented using an array.

(c) State with justification the time complexity for the operations of a priority queue implemented using the min-heap.

(d) Your friend claims that they have implemented a priority queue with 'extractMin()' taking $\mathcal{O}(\log\log n)$. Is this possible?

Binary Heaps

Recommended reading/useful links:

- This visualisation website
- CLRS **2nd edition** Chapter 19 (see here)
- These slides
- The paper that introduced this data structure.

Exercise 3.C.13 [Binomial heaps]

(a) Define the *binomial tree of degree k*.

(b) Draw the binomial trees of order 2, 3 and 4.

(c) How many children does the binomial tree of degree k contain? Prove it.

(d) How many subtrees does the root binomial tree of degree k have? Prove it.

(e) What is the height of binomial tree of degree k? Prove it.

(f) How many nodes are there in the i-th depth of a binomial tree of degree k? Prove it.

(g) Prove that the following definition is equivalent: the binomial tree of degree 0 is a single node. The binomial tree of degree k is a tree with k children $t_0, t_1, \ldots, t_{k-1}$, where t_i is a binomial tree of degree i.

(h) Define the *binomial heap*.

(i) Prove that if a binomial heap has n nodes, then it contains $\mathcal{O}(\log n)$ binomial trees and the largest of these has degree $\mathcal{O}(\log n)$. Determine exactly the orders of the binomial trees.

Exercise 3.C.14 [Binomial heaps] Describe the implementation of a priority queue using a binomial heap, stating the time complexity for each operation. What is the connection between merging binomial heaps and adding two integers?