Technische Universität München
Fakultät für Informatik
Lehrstuhl für Algorithmen und Komplexität
Prof. Dr. Harald Räcke
Richard Stotz

Winter term 2018/19
Assignment 8
December 3, 2018

Efficient Algorithms and Data Structures I

*Deadline: December 10, 2018, 10:15 am in the **Efficient Algorithms** mailbox.*

Homework 1 (5 Points)

Insert the keys 4, 55, 37, 34, 7, 16, 23, 85, 61 into a hash table of length $m = 11$ using open addressing with the primary hash function $h_1(k) = k \bmod m$.
Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_2(k) = 1 + (k \bmod (m-1))$.

Homework 2 (4 Points)

(a) Suppose that we use hashing with chaining and keep the lists in sorted order. How does this modification affect the expected running time of successful searches, unsuccessful searches and insertions?

(b) Suppose that we use uniform hashing *with chaining*, i.e., the hashing function is chosen at uniformly at random from the set of all hash functions. Determine the expected number of empty table entries in a table of size m after n different inserts.

Homework 3 (5 Points)

We are given a huge array T that initially contains garbage data. Array T should be used as a dictionary of pointers, but initializing all its entries to 0 is infeasible due to its size.
Describe a scheme for implementing a direct-address dictionary on T.
Each stored object should use $\mathcal{O}(1)$ space; the operations SEARCH, INSERT, and DELETE should take $\mathcal{O}(1)$ time each; and initializing the data structure should take $\mathcal{O}(1)$ time.
Example: When accessing $T[i]$, your scheme must detect whether the content of $T[i]$ is garbage or actual data. It $T[i]$ is valid, your scheme must provide a way to access – in constant time – the data associated with $T[i]$.
Hint: You may use a constant number of additional arrays. These arrays can be treated somewhat like a stack whose size is the number of keys actually stored in the dictionary, to help determine whether a given entry in the huge array is valid or not.

Homework 4 (6 Points)

Suppose that we have a hash table with n slots using hashing with chaining. We insert n keys into the table, choosing the hash position uniformly at random for each key (uniform hashing).

We want to show that the slot with the maximum length of any chain is $\mathcal{O}(\frac{\log n}{\log \log n})$ with high probability.

(a) Let X be the maximum length of a list and let X_i denote the length of the ith list. How is the random variable X_i distributed? Show that $\Pr[X < t] \geq 1 - 1/n$ is implied by

$$\Pr[X_1 \geq t] \leq \frac{1}{n^2} \ .$$

(b) Show that there is a constant $c > 0$ such that for n sufficiently large, we have

$$\Pr\left[X_1 \geq c \cdot \frac{\log n}{\log \log n}\right] \leq \frac{1}{n^2}$$

Hint: There are several ways to prove this. Chernoff-Bounds or the bounds on Slide 214 are helpful. Use some constant base for the logarithm that suits your proof, e.g. base 2 or base e.

Tutorial Exercise 1

Suppose we have stored n keys in a hash table of size $m > 0$, with collisions resolved by chaining. Let $\alpha = n/m$. Further assume that we know the length of each chain, including the length L of the longest chain.
Describe a randomized procedure that returns any key in the hash table with the same probability. The expected running time of your algorithm should be $\mathcal{O}(L(1 + 1/\alpha))$.

Additional Information: The head of any chain can be accessed in constant time and it stores the length of the chain. You have access to a random generator that returns uniformly at random an integer between 1 and some integer c that you may specify.

Tutorial Exercise 2

Let $U = \{0, \ldots, p-1\}$ for a prime $p > n \geq 2$. For $x \in \mathbb{Z}_p$, define the hash function $h_{a,b}(x)$ as

$$h_{a,b}(x) = (ax + b \bmod p) \bmod n$$

Consider the class of hash functions

$$\mathcal{H} = \{h_{a,b} | a, b \in \mathbb{Z}_p\}$$

(a) Show that $\mathcal{H}$ is not universal for all $n \geq 2$.

(b) Show that $\mathcal{H}$ is $(1.1, 2)$ independent for $n \geq 2$ and $p \geq 100n^2$.

(c) Why would you not choose $\mathcal{H}$ as a class of hash functions?

> Somehow the verb 'to hash' magically became standard terminology for key transformation during the mid-1960s, yet nobody was rash enough to use such an undignified word publicly until 1967.
>
> - D. E. Knuth