Weekplan: Hashing and Dynamic Sets

Philip Bille   Inge Li Gørtz

Reading

Introduction to Algorithms, Cormen, Rivest, Leisersons and Stein (CLRS): Chapter 11 excluding 11.5.

Exercises

1 Run by Hand and Properties

1.1 $[w]$ Insert the key sequence $K = 7, 18, 2, 3, 14, 25, 1, 11, 12, 1332$ into a hash table of size 11 using chained hashing with hash function $h(k) = k \mod 11$.

1.2 $[w]$ Insert the key sequence $K = 2, 32, 43, 16, 77, 51, 1, 17, 42, 111$ into a hash table of size 17 using linear probing with hash function $h(k) = k \mod 17$.

1.3 Delete 111 and 51 from the hash table produced in exercise 1.2.

1.4 Assume we do deletion in linear probing *without* reinserting the elements in the chunk to the right of the deleted element. Give a shortest possible sequence of dictionary operations that show this does not work correctly.

1.5 Let K be a sequence of keys stored in a hash table A using chained hashing. Given A, can one efficiently find the maximum element in K?

2 Divisors in the Division Method Consider the hash function $h(k) = k \mod 10$ and the key sequence $K = 0, 5, 20, 40, 65, 15, 90, 95, 80, 55$.

2.1 Why is the choice of hash function problematic in relation to K?

2.2 Explain why we use prime numbers in the division method.

3 Lazy Deletion in Linear Probing Consider the following ”lazy” strategy for deletion in linear probing. When an element is deleted on position p we mark that the element on position p has been deleted.

3.1 Explain how SEARCH and INSERT should be modified to work when using this strategy.

3.2 Explain benefits and drawbacks using this method compared to ”eager” deletion.

4 Game Server Statistics For your new extremely successful online game you would like to keep track of whether the active users come from a small group of very active players, or a large group of different players who only play infrequently. Each player has a unique ID and from your game server you can access the sequence of player IDs from all game sessions.

4.1 Give an algorithm that counts the number of *unique* players on the game server.

4.2 Give an algorithm that finds the player who has played the most games.

5 Bit Vectors Computers are often referred to as *w-bit computers*. For instance, most modern computers are 64-bit computers. This means that registers and memory cells stores w-bits each and the primitive data types, such as integers, floating point numbers, and pointers, are represented in w-bits. Standard programming languages support bit manipulation operations w-bits in constant time (see the manual for your preferred programming language), including shifting and bitwise logical operators. We want to use these to efficiently implement arrays of bits, called *bit vectors*. Suppose you are working on a w-bit computer. Solve the following exercises.

- **5.1** Show how to compactly represent a bit vector B of length w, such that the i'th bit can be accessed or flipped in $O(1)$ time.
- **5.2** Show how to compactly represent a bit vector B of length n (for large $n \gg w$) such that the i'th bit can be accessed or flipped in $O(1)$ time.
- **5.3** Show how a bit vector can be used to represent a dynamic set without satellite data using direct addressing.

6 [$*$] **Sorting in Small Universes** Let $A[0..n-1]$ be an array of integers from $\{0,\ldots,n-1\}$. Give an algorithm that sorts A in $O(n)$ time. *Hint:* start by inserting the numbers into a chained hash table with the identity function as hash function.

7 [$**$] **Uninitialized Arrays** We want to implement a *huge* array A such that we can efficiently access and change an entry in A. In the beginning the entries of A might contain "garbage" and because of the size we do not want to spend time on initializing all the entries. Give a solution that uses linear space in the size of the array, allows access and updates in $O(1)$ time per entry, and only uses $O(1)$ time for initialization.