Technische Universität
Kaiserslautern

Department of Computer Science
Algorithms & Complexity Group

Issue Date: 09.06.2014
Version: 2014-06-10 10:35

Exercise Sheet 8 for Algorithm Engineering, SS 14

Hand In: *Until Monday,* ***23.06.2014****, 10:00 am, email to* `wild@cs...` *or in lecture.*

Problem 18

2 + 3 points

We consider hashing where the probability that our hash function returns an *even* address is c times the probability for an odd address. Conditional on returning an even address, all even addresses are chosen with equal probability, and the same holds true for the odd addresses.

You may assume that the size of the hash table is even.

a) Under this model, determine the expected costs for an unsuccessful search in a hash table of size m that stores n elements and uses *indirect chaining*.

b) Determine the expected costs for an unsuccessful search in a hash table of size m that stores n elements and uses *direct chaining with relocation*.

Please turn!

Problem 19

1 + 2 + 2 points

In this exercise, we consider some details of the implementation of extendible hashing.

For all your implementations, analyze the number of memory accesses and I/O operations depending on the current "depth" ℓ of the hash function (which means that hash table H has 2^ℓ cells) and the block size B. Here, we assume that H is already too big to be held in main memory, so H and all leaves are initially only stored in external memory.

a) Give an efficient pseudocode implementation of **find**(H, x), which is given a key x and returns the data $D(x)$ associated with key x if x is present in hash table H and `null` otherwise.

b) Give an efficient pseudocode implementation of **grow-ht**(H), the auxiliary function that doubles the size of the current hash table upon overflow of a leaf, when the used hash function is $h_\ell : [0..2^n - 1] \to [0..2^\ell - 1]$ with

$$h_\ell(x) \;=\; x \operatorname{div} 2^{n-\ell}\,,$$

i. e., we take the ℓ most significant bits of x.

You do *not* need to implement the splitting of the leaf itself, only the code for creating the larger hash table including valid links to the (old) leaves is required.

c) Implement **grow-ht**(H) for the alternative hash function $\tilde{h}_\ell : [0..2^n{-}1] \to [0..2^\ell{-}1]$ with

$$\tilde{h}_\ell(x) \;=\; x \bmod 2^\ell\,.$$

Assuming that we can actually enlarge the array H (instead of allocating a brand-new array with double size), which hash function allows the more efficient implementation of grow-ht(H)?

Problem 20

4 points

Prove *Stirlings formula* $n! \sim \sqrt{2\pi n} \cdot n^n/e^n$ using the saddle-point method.

Hint: Consider the coefficent integral

$$\frac{1}{2\pi i} \oint\limits_{|z|=r} \frac{e^z}{z^{n+1}}\, dz$$

with the choice $r = n$. Computations will be easier, if you represent the argument of the integral in polar coordinates.

You are expected to explain, why the choice $r = n$ is reasonable and why the function is "well-behaved".