Laboration 2

Due 191209 (Part I) and 191216 (Part II).

Part I

Hashing with linear probing, $f_1(x,i) = (h(x)+i) \bmod m$ $(i = 0, 1, \cdots, m-1)$, suffers from primary clustering; which makes the probe sequence longer. The performance of linear probing degrades on average as the table gets loaded. The purpose of this assignment is to empirically study the performance of a hash table with some probing strategies which are variants of linear probing.

Recall that in linear probing, to insert a key x, slots are tried in sequential order, starting from the home address (i.e., $h(x)$), until an empty slot is founded. Consider the following variant of linear probing:

On inserting a key x, instead of just checking locations $h(x), h(x)+1, h(x)+2, \cdots$, one can also try to store x at the first empty slots of $h(x), h(x)-1, h(x)-2, \cdots$. For this purpose, maintain two accounts for each location ℓ in the hash table:

- ℓ_{down}: The number of keys x in the table that have the home address $h(x) = \ell$ and been inserted into the table using $f_1(x,i) = (h(x)+i) \bmod m$ $(i = 0, 1, \cdots, m-1)$.
- ℓ_{up}: The number of keys x in the table that have the home address $h(x) = \ell$ and been inserted into the table using $f_2(x,i) = (h(x)-i) \bmod m$ $(i = 0, 1, \cdots, m-1)$.

When inserting x and $h(x) = \ell$ is occupied, run $f_1(x,i)$ if $\ell_{down} \leq \ell_{up}$; otherwise, run $f_2(x,i)$.

Design and Implementation

Your task is to implement linear probing and the above variant, and evaluate the performance of insert operations. You may assume that all the keys are distinct. One may take into account the following measures: running time, the load factor, the hashing functions used, the number of probes, the number of insertions that encountered a collision, the length of the longest collision chain, and so on.

Part II

Given n $(n \geq 4)$ distinct elements, design a divide and conquer algorithm to compute the 4^{th} smallest element. Your algorithm should run in linear time in the worst case. Analyze your algorithm (namely, show that your algorithm is correct and calculate the *exact* number of comparisons used by the algorithm). You may assume that $n = 2^k$ for some positive integer k. No implementation and test of the algorithm is needed. *Hint: One can use the induction technique to show the correctness. Check Chapter 4 for more examples of performance analyses.*

Each group should turn in a report describing and illustrating the theoretical and experimental results. The report can be written in either Swedish or English and should not be handwritten. The deadline for the submission of the report: Part I (191209) and Part II (191216).

Before submitting your report, you should discuss your solution to the laboration (design, implementation, test, and report) with your lab-assistants.