Jakob Wulf-Eck

Lab 6 Writeup

1. I set up my experimental profiling procedure using a series of float arrays to hold the values that will ultimately be displayed when profiling is complete. The whole profiler loops five times to repeat the experiment and build an average. Within this loop, it loops again to make use of five different input sizes and five different seeds. Each repetition/seed combination times every build and adds it to the array position for the appropriate size. They also time and add every find operation and count the number of successful and unsuccessful finds for each size. The build times are divided by 25 and the successful/unsuccessful find times are divided by the number of successful/unsuccessful finds to produce the average. All times are then converted from CPU cycles to milliseconds.
2. To generate the random data, C++’s srand function is set with one of five predefined seeds to produce 5 different series of random numbers. An array of the desired input size is created and rand() is used to generate the random numbers. A modulo of 5 times the hashtable size is used to limit the random numbers to the range from 1 to 5m. The same procedure is used to generate numbers to be searched for.
3. The insert and find operations were timed using C++’s clock function. Clock was called immediately before the call to insert/find an element and its value was stored. Clock was then called again after each insertion/find. The second clock value minus the first one was the value added to the result.
4. The sums of all insertion times for each table and input size were divided by 25, the total number of times the table was built with each input size. The find and unsuccessful find times were divided by the total number of successful/unsuccessful finds for that size. All times were then divided by CLOCKS\_PER\_SEC/1000 to scale them from CPU clocks to milliseconds.
5. The table with quadratic probing generally built the fastest. The double hashing table was slightly slower, tending to take about 2-3% longer. Open hashing was approximately 15% slower than quadratic probing. Times for all three tables corresponded roughly linearly with input size – the time taken to insert 500,000 elements was approximately five times the time taken to insert 100,000 in all cases. Not-found times were higher than found times for all tables except open hashing. Find times did not appear to vary directly with size.
6. The experimental results support the constant-time nature of hash table insert and find operations. The process of building the hash table requires n\*O(1) operations, and should thus be an O(n) operation. The time needed to build the hash table in this experiment grew linearly with input size and appears to be O(n). Find times did not appear to be strongly correlated with input size. As the find operations were timed and averaged individually, this supports the idea that hash-table find operations run in constant time.