CS 400/600 Programming Assignment #4 Hash Table

Shumin Guo

Due Date: Nov. 11th, 2010

Report

Along with your working source code (.cpp and .h files), you should submit a brief (< 5 pages) report describing how the closed hash function performed as a function of the load factor of the hash table, alpha. You should perform a variety of searches, and compare the number of nodes searched for each index. You should address questions such as:

- How does the hash perform in terms of search efficiency when the hash table is (1) nearly empty (the alpha is very small, like less than 0.1), (2) moderately full (alpha is around 0.5), (3) very full (alpha is larger than 0.8).

Double Probing		Random Probing	
α	Probes	α	Probes
0.015274	0.017	0.015274	0.017
0.0458221	0.049	0.0458221	0.052
0.0763702	0.065	0.0763702	0.08
0.106918	0.116	0.106918	0.101
0.137466	0.148	0.137466	0.164
0.168015	0.205	0.168015	0.173
0.198563	0.278	0.198563	0.247
0.229111	0.338	0.229111	0.294
0.259659	0.367	0.259659	0.356
0.290207	0.451	0.290207	0.394
0.320755	0.461	0.320755	0.483
0.351303	0.636	0.351303	0.54
0.381851	0.677	0.381851	0.601
0.412399	0.808	0.412399	0.721
0.442947	0.907	0.442947	0.79
0.473495	1.017	0.473495	0.88
0.504044	1.154	0.504044	0.995
0.534592	1.185	0.534592	1.075
0.56514	1.365	0.56514	1.313
0.595688	1.735	0.595688	1.413
0.626236	1.843	0.626236	1.745
0.656784	2.184	0.656784	1.983
0.687332	2.423	0.687332	2.287
0.71788	2.8	0.71788	2.457
0.748428	3.415	0.748428	3.033
0.778976	3.959	0.778976	3.667
0.809525	4.665	0.809525	4.193
0.840073	5.275	0.840073	5.227
0.870621	6.792	0.870621	6.798
0.901169	9.986	0.901169	9.422
0.931717	14.752	0.931717	14.45
0.962265	28.271	0.962265	25.891

Table 1: Load Factor α VS. Probes.

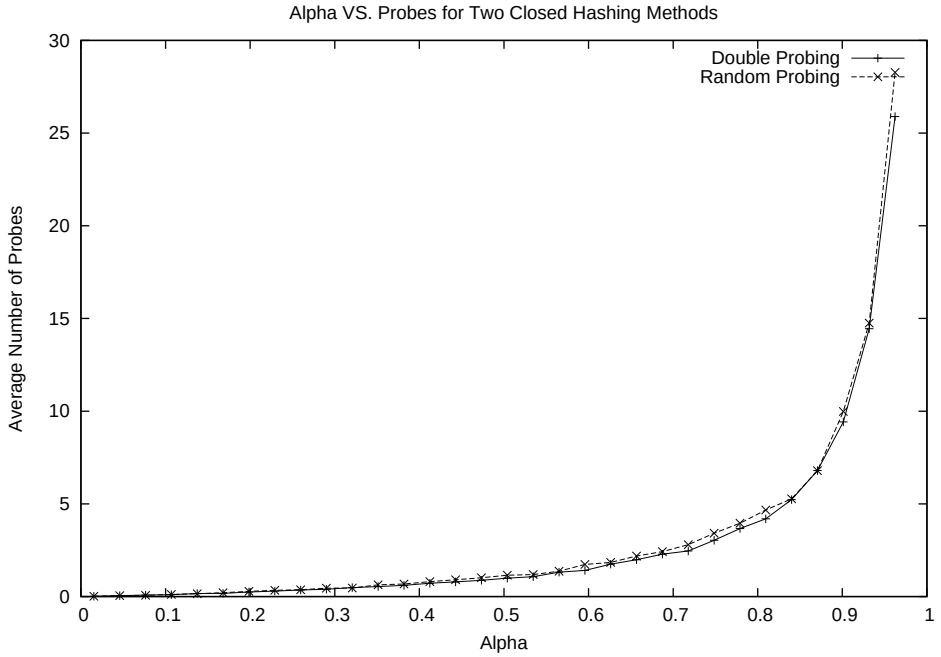

Figure 1: Average Search Efficiency for Double and Pseudo-random Probing Hashing.

From Table 1 and Figure 1 we can see that double hashing and random hashing has identical search efficiency. When the Hash Table is nearly empty or when α is very small close to zero, The average probes for search and insert is minimal, it is far blow one on average. When the Hash Table is relatively full, that is when α is close to 0.5, the average number of probes for insert and search is relatively increased as compared with the previous case, but the number is still very low on average. When the Hash Table is becoming full, the number of probes start to increase far when α is above 0.8, and the worst case began to happen when α is above 0.9. The average number of collisions skyrocketed to over 20 and even more than 25 for random probing. In this case, the Hash Table is heavily overloaded.

- Give some intuitive comments about the performance of your hash table regarding the load factor, α.

The data for the above question is collected by doing several experiments for different load factor α. So, it can refect the time it takes to insert an element or find an element in the Hash Table. I experienced some time variations with different load factor. When the factor is very low, it is very fast to search/insert element, it is almost constant time operation, which clearly agrees with the result in the table and the figure. As the load factor increases with more and more elements inserted into the hash tableit takes a little longer time to do the same operations. And when load factor becomes very close to one. It takes much longer time than the previous scenarios, which mean more collisions are happening. This intuitive experience together with the concrete experiment data teaches us that the load factor should be kept at a relatively low level for it to work efficiently(constant time operation). Usually the load factor should be around 0.5 when 1 probes happen on average. When the load factor is too close to one, the hash table will lose its competativeness as compared with other data structures.

Your report should be supported with data in the form of tables, charts, graphs, etc.
Make sure to base your report on average results over many searches, and not just one search.