MARMARA UNIVERSITY

CSE4077

ADVANCED DATA STRUCTURES

PROJECT 2

Comparison of Collision Resolution Methods Hashing

Eren Başpınar – 150119031

Yiğit Mesut Ak – 150120006

PART 1: EXPERIMENT

1. Experiment Setup

We created 10 different datasets including 600 thousand numbers in different ranges. These datasets are used to measure average insertion times of each hashing method. We measured the time during the insertion of whole dataset and divide the over all result by the insertion number which is 600 thousand. Therefore, we got an average insertion time for a key value for each iteration.

a. Experiment Setup for Quadratic Probing

To select the best coefficients of quadratic probing, we performed measurements for selected values of coefficients. For each coefficient we selected 21 different values and run our program 21*21 times for each dataset. We started our iteration by selecting 1 for the smallest coefficient and set the upper limit as the greatest prime number less than the table size. In our case, since the table size is one million, we set the upper limit as 999'983.

b. Experiment Setup for Double Hashing

As it is taught in the lectures, we wanted to test if the greatest prime number less than the table size is really the best R value for double hashing. In our hypothesis, the best value for R coefficient must be 999'983. To test this claim, we run the experiment for 500 prime numbers.

c. Experiment Setup for Cuckoo Hashing

To test the most efficient cuckoo hashing method, 3 sets of cuckoo hashing is compared. M indicates the table size and P indicates the greatest prime number less than the table size. k is the key value.

$$h_1(k) = k\ (mod\ M)$$

$$h_2(k) = k \times P\ (mod\ M)$$

$$h_3(k) = k + P\ (mod\ M)$$

Among these hash functions, (h_1, h_1), (h_1, h_2), and (h_1, h_3) are used as sets. Experiment ran for 10 datasets. In our case, the table size was one million and P was 999'983.

2. Experiment Results

a. Quadratic Probing Results and Interpretation

By using our program's results, we prepared a heat map to decide on the most appropriate value for coefficients. On table below, the mean value of the total search time for each dataset is given according to different coefficients. The given values represent the average insertion time of a key value in nanoseconds.

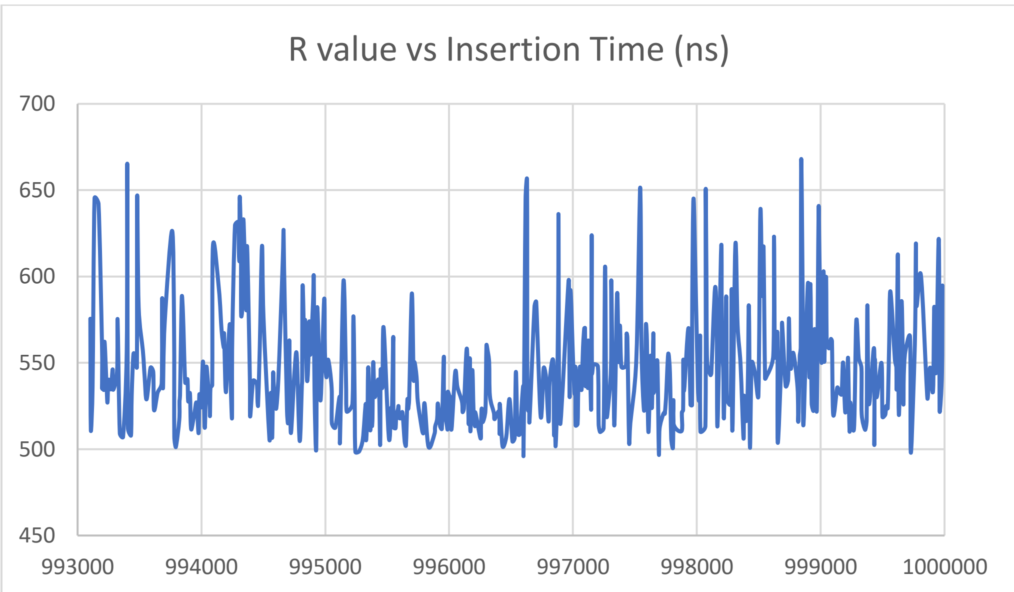
Table-1: Average insertion time of a key value according to different coefficients.

c2 \ c1	1	50000	99999	149998	199997	249996	299995	349994	399993	449992	499991	549990	599989	649988	699987	749986	799985	849984	899983	949982	999981
1	1016.36	975.048	787.089	852.586	1017.53	787.987	783.009	798.783	875.228	795.068	753.788	807.054	756.397	778.015	804.097	815.694	739.85	778.531	740.059	809.738	763.437
50000	936.007	1045.59	825.232	866.984	950.486	793.684	803.239	858.429	840.259	740.462	943.222	798.917	799.565	763.371	755.95	783.281	819.903	805.211	852.363	807.586	792.271
99999	971.613	963.143	799.221	847.757	967.446	730.901	791.266	803.896	819.958	747.602	865.332	749.524	784.518	794.438	814.018	770.938	755.783	770.314	806.862	766.742	793.996
149998	1045.76	820.183	779.568	839.795	953.586	777.764	772.228	791.094	785.157	768.118	798.048	803.675	796.516	741.577	767.669	837.119	797.36	778.702	758.815	788.375	771.331
199997	892.824	806.863	780.886	890.237	1046.67	813.295	743.129	801.82	731.986	782.35	818.103	761.951	834.299	793.791	780.937	808.654	732.737	791.571	742.26	762.587	843.4
249996	943.727	873.921	744.449	865.152	985.636	799.937	774.115	795.243	803.853	791.48	775.952	783.631	765.463	758.902	790.223	764.461	794.706	808.825	781.447	769.216	811.091
299995	978.444	837.698	777.907	876.768	934.268	832.259	783.946	749.795	783.289	757.958	772.805	799.08	760.873	772.646	804.318	776.239	778.82	790.977	814.968	816.839	834.62
349994	751.746	800.283	788.469	828.726	960.912	840.161	765.9	764.454	785.209	777.068	772.715	786.452	779.585	770.751	813.004	766.293	751.119	766.945	779.796	864.393	791.375
399993	756.874	927.455	816.8	895.645	962.731	792.698	750.683	724.938	772.43	778.75	787.403	802.07	808.534	786.405	829.523	834.237	765.612	845.536	772.008	793.35	771.007
449992	798.403	1004.52	859.862	818.129	966.26	757.783	791.303	788.866	771.517	837.609	818.724	808.089	751.146	812.839	782.535	819.65	775.667	844.054	774.416	783.708	730.348
499991	802.952	1001.96	847.609	866.121	963.272	777.242	788.585	772.852	811.809	862.814	762.195	767.959	795.706	799.61	770.593	761.747	791.19	748.801	759.155	792.137	739.235
549990	829.789	1017.57	864.966	892.253	961.307	777.23	785.437	816.608	756.689	791.929	799.421	779.079	802.679	789.796	816.326	779.308	801.453	745.867	843.972	825.342	788.092
599989	809.796	981.972	841.437	840.98	950.208	807.559	773.979	803.985	863.366	784.064	786.298	783.351	810.121	779.602	812.439	753.431	808.242	742.49	762.948	863.23	792.931
649988	770.603	1040.44	846.97	856.306	941.584	780.046	758.184	812.424	781.626	773.836	821.183	764.495	786.54	786.526	739.462	772.404	793.447	782.035	763.098	822.429	787.277
699987	797.18	911.363	848.174	808.964	893.875	766.598	747.629	757.319	766.719	763.58	808.702	778.107	768.628	742.22	814.359	750.179	799.556	772.925	787.828	783.551	775.442
749986	823.356	1033.34	871.213	885.929	964.491	793.25	781.509	769.41	823.531	763.768	845.762	780.126	814.702	778.995	828.982	803.362	819.225	800.825	834.921	798.779	826.114
799985	794.311	810.238	885.697	945.515	966.095	800.733	768.056	786.963	795.069	822.86	795.364	808.586	789.422	806.561	824.564	791.122	792.687	868.287	789.553	762.583	755.413
849984	818.618	762.652	812.261	974.11	939.116	805.365	814.377	772.359	770.404	811.673	818.027	792.549	768.046	811.759	788.295	771.599	747.107	802.37	779.164	768.058	759.069
899983	959.32	861.383	868.044	977.036	990.297	780.077	816.441	799.086	835.494	756.108	748.029	779.216	770.439	803.435	790.757	822.044	836.37	795.626	810.455	767.497	761.478
949982	933.54	819.335	860.204	961.281	952.439	794.965	794.855	781.8	780.374	820.735	761.391	836.516	784.888	763.625	810.673	800.37	790.053	772.474	802.955	803.667	802.321
999981	960.038	837.993	820.678	995.221	844.027	798.355	805.297	902.473	784.237	777.8	755.123	771.921	804.244	791.373	767.903	789.895	774.906	760.723	787.49	750.212	759.942

The empirical approach on this case has shown us that the best coefficients for quadratic hashing are 349'994 for c_1 and 399'993 for c_2.

b. Double Hashing Results and Interpretation

By using the results that we got from our program, we made a graph to see the effects of R value on insertion time.



By the graph, we could not get a meaningful idea on the ideal R value but according to our empirical analysis, the best R value for double hashing was 999'727.

c. Cuckoo Hashing Results and Interpretation

The comparison of different sets of cuckoo hashes are shown below. The average insertion times for a key value for each dataset and the average values are represented. Unsuccessful insertions represent the times when the cuckoo hashing stuck on a cycle.

$h_1(k)$ and $h_1(k)$:

dataset 0	dataset 1	dataset 2	dataset 3	dataset 4	dataset 5	dataset 6	dataset 7	dataset 8	dataset 9	Avg.
1234.75	1044.45	858.683	966.437	1049.3	1095.04	891.176	956.747	933.833	986.187	1001.7

unsuccessful insert per iteration: 176777

$h_1(k)$ and $h_2(k)$:

dataset 0	dataset 1	dataset 2	dataset 3	dataset 4	dataset 5	dataset 6	dataset 7	dataset 8	dataset 9	Avg.
873.126	927.764	952.999	920.189	980.31	1032.34	938.04	1190.06	1027.97	1018.55	986.14

unsuccessful insert per iteration: 52490

$h_1(k)$ and $h_3(k)$:

dataset 0	dataset 1	dataset 2	dataset 3	dataset 4	dataset 5	dataset 6	dataset 7	dataset 8	dataset 9	Avg.
828.702	919.429	853.743	951.566	893.961	1026.52	1018.88	992.016	911.539	1002.49	939.88

unsuccessful insert per iteration: 176777

According to our measurement, the most efficient cuckoo hashing uses $h_1(k) = k\ (mod\ M)$ and $h_2(k) = k \times P\ (mod\ M)$ with the smaller number of unsuccessful insertions.

PART 2: BEST HASH FUNCTION

As a result of the tests, we carried out in the first phase of the project, we found that the best c1 and c2 values for quadratic probing were 349994 and 399993. We determined the ideal R value for double hashing to be 999727. And finally, we found the best hash function number for Cuckoo hashing to be 2.

In the second stage of the project, we used the dataset given to us for the values we found and thus compared the working speeds of the algorithms. In the table below (see Figure 1), you see the insertion time values obtained as a result of running the same dataset with the same data 10 different times. As can be seen from the table, although Cuckoo hashing is generally the fastest running among the functions, double hashing runs almost at the same speed and even surpasses cuckoo hashing in some cases. Linear probing was observed to be by far the slowest of them all. In addition, in our tests, the number of unsuccessful inserts for Cuckoo Hashing was observed to be 0 (See Figure 2)

	Linear Probing	**Quadratic Probing**	**Double Hashing**	**Cuckoo Hashing**
Run 1	745.0647	533.1345	465.793	456.4318
Run 2	730.126	559.9565	445.9753	443.1841
Run 3	739.4655	568.4273	442.8132	475.62
Run 4	718.1868	549.6157	437.6627	460.8852
Run 5	737.1515	565.7008	491.7525	453.3597
Run 6	758.283	550.7783	474.8957	459.7517
Run 7	704.7872	571.422	436.0687	466.3477
Run 8	702.2602	558.0375	433.7481	448.935
Run 9	710.705	593.1373	438.5538	461.149
Run 10	710.155	572.3096	466.8978	444.63

Figure 1: Test results for determined optimum values

```
Linear Probing Result: 699.8655
Quadratic Probing Result for c1=349994 and c2=399993: 558.4748333333333
Double Hashing Result for R=999727:     451.5423333333333
Cuckoo Hashing Result for hash functions 1 and 2:   476.80583333333334
unsuccessful insert per iteration:  0
```

Figure 2: Example output of the code for part 2