

**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.

* 1. **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.

* 1. **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.

* 1. **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.

Among these hash functions, (*h1*, *h1*), (*h1*, *h2*), and (*h1*, *h3*) are used as sets. Experiment ran for 10 datasets. In our case, the table size was one million and *P* was 999'983.

1. **Experiment Results**
   1. **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.*

A screen shot of a computer

Description automatically generated

The empirical approach on this case has shown us that the best coefficients for quadratic hashing are 349'994 for c1 and 399'993 for c2.

* 1. **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.

* 1. **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.

*h1(k)* and *h1(k)*:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dataset0 | dataset1 | dataset2 | dataset3 | dataset4 | dataset5 | dataset6 | dataset7 | dataset8 | dataset9 | 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

*h1(k)* and *h2(k)*:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dataset0 | dataset1 | dataset2 | dataset3 | dataset4 | dataset5 | dataset6 | dataset7 | dataset8 | dataset9 | 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

*h1(k)* and *h3(k)*:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dataset0 | dataset1 | dataset2 | dataset3 | dataset4 | dataset5 | dataset6 | dataset7 | dataset8 | dataset9 | 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 and 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*

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 2: Example output of the code for part 2