Imani Palmer

6/10/13

Hashing Analysis

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table | 0.3 | | 0.4 | | 0.5 | | 0.6 | | 0.7 | | 0.8 | | 0.9 | | 0.95 | | 0.98 | |
| 1009 | 1.18 | 0.50 | 1.37 | 0.90 | 1.47 | 1.65 | 1.71 | 2.27 | 2.01 | 3.83 | 2.92 | 11.03 | 3.81 | 20.60 | 8.71 | 99.08 | 9.38 | 103.45 |
| Table | 0.3 | | 0.4 | | 0.5 | | 0.6 | | 0.7 | | 0.8 | | 0.9 | | 0.95 | | 0.98 | |
| 2003 | 1.22 | 0.53 | 1.35 | 0.87 | 1.45 | 1.35 | 1.78 | 2.66 | 2.35 | 5.22 | 2.86 | 11.04 | 5.56 | 44.50 | 5.52 | 52.69 | 9.15 | 168.35 |
| Table | 0.3 | | 0.4 | | 0.5 | | 0.6 | | 0.7 | | 0.8 | | 0.9 | | 0.95 | | 0.98 | |
| 4001 | 1.26 | 0.52 | 1.33 | 0.88 | 1.48 | 1.42 | 1.83 | 2.79 | 2.16 | 4.75 | 2.74 | 11.28 | 4.26 | 36.20 | 6.04 | 89.52 | 10.07 | 228.06 |
| Table | 0.3 | | 0.4 | | 0.5 | | 0.6 | | 0.7 | | 0.8 | | 0.9 | | 0.95 | | 0.98 | |
| 8009 | 1.21 | 0.51 | 1.33 | 0.86 | 1.46 | 1.42 | 1.76 | 2.56 | 2.23 | 5.91 | 2.66 | 10.76 | 4.90 | 33.73 | 6.28 | 72.09 | 15.74 | 410.08 |
| Table | 0.3 | | 0.4 | | 0.5 | | 0.6 | | 0.7 | | 0.8 | | 0.9 | | 0.95 | | 0.98 | |
| 16001 | 1.19 | 0.50 | 1.31 | 0.84 | 1.47 | 1.42 | 1.73 | 2.48 | 2.10 | 4.72 | 2.88 | 10.49 | 5.53 | 38.26 | 8.76 | 98.55 | 16.91 | 324.06 |

Linear Probing Results

In the analysis of linear probing, there are many items to cite. First, as alpha grew larger so did the number of successful and unsuccessful searches. However, as seen above number of unsuccessful searches grew faster as alpha grew larger. It appears at alpha values at 0.6 – 0.7 is a turning point for both successful and unsuccessful searches. As shown in the graph this is a definite upward trend shown at these points. It is also interesting to note that the number of unsuccessful searches grows greatly as the alpha value increases. This is important and shows the disadvantage of linear probing. If linear probing has numerous hash collisions the method of dealing with those hash collisions, is inefficient. This is cleared up in double hashing where the increment is greater than one and can be found with ease. At first the theoretical formulas resemble the results for both successful and unsuccessful finds; however, once again around alpha values 0.6 and 0.7 they lose their accuracy. I believe this is due to the increasing possibility of duplicates. Duplicates are detrimental to the performance of linear probing because it initially hashes to the same location and then has to be incremented. If there are multiple duplicates then the incrementing results in poor hashing.

Separate Chaining

* How did the different schemes compare to each other for small alpha values and for large alpha values (and those in between)?
* Was there a significant difference in results for successful and unsuccessful searches, and, if so, why?

Double Hashing

|  |  |
| --- | --- |
| DH Successful Search | DH Unsuccessful Search |
| 1.18891648 | 1.428571429 |
| 1.277064059 | 1.666666667 |
| 1.386294361 | 2 |
| 1.52715122 | 2.5 |
| 1.719961149 | 3.333333333 |
| 2.011797391 | 5 |
| 2.558427881 | 10 |
| 3.153402393 | 20 |
| 3.99186021 | 50 |

I was unable to properly complete the double hashing method. However, I believe that the double hashing method would have followed a similar scheme of linear probing. The true difference would possibly

* How did the different schemes compare to each other for small alpha values and for large alpha values (and those in between)?
* Was there a significant difference in results for successful and unsuccessful searches, and, if so, why?
  + - DH successful search: [-ln(1-alpha)]/alpha
    - DH unsuccessful search: 1/(1-alpha)

Summation

As the table sizes increased for did the amount of unsuccessful searches. As I noted above the amount of unsuccessful search makes a noted turned especially when the alpha value is around 0.6.

* Explain the reason for this change if it occurred. Were the results for the probes and for the physical run-times consistent?
* For linear probing, if duplicate are an exception and not the rule then table will have good performance. However, if the number of duplicates approaches the number of items in the table, the the time to find a specific element grows to O(n). For separate chaining, if duplicates are allowed insertion is simple and duplicates are easily handled because they are just linked to the same place. This may cause a little extra overhead, however, this is performs better than linear probing. For double hashing, duplicates should be avoided at all costs. This is important because duplicates will have the same hash value as well as the same increment value. This will cause confusion and is a special case that should be handled. The handling of smaller or larger range of random numbers is another factor that could have affected my results. This a factor for all three tables because the number of holes created by the range of random numbers may lead to a less than efficient performance. This is because a smaller range of random numbers could possibly allow for fewer holes in the hash tables. The more clustering there is the harder it is for a table to find a proper position in the table as well as causing more time in rehashing. This is only a major problem for linear probing and double hashing.
* Did you notice anything else of interest that is worth mentioning?