

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Hashing Function Analysis | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Ben Khabazan |  | | | |
|  | | | | March 11, 2020—Data Structures, Algorithms, and Discrete Mathematics II—Christopher Esposito |  | | | |
|  | | |  | | |  | | |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | |
|  | | Understanding Hashing | | |  | |
|  | |  |  |  |  | |
|  | Hash Tables Hash table I a data structure that stores data in an associative manner which allows for a highly efficient calling, storing, and retrieving of data. In this assignment, I have used an array of size 4177 that stores Node values that hold one string format, and a link to another Node (defaulted to 0). The collision handling that I have implemented is to list the Nodes after one another such as a linked-list. This allows for collisions to be chained together and later called if needed without hassle. | | | | |  |
|  | Hash Functions The hash function is how a hash table stores data efficiently. It calculates an automated index for the data passed in and stores it within the array based on that index. If the hashing function is efficient and ‘perfect’, there would be minimal to no collision within the array, making the usage of a hash table very efficient. In this assignment, there are three total hashing algorithms used and compared in order to understand the distribution of data and its  efficiency. The outcome of the tests are stored within the spreadsheets attached with this document and includes the histogram with the chain length and the number of times repeated specified within the document. | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | |
|  | | Addition Hashing | | |  | |
|  | |  |  |  |  | |
|  | The addition hashing is the addition of the ASCII value for each character value included in the string provided as key. This makes the hashing function very inefficient for the data given as the length of the parameters are the same and make the range of numbers that can be achieved very limited. As shown in the spread sheet and the histogram, most indices are set to 0 chains as there are no data stored within them. The maximum chaining within this hashing function is 265, showing that the collision is high. The total collisions that happened, is 45432, which is the highest out of the three hashing functions used. | | | | |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | |
|  | | Multiplication Hashing | | |  | |
|  | |  |  |  |  | |
|  | Multiplication hashing indicates that the ASCII values are being  multiplied together. The exact function that I used takes in account two things, the total value of the hash value that is being increased and the value that it is being multiplied by. If the hashing value reaches the maximum long value, then it will cause many collisions. Therefore, I have a value that checks if the hash value has reached the maximum long value and divides it by the next ASCII value multiplied by the index of the character. If that has not happened, the hash value is multiplied by the ASCII value and then divided by index of the character + 2 to keep the total value down and random. This algorithm was very efficient as it spread the data evenly between the indices of the array with the maximum chain of 25 and minimum of 2. The standard deviation was 3.30 and total collision was 42155. For further evaluation visit second sheet of the spreadsheet. This hashing function was the second most efficient algorithm used in this test. | | | | |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | |
|  | | SDBM Variation Hashing | | |  | |
|  | |  |  |  |  | |
|  | During the hashing, I implemented my own variation of the SDBM hashing function that takes in the key and performs the hashing based on the size of the characters of the key. The variation reduces the size of the ASCII by 17 to change the variation level and reduce the value. Looking at the outcome of this hashing function, it has the minimum value of 3 chains, and maximum of 22. This is a bit more spread compared to the multiplication and has a lower standard deviation of 3.06. This is very important to note because the total collision was the same between the two (42155) allowing the hashing function to be the best ranked amongst the three. For more information and evaluation visit the spreadsheet attached. | | | | |  |