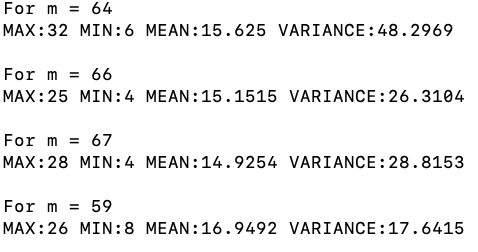
**Problem 1**

Firstly, calculate h(key) by the hash function given. Secondly, put the key value in the hash table correspondingly. If the position is occupied, put key value in the next position until the position is available. Probe sequence is the position that might be available. Finally if the position in hash table is not enough, the remaining values will not be filled in table.

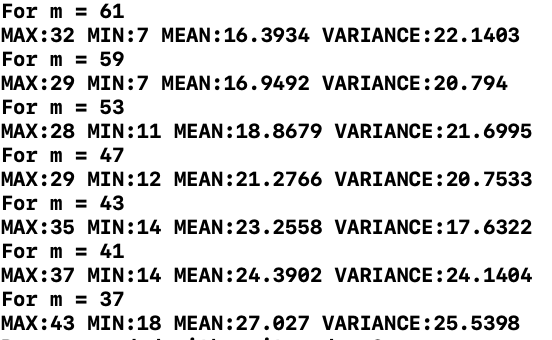
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key Value** | **h(key)** | **Probe Sequence** |  | **Final Hash Table Contents** |
| 43 | 0 | 0 | 0 | 43 |
| 23 | 6 | 6 | 1 | 0 |
| 1 | 3 | 3 | 2 | 31 |
| 0 | 1 | 1 | 3 | 1 |
| 15 | 7 | 7 | 4 | 5 |
| 31 | 2 | 2 | 5 | 7 |
| 4 | 9 | 9 | 6 | 23 |
| 7 | 5 | 5 | 7 | 15 |
| 11 | 5 | 5, 6, 7, 8 | 8 | 11 |
| 3 | 7 | 7, 8, 9, 10 | 9 | 4 |
| 5 | 0 | 0, 1, 2, 3, 4 | 10 | 3 |
| 9 | 10 | 10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, overflow |  |  |

**Problem 2**

Result in COE Linux:



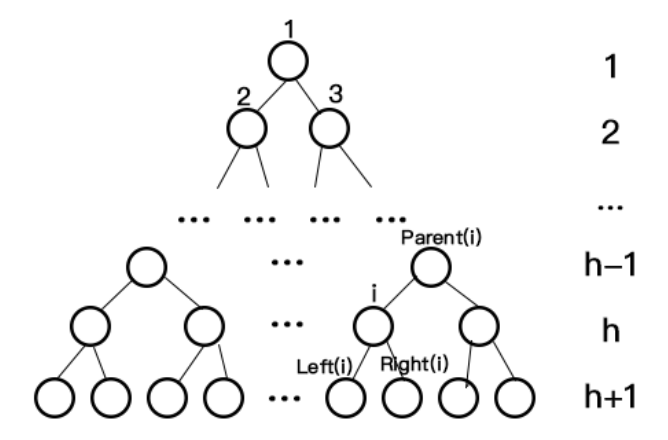
Compare different p:



I pick p as 59, because comparing with 61, 53, 47,43,41 and 37, its mean and variance are relatively small. Comparing with 64, 66 and 67, 59 is far away from 2^8. Also when m = 59, the hash table has good performance on variance, and its mean is not bad.

From the results we can find that larger the hash table size is, smaller its mean and variance are, and correspondingly lower the collision rate is. But if slot is close to 2^n, collision rate will rise.

**Problem 3**



Suppose is the element of the line, and

In line, there are elements in front of , so

And

In line, there are elements in front of when is left child of , and elements in front of when is right child of , so

When is right child, is odd number. And because index is integer, . So