Seçkin Savaşçı
Boğaziçi University
Sophomore CmpE
to_seckin206@yahoo.com.tr

July 25th, 2010

Summer Term 2010 – Cmpe250 Project 2

Implementation of Robin Hood Hashing

Robin Hood hashing [1] is a variation of double hashing. This is to say that, a key that has already been inserted in the hash table can be displaced by a new key (a key that needs to be inserted at that time) if the probe count of the new key is larger than that of the key at the current position. The displaced key then needs to be inserted to a new location as dictated by its new probe count (which is incremented by 1) as it is displaced. The analogy is that, larger probe counts deem a key to be poorer and richer keys are displaced to show favor for poorer keys. The *probe count* term related to a key refers to how many times that key has collided with other inserted keys prior to being inserted to its current location. Displacement events should also be counted as collisions for the displaced key. The net effect of Robin Hood hashing is that it reduces worst case search times in the hash table.

Implementation Related Description

Given in SRS, our goal is to implement a system that can read values from a text file named input.txt , do Robin Hood Hashing, and logs into output.txt . To reach this goal, there is no restriction in class or package usage. But I consider the general idea of having simple and working code during the implementation process. I also follow the self-documented-code paradigm during implementation.

For input, a text file will be given in such format :
<table size, positive integer value>
<value1,integer value>
<value2,integervalue>
...

For all values from input.txt(except table size) , I created node objects, and tried to insert this nodes to hash table, this simplified the method and their implementations. After all insertions are done, a simple message box is implemented for general info.

My main problem was deciding whether or not I resize the hash table if needed. But I avoid resizing , but I come up with a method that checks the validation of input.txt, simply it checks the line count and first line values, and informs the user.

I avoid to include hierarchical package-class descriptions in this SDD. But I write complete javadoc with extensive information, including private fields. (As you know by default, javadoc never includes private fields and methods)

Click to open Javadoc
Click to open the original Robin Hood Hashing article

What’s EXTRA?

- Extended JAVADOC (including private fields and methods, fully descriptive)
- Simple information GUI
- Original Article of Robin Hood Hashing
- Generic implementation for later developments (such as isActive field for lazy deletion etc)

Sample Run

Input.txt   is given as:

```
19
48
21
11
14
35
77
20
70
36
87
69
95
23
88
9
40
52
86
```

Output.txt is :

```
Node 48 Inserted @ room 10 - collision count for this node is 0
Node 21 Inserted @ room 2 - collision count for this node is 0
Node 11 Inserted @ room 11 - collision count for this node is 0
Node 14 Inserted @ room 14 - collision count for this node is 0
Node 35 Inserted @ room 16 - collision count for this node is 0
Node 77 Inserted @ room 1 - collision count for this node is 0
Node 20 collided with node 77 while inserting -  collision count for the former node is 1
Node 20 Inserted @ room 15 - collision count for this node is 1
Node 70 Inserted @ room 13 - collision count for this node is 0
Node 36 Inserted @ room 17 - collision count for this node is 0
Node 87 collided with node 11 while inserting -  collision count for the former node is 1
Node 87 Inserted @ room 7 - collision count for this node is 1
Node 69 Inserted @ room 12 - collision count for this node is 0
Node 95 Inserted @ room 0 - collision count for this node is 0
Node 23 Inserted @ room 4 - collision count for this node is 0
Node 88 collided with node 69 while inserting -  collision count for the former node is 1
Node 88 collided with node 87 while inserting -  collision count for the former node is 2
Node 88 collided with node 21 while inserting -  collision count for the former node is 3
Replacing node 21 - collision count for this node is 1
Node 88 Inserted @ room 2 - collision count for this node is 3
Node 21 collided with node 20 while inserting -  collision count for the former node is 2
Node 21 Inserted @ room 9 - collision count for this node is 2
Node 9 collided with node 21 while inserting -  collision count for the former node is 1
Node 9 collided with node 36 while inserting -  collision count for the former node is 2
Replacing node 36 - collision count for this node is 1
Node 9 Inserted @ room 17 - collision count for this node is 2
Node 36 collided with node 70 while inserting -  collision count for the former node is 2
Replacing node 70 - collision count for this node is 1
Node 36 Inserted @ room 13 - collision count for this node is 2
Node 70 collided with node 21 while inserting -  collision count for the former node is 2
Node 70 Inserted @ room 5 - collision count for this node is 2
Node 40 collided with node 88 while inserting -  collision count for the former node is 1
Node 40 collided with node 36 while inserting -  collision count for the former node is 2
Node 40 collided with node 70 while inserting -  collision count for the former node is 3
Node 40 collided with node 35 while inserting -  collision count for the former node is 4
Replacing node 35 - collision count for this node is 1
Node 40 Inserted @ room 16 - collision count for this node is 4
Node 35 collided with node 36 while inserting -  collision count for the former node is 2
Node 35 collided with node 48 while inserting -  collision count for the former node is 3
Replacing node 48 - collision count for this node is 1
Node 35 Inserted @ room 10 - collision count for this node is 3
Node 48 collided with node 36 while inserting -  collision count for the former node is 2
Node 48 collided with node 40 while inserting -  collision count for the former node is 3
Node 48 collided with node 95 while inserting -  collision count for the former node is 4
Replacing node 95 - collision count for this node is 1
Node 48 Inserted @ room 0 - collision count for this node is 4
Node 95 collided with node 87 while inserting -  collision count for the former node is 2
Node 95 collided with node 14 while inserting -  collision count for the former node is 3
Replacing node 14 - collision count for this node is 1
Node 95 Inserted @ room 14 - collision count for this node is 3
Node 14 collided with node 9 while inserting -  collision count for the former node is 2
Node 14 collided with node 77 while inserting -  collision count for the former node is 3
Replacing node 77 - collision count for this node is 1
Node 14 Inserted @ room 1 - collision count for this node is 3
Node 77 collided with node 21 while inserting -  collision count for the former node is 2
Node 77 collided with node 9 while inserting -  collision count for the former node is 3
Node 77 Inserted @ room 6 - collision count for this node is 3
Node 52 collided with node 95 while inserting -  collision count for the former node is 1
Node 52 collided with node 11 while inserting -  collision count for the former node is 2
Replacing node 11 - collision count for this node is 1
Node 52 Inserted @ room 11 - collision count for this node is 2
Node 11 collided with node 9 while inserting -  collision count for the former node is 2
Node 11 collided with node 23 while inserting -  collision count for the former node is 3
Replacing node 23 - collision count for this node is 1
Node 11 Inserted @ room 4 - collision count for this node is 3
Node 23 collided with node 20 while inserting -  collision count for the former node is 2
Node 23 collided with node 87 while inserting -  collision count for the former node is 3
Replacing node 87 - collision count for this node is 2
Node 23 Inserted @ room 7 - collision count for this node is 3
Node 87 Inserted @ room 3 - collision count for this node is 2
Node 86 collided with node 35 while inserting -  collision count for the former node is 1
Node 86 collided with node 23 while inserting -  collision count for the former node is 2
Node 86 collided with node 11 while inserting -  collision count for the former node is 3
Node 86 collided with node 14 while inserting -  collision count for the former node is 4
Node 86 collided with node 9 while inserting -  collision count for the former node is 5
Replacing node 9 - collision count for this node is 3
Node 86 Inserted @ room 17 - collision count for this node is 5
Node 9 collided with node 95 while inserting -  collision count for the former node is 4
Node 9 collided with node 87 while inserting -  collision count for the former node is 5
Replacing node 87 - collision count for this node is 3
Node 9 Inserted @ room 3 - collision count for this node is 5
Node 87 Inserted @ room 18 - collision count for this node is 3
```

References

1 Celis, Pedro (1986). Robin Hood hashing. Technical Report Computer Science Department, University of Waterloo CS-86-14.