Wednesday, 23 November 2016

MPCS 51100 HW3 Report

Yan Cui

- **P1**

I've implemented three hash function for strings: BKDRHash, SUMHash and DJBHash.

The dictionary also support command "rehash", you can change the number of bins and the hash function of the hash table.

To test these three hash function, I tested the minimum and maximum depth of bins when the load factor is around 0.5, and here's the result:

```
HashFunc: BKDRHash    Op: ReHash    Key:    LF: 0.474415    OCC([min, max] depth): [0, 4]
HashFunc: SUMHash    Op: ReHash    Key:    LF: 0.474415    OCC([min, max] depth): [0, 7]
HashFunc: DJBHash    Op: ReHash    Key:    LF: 0.474415    OCC([min, max] depth): [0, 5]
```

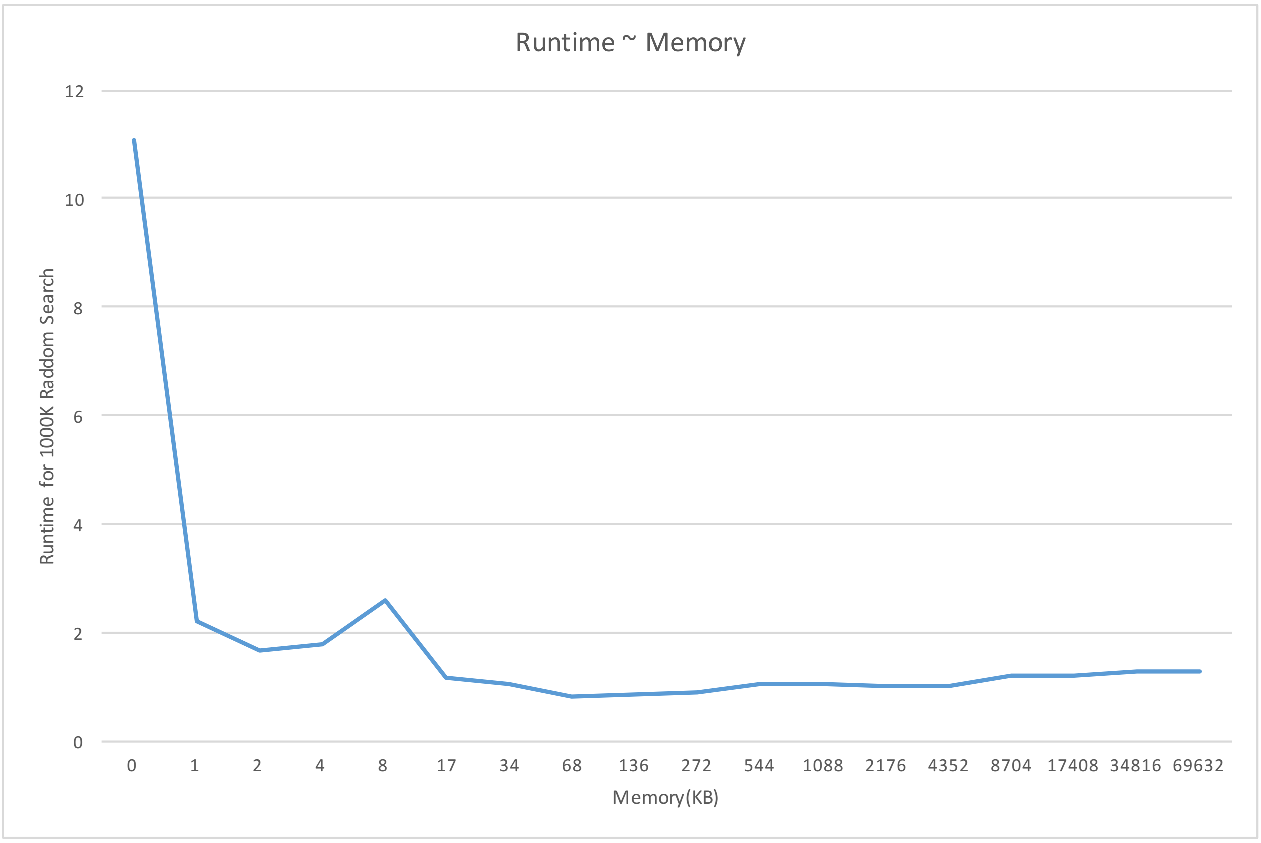
From the result, we can see that the BKDRHash assign the strings more evenly than the other two functions.

- **P2**

I use a Hash Table to remove duplicates in an array. I check from the smallest index, if the element is already in the hash table, then find a unique element from the back of array and swap them, then all the unique element will locate at the begin of the array.

- **P3**

I change the number of bins from 1 to 20000 and compare the runtime. Here's the plot of runtime~memory curve.



Based on the curve, I choose 512 as my optimal number of bins and here's the runtime comparing with Figure3 in HW2.

Runtime with table in Figure 3: 9.218189

Runtime with hash table: 12.024498

The hash kernel is a little bit slower but it's efficient in memory, it only need 272KB while table in Figure 3 need 64945KB.