Project 1

Name: Xiaowei Tan

Student ID: 69203272

Theoretical Analysis

1. Chained Hash

The pseudo code is shown as below:

```
insert key, val:
    h = hash(key)
    list = H[h]
    if list is null
        add key, val to new list l
        H[h] = l
    else
        if key in list then replace original value with val
        else add key, val to list

search key:
    h = hash(key)
    list = H[h]
    if list is null or key not in list
        return key not found
    else
        return the value

delete key:
    h = hash(key)
    list = H[h]
    if list is null or key not in list
        return key not found
    else
        delete the value
```

The time complexity

$= O(1 + E[length\ of\ H[h(k)]])$

$= O(1 + E[number\ of\ keys\ colliding\ with\ k])$

$= O(1 + Sum\ of\ probility\ of\ other\ keys\ collides\ with\ k$

$= O(1 + (n-1) * 1/N$

$= O(1 + \alpha)$

2. Linear Hash

The pseudo code is shown as below:

```
insert key, val:
    h = hash(key)
    while H[h] not empty and contains a key k not equal key
        h = (h+1)%size
    H[h] = val

search key:
    h = hash(key)
    while H[h] not empty and contains a key k not equal key
        h = (h+1)%size
    if H[h] is not empty
        return the value
    else
        return key not found

delete key:
    h = hash(key)
    while H[h] not empty and contains a key k not equal key
        h = (h+1)%size
    if H[h] is empty
        return key not found
    j = h+1
    while H[j] is not empty
        if hash(H[j].key) not in range [i+1, j]
            move H[j] to H[i]
            i = j
        j++
    delete H[i]
```

The time complexity

$\leq \sum L * L/c^L$

$= \sum_{L=1}^{n} L^2/c^L \leq \sum_{L=1}^{\infty} L^2/c^L$

$= O(1)$

3. Cuckoo Hash

The pseudo code is shown as below:

```
insert key, value:
    t=0
    while (k,v) is a nonempty pair:
        (k,v) ↔ Ht[ht(k)]
        t=1-t

search key:
    return H0[h0(key)] or H1[h1(key)]

delete key:
    delete H0[h0(key)] or H1[h1(key)]
```

The time complexity for search is $O(1)$

4. Quadratic Hash

The pseudo code is shown as below:

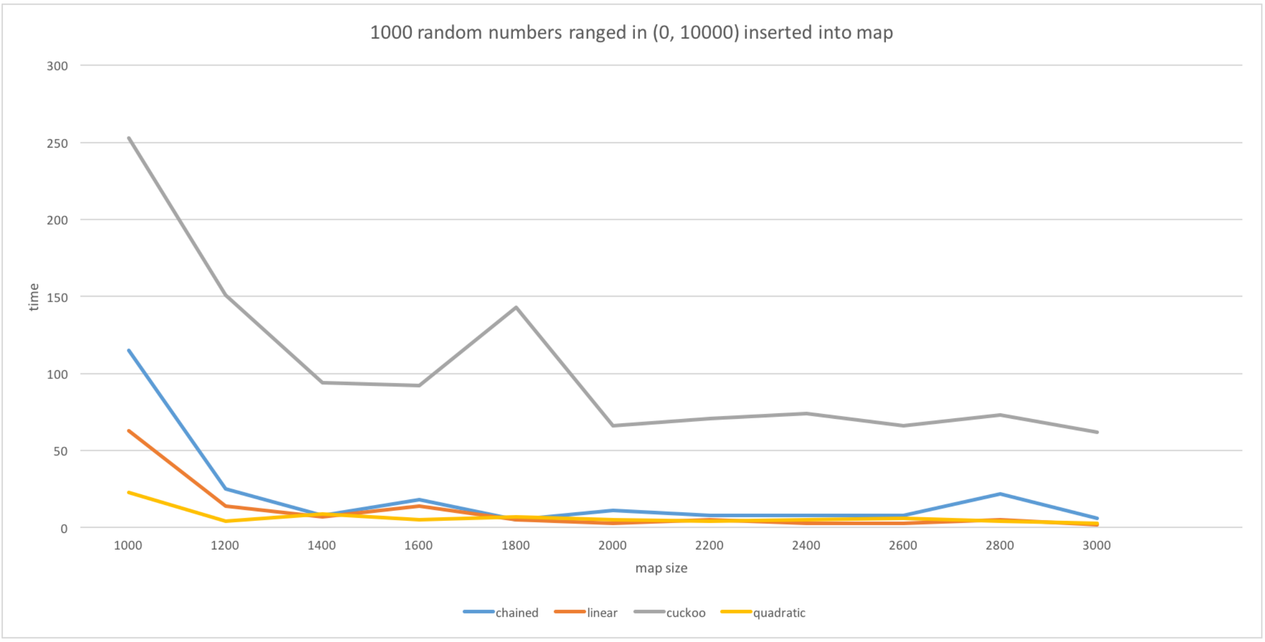
```
insert key, value:
    h = hash(key)
    i = 0
    while H[(h+i*i)%size] not empty and contains key not equal key and the
slot is not used
        i++
    H[(h+i*i)%size] = value

search key:
    h = hash(key)
    i = 0
    while H[(h+i*i)%size] not empty and contains key not equal key
        i++
    if H[(h+i*i)%size] empty or is not used
        return key not found
    return the value

delete key:
    h = hash(key)
    i = 0
    while H[(h+i*i)%size] not empty and contains key not equal key
        i++
    if H[(h+i*i)%size] empty or is not used
        return key not found
    set the slot to cleared status
```

The reason that I choose the quadratic hash as the additional algorithm is that it is most similar with linear hash and it doesn't need extra space. At the same time, I think it could be better than linear hash because instead of storing all the conflicting keys in the same part, it stores them seperately by quadratic addition.

Experimental Analysis

1. Insert Operation

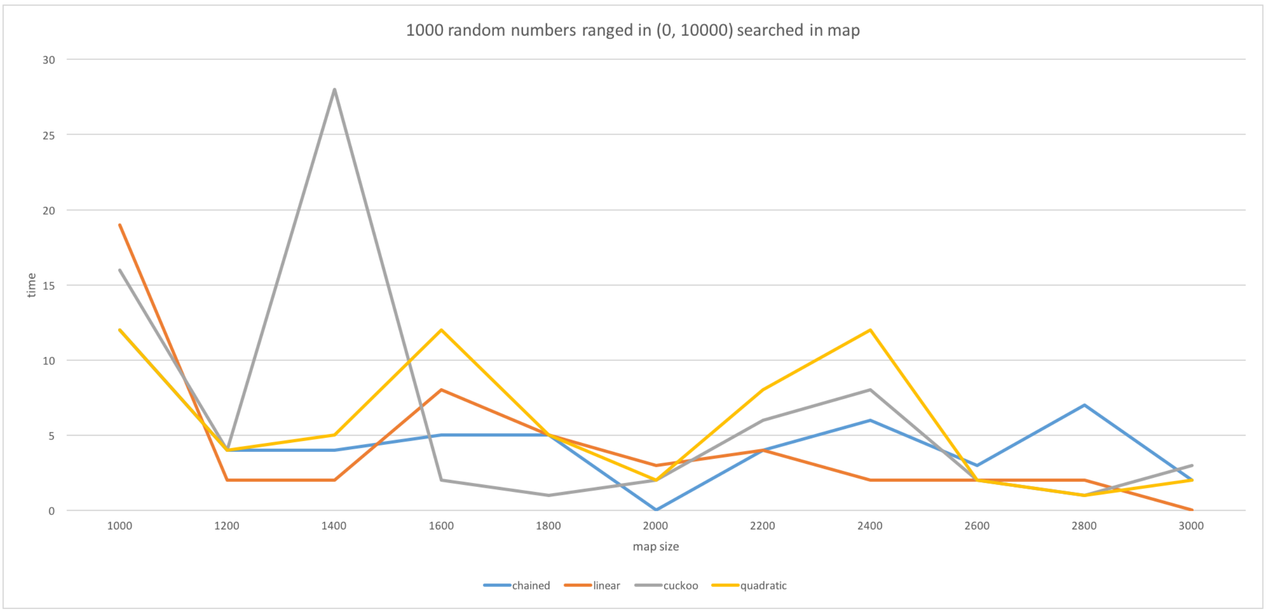
I tested the insert operation in the four hash methods with a dataset which contains 1000 random numbers ranged from 0 to 10000. And each insert operation is tested with different load factor ranged from 1 to 1/3 for 100 times and I sum the time up to get a line chart.



We can see from the chart that cuckoo hash's insert operation takes the most of time and all the time of operations decreases as the load factor goes down. They all comply with the theoretical analysis.

2. Search Operation

I tested the search operation in the four hash methods with a dataset which contains 1000 random numbers ranged from 0 to 10000. And each search operation is tested with different load factor ranged from 1 to 1/3 for 100 times and I sum the time up to get a line chart.



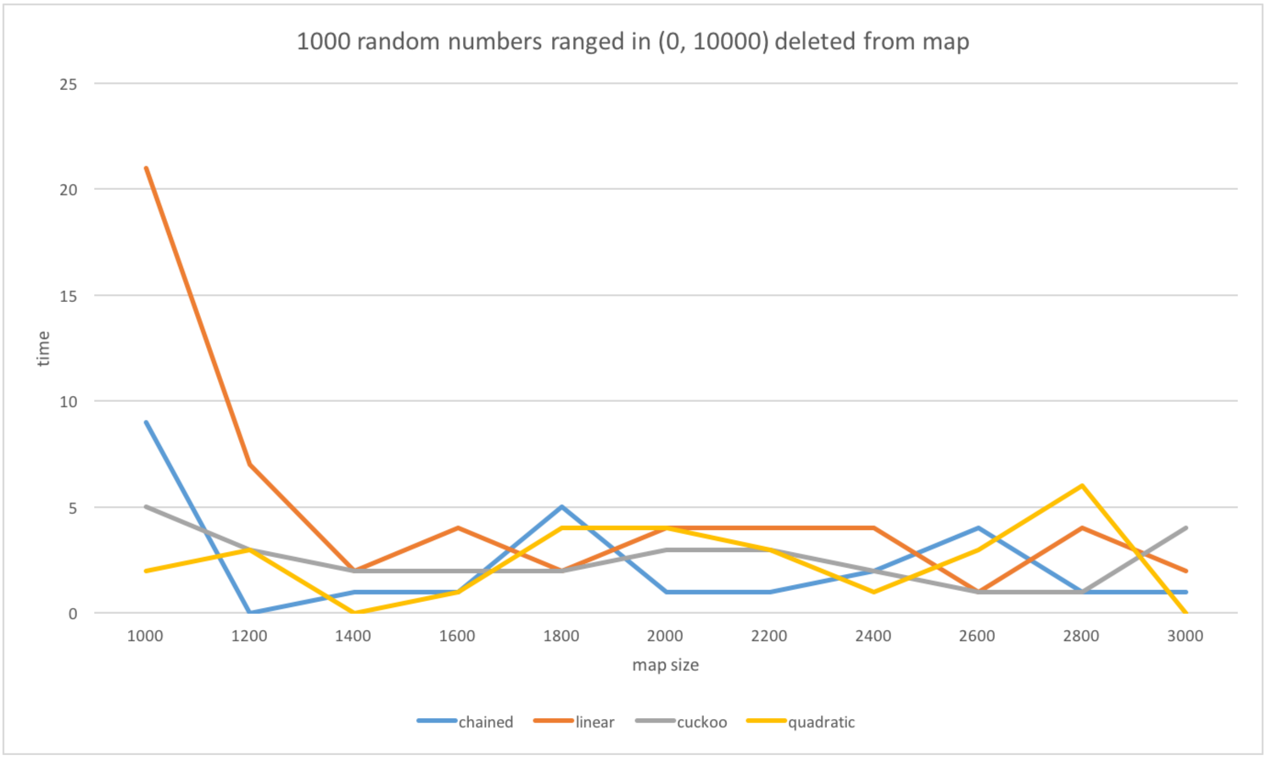
We can see from the chart that all the time of operations decreases as the load factor goes down and it is the same with theoretical analysis.

But the time of cuckoo hash is not the same. When it is in the worst case, cuckoo hash's time is not the least.

And the search time of quadratic hash is kind of more than others, because I implement it in lazy delete and it will cause the inefficiency in search.

3. Delete Operation

I tested the delete operation in the four hash methods with a dataset which contains 1000 random numbers ranged from 0 to 10000. And each delete operation is tested with different load factor ranged from 1 to 1/3 for 100 times and I sum the time up to get a line chart.



We can see from the chart that all the time of operations decreases as the load factor goes down and it is the same with theoretical analysis.

And linear hash can take more time to do the delete because the eager delete will move some elements. And quadratic hash can take less time because it just marks the slot as cleared instead of actually remove the element.