ECE374 Fall2020

Homework7

Name: Zhang Yichi 3180111309

Nov. 15^{th} 2020

11.1-1

Traverse the direct-address table T, compare each one of the element and find the largest one. The worst-case time complexity is O(m) where m is the length of T.

11.1-2

```
class Direct_Table_bit_vector:
    def __init__(self,K,m):
        self.T = [0]*m
        for i in K:
            self.T[i] = 1

    def search(self,k):
        if self.T[k]:
            return k
        else:
            return None

    def insert(self,x):
        self.T[x] = 1

    def delete(self,x):
        self.T[x] = 0
```

The i-th slot corresponds to the key i. If the bit in the i-th slot is 1, it means that there is an element i, 0 otherwise. As the codes shown above, the dictionary operations run in O(1) time.

11.2-1

let X denotes collide or not, Y denotes the total number of collision. Therefore, $Y = \sum_{k \neq l} X$ and $E[Y] = E[\sum_{k \neq l} X] = \sum_{k \neq l} E[X] = \sum_{k \neq l} \frac{1}{m} = \binom{n}{2} \frac{1}{m} = \frac{n(n-1)}{2} \frac{1}{m} = \frac{n(n-1)}{2m}$

11.2-3

For insertions and deletions, the time complexity does not change, still $\Theta(1+\alpha)$, one for computing $h(k)$, α for rearrange the items after. For successful and unsuccessful searches, the time complexity becomes $\Theta(1 + lg(\alpha))$ because the time complexity for searching in a sorted list is $O(lg(n))$.

11.4-1

```
def linear_probing_insert(T,k):
    i = 0
    while (i < 11):
        j = (k + i) % 11
        if T[j] == None:
            T[j] = k
            return j
        else:
            i = i + 1

def quadratic_probing_insert(T,k):
    i = 0
    while (i < 11):
        j = (k+i+3*i**2) % 11
        if T[j] == None:
            T[j] = k
            return j
        else:
            i = i + 1

def double_hashing_insert(T,k):
    i = 0
    while (i < 11):
        j = (k + i*(1 + (k % 10))) % 11
        if T[j] == None:
            T[j] = k
            return j
        else:
            i = i + 1

m = 11
linear = [None]*m
quadratic = [None]*m
double = [None]*m
A = [10, 22, 31, 4, 15, 28, 17, 88, 59]
for i in A:
    linear_probing_insert(linear,i)
    quadratic_probing_insert(quadratic,i)
    double_hashing_insert(double,i)
```

$h'(k) = k$, $m = 11$, $c_1 = 1$, $c_2 = 3$, $h_1(k) = k$, $h_2(k) = 1 + (k \mod (m-1))$
Insert 10, 22, 31, 4, 15, 28, 17, 88, 59.

Linear probing: $h(k,i) = (h'(k) + i) \mod m = (k + i) \mod 11$
The result is: [22, 88, None, None, 4, 15, 28, 17, 59, 31, 10]

Quadratic probing: $h(k,i) = (h'(k) + c_1 i + c_2 i^2) \mod m = (k + i + 3i^2) \mod 11$
The result is: [22, None, 88, 17, 4, None, 28, 59, 15, 31, 10]

Double hashing: $h(k,i) = (h_1(k) + ih_2(k)) \mod m = (k + i(1 + (k \mod 10))) \mod 11$
The result is: [22, None, 59, 17, 4, 15, 28, 88, None, 31, 10]

11.4-2

```
Hash-Deletion(T,k)
i = 0
repeat
    j = h(k,i)
    if T[j] == k
        T[j] = DELETED
        return j
    else i = i + 1
until i == m or T[j] == NIL
error "no k found"
```

```
Modified-Hash-Insert(T,k)
i = 0
repeat
    j = h(k,i)
    if T[j] == NIL or T[j] == DELETED
        T[j] = k
        return j
    else i = i + 1
until i == m
error "hash table overflow"
```