Solutions

11.1-2

A bit vector is simply an array of bits (0s and 1s). A bit vector of length m takes much less space than an array of m pointers. Describe how to use a bit vector to represent a dynamic set of distinct elements with no satellite data. Dictionary operations should run in $O(1)$ time.

Solution

Consider a bit vector of length m to represent a dynamic set, the m bits of the vector corresponds to m slots of hash table.
Assuming that initially the bit vector is all 0s and there is no data available.When an element arrives at key k, set the kth bit to 1 which signifies that the slot at this position is non-empty.
All the dictionary operation such as Search, Insert, Delete in this case will be $O(1)$.

11.2-1

Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming simple uniform hashing, what is the expected number of collisions? More precisely, what is the expected cardinality of $\{\{k,l\} : k \neq l \quad and \quad h(k) = h(l)\}$

Solution:

Let us define the Indicator random variable $X_{lk} = I\{h(k) = h(l)\}$,where k≠l
The probability of X_{lk} in this case where we are using the simple hash function and have m slots is:

$$Pr\{X_{lk} = I\} = Pr\{h(k) = h(l)\} = \frac{1}{m}$$

According to Lemma 5.1 (CLRS Page 118)
The expectation will be E[X_{lk}]=$\frac{1}{m}$

Now let's consider the random variable T_{col} which defines the total number of collisions, i.e. ∀ k≠l, h{k}=h{l}, then

$$T_{col} = \sum_{k \neq l} \{X_{lk}\}$$

The expected number of collisions would be

$$E[T_{col}] = E[\sum_{k \neq l} \{X_{lk}\}]$$
$$= \sum_{k \neq l} E[X_{lk}]$$

Using the value of E[X_{lk}]=$\frac{1}{m}$ from above

$$E[T_{col}] = \sum_{k \neq l} \frac{1}{m}$$

Using Permutation and combinations, to choose two keys 'k' and 'l' from n distinct values such that $h(k) = h(l)$ will be $\binom{n}{2}$, so the above equation can be rewritten as:

$$E[T_{col}] = \binom{n}{2} \frac{1}{m}$$
$$\therefore, \ \textit{Expected no. of collisions}, \ E[T_{col}] = \frac{n(n-1)}{2m}$$

11.4-1

Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m-1))$

Solution

Linear Probing:

The hash function is h(k,i)=($h^{'}(k)$+i)$mod\, m$ =(k+i)$mod\, m$
Now, calculating the slot where the key will be inserted for all the incoming keys.
1. $h(10, 0) = (10 + 0) \pmod{11} = 10 \ \therefore$ **T[10]=10**
2. $h(22, 0) = (22 + 0) \pmod{11} = 0 \ \therefore$ **T[0]=22**

3. $h(31,0) = (31+0) \pmod{11} = 9 \;\therefore$ **T[9]=31**
4. $h(4,0) = (4+0) \pmod{11} = 4 \;\therefore$ **T[4]=4**
5. $h(15,0) = (15+0) \pmod{11} = 4$ As T[4] is occupied, it will be a collision. So, we probe again with i=1, we have
$h(15,1) = (15+1) \pmod{11} = 5 \;\therefore$ **T[5]=15**
6. $h(28,0) = (28+0) \pmod{11} = 6 \;\therefore$ **T[6]=28**
7. $h(17,0) = (17+0) \pmod{11} = 6$ As T[6] is occupied, it will be a collision. So, we probe again with i=1, we have
$h(17,1) = (17+1) \pmod{11} = 7 \;\therefore$ **T[7]=17**
8. $h(88,0) = (88+0) \pmod{11} = 0$ As T[0] is occupied it will be a collision. So, we probe again with i=1, we have
$h(88,1) = (88+1) \pmod{11} = 1 \;\therefore$ **T[1]=88**
9. $h(59,0) = (59+0) \pmod{11} = 4$ As T[4] is occupied it will be a collision. So,we probe again with i=1, we have
$h(59,1) = (59+1) \pmod{11} = 5$ Another colision.So, we probe again with i=2,
$h(59,1) = (59+2) \pmod{11} = 6$ Another colision.So, we probe again with i=3,
$h(59,1) = (59+3) \pmod{11} = 7$ Another colision.So, we probe again with i=4,
$h(59,1) = (59+4) \pmod{11} = 8 \;\therefore$ **T[8]=59**

T : $22->88->NULL->NULL->4->15->28->17->59->31->10$
Note: A NULL represent that a slot in empty.

Quadratic Probing:

The hash function is h(k,i)=$(h^{'}(k)+c_1 i+c_2 i^2) mod\ m$=(k+i+3$i^2$)$mod\ m$
Now we calculate the slot where the key will be inserted for all the incoming keys.
1. $h(10,0) = (10+0+0) \pmod{11} = 10 \;\therefore$ **T[10]=10**
2. $h(22,0) = (22+0+0) \pmod{11} = 0 \;\therefore$ **T[0]=22**
3. $h(31,0) = (31+0+0) \pmod{11} = 9 \;\therefore$ **T[9]=31**
4. $h(4,0) = (4+0+0) \pmod{11} = 4 \;\therefore$ **T[4]=4**
5. $h(15,0) = (15+0+0) \pmod{11} = 4$ As T[4] is occupied we probe again with i=1, we have
$h(15,1) = (15+1+3) \pmod{11} = 8 \;\therefore$ **T[8]=15**
6. $h(28,0) = (28+0+0) \pmod{11} = 6 \;\therefore$ **T[6]=28**
7. $h(17,0) = (17+0+0) \pmod{11} = 4$ As T[6] is occupied we probe again with i=1, we have
$h(17,1) = (17+1+3) \pmod{11} = 10$,We probe again with i=2 as T[10] is also occupied,
$h(17,2) = (17+2+12) \pmod{11} = 9$, again occupied ,probing with i=3
$h(17,3) = (17+3+27) \pmod{11} = 3$, $\therefore$ **T[3]=17**
8. $h(88,0) = (88+0+0) \pmod{11} = 8$ As T[8] is occupied we probe again

with i=1, we have
$h(88,1) = (88+1+3) \pmod{11} = 4$ (again occupied), Probing with i=3 we have
$h(88,3) = 8$ (occupied), i=4$->$ $h(88,4) = 8$ (occupied), we keep on trying and finally probing with i=8 yields
$h(88,8) = 2$ $\therefore$ **T[2]=88**
9. $h(59,0) = (59+0+0) \pmod{11} = 4$ As T[4] is occupied we probe again with i=1, we have
$h(59,1) = (59+1+3) \pmod{11} = 8$,as T[8] is occupied we probe again with i=2, now
$h(59,2) = (59+2+12) \pmod{11} = 7$ $\therefore$ **T[7]=59**
T : $22->NULL->88->17->4->NULL->28->59->15->31->10$
Note: A NULL represent that a slot in empty.

Double Hashing:

The hash function is h(k)=$(h_1(k)+ih_2(k)) mod\ m$ = (k+i*{1+k (mod)(m-1)}) (mod)m
Now we calculate the slot where the key will be inserted for all the incoming keys.
1. $h(10,0) = (10+0.h_2(10)) \pmod{11} = 10$ $\therefore$ **T[10]=10**
2. $h(22,0) = (22+0.h_2(22)) \pmod{11} = 0$ $\therefore$ **T[0]=22**
3. $h(31,0) = (31+0.h_2(31)) \pmod{11} = 9$ $\therefore$ **T[9]=31**
4. $h(4,0) = (4+0.h_2(4)) \pmod{11} = 4$ $\therefore$ **T[4]=4**
5. $h(15,0) = (15+0.h_2(15)) \pmod{11} = 4$ As T[4] is occupied we probe again with i=1, we have
$h(15,1) = (15+1.h_2(15)) \pmod{11} = (15+(1+15 \pmod{10})) \pmod{11} = 10$
As T[10] is again occupied we probe again with i=2 then we have
$h(15,2) = 5$ $\therefore$ **T[5]=15**
6. $h(28,0) = (28+0.h_2(28)) \pmod{11} = 6$ $\therefore$ **T[6]=28**
7. $h(17,0) = (17+0.h_2(17)) \pmod{11} = 6$ As T[6] is occupied we probe again with i=1, $h(17,1) = 3$ $\therefore$ **T[3]=17**
8. $h(88,0) = (88+0.h_2(88)) \pmod{11} = 0$ As T[0] is occupied we probe again with i=1, $h(88,1) = 9->$again occupied, now probing with i=2
$h(88,2) = 7$ $\therefore$ **T[7]=88**
9. $h(59,0) = (59+0.h_2(59)) \pmod{11} = 4$ As T[4] is occupied we probe again with i=1, $h(59,1) = 3->$again occupied, now with i=2
$h(59,2) = 2$
$\therefore$ **T[2]=59**

T : $22->NULL->59->17->4->15->28->88->NULL->31->10$

11.4-2

Write pseudocode for HASH_DELETE as outlined in the text, and modify HASH_INSERT to handle the special value DELETED.

Solution

```
#PseudoCode
def HASH_DELETE(T,k)
        i = 0
        repeat
                j = h(k,i)
                if T[j] == k
                #if key is found mark that slot as DELETED, instead of NIL
                        T[j] = DELETED
                        return
                else
                        i += 1
        until T[j] == NIL or i == m
        #if execution comes here it means the key was not found
        return NIL
```

The HASH_INSERT function with the special value DELETED

```
#PseudoCode
def 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 += 1
        until i equals m
        #if the execution comes here it means no empty slot was found
        error "hash table overflow"
```

11.4-3

Consider an open-address hash table with uniform hashing. Give upper bounds on the expected number of probes in an unsuccessful search and on the expected number of probes in a successful search when the load factor is $\frac{3}{4}$ and when it is $\frac{7}{8}$

Solution

For Load factor $\alpha = \frac{3}{4}$:

The expected number of probes in an **unsuccessful** search with load factor $\frac{3}{4}$, i.e the upper bound on the number of probes $= \frac{1}{1-\frac{3}{4}} = 4$

The expected number of probes in a **successful** search with load factor $\frac{3}{4}$ i.e the upper bound on the number of probes $= \frac{1}{\frac{3}{4}} \ln \frac{1}{1-\frac{3}{4}} = 1.84$

For Load factor $\alpha = \frac{7}{8}$:

The expected number of probes in an **unsuccessful** search with load factor $\frac{7}{8}$, i.e the upper bound on the number of probes $= \frac{1}{1-\frac{7}{8}} = 8$

The expected number of probes in a **successful** search with load factor $\frac{7}{8}$ i.e the upper bound on the number of probes $= \frac{1}{\frac{7}{8}} \ln \frac{1}{1-\frac{7}{8}} = 2.37$