Chapter 11. Hash Tables

11.1-1.
Finding the maximum element of S is just traversing down from the highest slot of the direct-address table.
The worst-case arise if S is empty, which takes O(m).

11.1-2.
A bit vector of length m can represent presence of m elements by marking each bit position by 1 if and only if the corresponding element is present.
With the bit vector in which the single bit corresponding to the given element is turned on, SEARCH can be done with bitwise OR operation, INSERT can be done with turning on that bit, DELETE can be done with turning off that bit.

11.1-3.
Storing objects mapped with the specific key into a doubly linked list would suffice to provide INSERT, DELETE, SEARCH operations in O(1) time.

11.1-4.
Use an additional array S. Each object stored in the huge array will have the key value and a pointer to an element of the stack, which is a pointer back to the corresponding address of the object in the huge array.
INSERT can be done by pushing an element to the stack contains a pointer to the address having the value of the key being inserted. The address of the key in the huge array would contain a pointer to the address of the object in the stack being inserted.
SEARCH can be done by checking whether the address has the pointer to the element in the stack that contains the pointer back to the initially seeked address.
DELETE can be done by invalidating the element pointed by the sought

element, followed by popping an element from the stack and inserting the popped element to the invalidated slot, and updating the pointer in the huge array that points the popped element accordingly.
Each of these operations take O(1) time.

11.2-1.
The expected number of times that the key that is hashed in the i-th order is collided by other keys is $\frac{n-i}{m}$. Therefore, the expected total number of collisions is

$$\sum_{i=1}^{n} \frac{n-i}{m} = \frac{n^2-n}{2m}.$$

11.2-2.
0 : -
1 : 10 - 19 - 28
2 : 20
3 : 12
4 : -
5 : 5
6 : 33 - 15
7 : -
8 : 17

11.2-3.
Unsuccessful searches : $\Theta(1+\alpha)$.
Successful searches : Same with unsorted case, $\Theta(1+\alpha)$.
Insertions : Same with successful searches, $\Theta(1+\alpha)$.
Deletions : Same with successful searches, $\Theta(1+\alpha)$.

11.2-4.
A free slot can be implemented as a doubly linked list of all free slots in the table. The flag in each slot can represent the presence of the element in the hash table. A used slot contains an element and a pointer to the next element that hashes into this slot.
INSERTION : If the hashed-to slot is free, remove the slot from the free list (to provide this in O(1) time, the free list must be doubly linked) and store the element there. If the hashed-to slot is occupied, check whether its key also slots to the same slot. If so, allocate a free slot for the new element and

put this slot at the head of the list pointed to by the hashed-to slot. If not, move it to a new slot by allocating one from the free list, copying the old slot's contents to the new slot, and updating the pointer accordingly.
DELETION : First check if the hashed-to slot contains only single element. If so, then the list will be empty afterwards, so insert the hashed-to slot into the free list and update the flag accordingly, and delete the element from the hashed-to slot.
SEARCH : Same with the original version.
All of the operations above is O(1).

11.2-5.
Since each of the m - 1 slots contains at most n - 1 elements, the number of remaining elements in the universe is at least $|U| - (m-1)(n-1) > n + m - 1 \geq n$, so the universe has a subset of size n consisting of keys that all hash to the same slot.

11.2-6.
Pick a number from 1 to L uniformly at random. Repeat this until the picked number is not greater than the number of elements stored in the hashed-to slot. Any element in the hash table will be selected equally likely. The expected number of times required for the process satisfies $\mathbb{E}[X] = \frac{\alpha}{L}(1+\alpha) + (1-\frac{\alpha}{L})(1+\mathbb{E}[X])$, hence we have $\mathbb{E}[X] = L(\frac{\alpha}{L} + \frac{1}{\alpha}) = O(L(1+\frac{1}{\alpha}))$.

11.3-1.
We can check if the hash value of the key is present in the linked list of hashed values, instead of comparing long character strings.

11.3-2.
Use the remainder divided by m of the radix-128 number representation of the string.

11.3-3.
In this case, hash value of the string just becomes the sum of the character values mod m. If we implement a dictionary of words as a hash table, this property would be undesirable because the words with same permutation are guaranteed to be collided.

11.3-4.

h(61) = 700, h(62) = 318, h(63) = 936, h(64) = 554, h(65) = 172.

11.3-5.
We have the following lemma: the total number of collisions is minimized when the number of elements that map to each slot is $\frac{|U|}{|B|}$. To prove this, let's call the slot j as underloaded if the number of elements mapped is less than $\frac{|U|}{|B|}$, and call the slot j as overloaded if the number of elements mapped is more than $\frac{|U|}{|B|}$. If we break a balanced situation in which all slots are balanced by moving elements from underloaded slots to overloaded slots, we show that such move increases number of collisions.
If we move an element from an underloaded slot having u_j elements to an overloaded slot having u_k elements, the number of collisions changed is $\frac{(u_j-1)(u_j-2)}{2} + \frac{(u_k+1)u_k}{2} - \frac{(u_j-1)u_j}{2} + \frac{(u_k-1)u_k}{2} = u_k - u_j + 1 > 0$. Therefore we proves the lemma above.
Therefore, for any hash function, the total number of collisions must be at least $|B|\frac{|U|}{|B|}(\frac{|U|}{|B|} - 1)\frac{1}{2}$. Since the number of pairs of distinct elements is $\frac{|U|(|U|-1)}{2}$, the number of collisions per pair of distinct elements must be at least $\frac{\frac{|U|}{|B|}-1}{|U|-1} > \frac{1}{|B|} - \frac{1}{|U|}$. Therefore the bound ϵ for the probability of collision for any pair of distinct elements cannot be less than $\frac{1}{|B|} - \frac{1}{|U|}$.

11.3-6.
By exercise 31-4.4, the number of $h_b(y)$ that $h_b(x)$ collides with is at most $n-1$. Since there are total of p possible values that h_b can take, the probability of collision is bounded above by $\frac{n-1}{p}$. This holds for any b, so $\mathcal{H}$ is $\frac{n-1}{p}$-universal.

11.4-1.
LInear probing:
0 : 22
1 : 88
2 : -
3 : -
4 : 4
5 : 15
6 : 28
7 : 17

8 : 59
9 : 31
10 : 10

Quadratic probing:
0 : 22
1 : -
2 : 88
3 : 17
4 : 4
5 : -
6 : 28
7 : 59
8 : 15
9 : 31
10 : 10

Double hashing:
0 : 22
1 : -
2 : 59
3 : 17
4 : 4
5 : 15
6 : 28
7 : 88
8 : -
9 : 31
10 : 10

11.4-2.

Algorithm 1: HASH-DELETE(T, k)

```
i = 0;
while $T[j] \neq NIL$ and $i \neq m$ do
    j = h(k, i);
    if $T[j] == k$ then
        T[j] = DELETED;
        return j;
    else
        i += 1;
    end
end
error "element does not exist";
```

Algorithm 2: HASH-INSERT(T, k)

```
i = 0;
while $i \neq m$ do
    j = h(k, i);
    if $T[j] == NIL$ or $T[j] == DELETED$ then
        T[j] = k;
        return j;
    else
        i += 1;
    end
end
error "hash table overflow";
```

11.4-3.
For $\alpha = \frac{3}{4}$, unsuccessful search probes 4 nodes in average, successful search probes 1.85 nodes in average.
For $\alpha = \frac{7}{8}$, unsuccessful search probes 8 nodes in average, successful search probes 2.38 nodes in average.

11.4-4.
The least common multiplier is $l = \frac{mh_2(k)}{d}$, therefore if we fix k then $h(k, i)$ has a period of $\frac{m}{d}$.

11.4-5.

$$\frac{1}{1-\alpha} = \frac{2}{\alpha}\ln\frac{1}{1-\alpha} \Rightarrow \alpha \simeq 0.7153.$$

11.5-1.

$$p(n,m) = \frac{m}{m}\cdots\frac{m-n+1}{m}.$$

$$(m-i)(m-n+i) \le (m-\frac{n}{2})^2 \Rightarrow p(n,m) \le \frac{m(m-\frac{n}{2})^{n-1}}{m^n}$$
$$= (1-\frac{n}{2m})^{n-1} \le (e^{-\frac{n}{2m}})^{n-1} = e^{-\frac{n(n-1)}{2m}}.$$

11-1.
a. If we let X be the random variable denoting the number of probes in an unsuccessful search, then $P(X > k) \le \alpha^k \le 2^{-k}$.
b. Substituting $k = 2\lg n$ into the result of a. gives $P(X > k) \le \frac{1}{n^2} = O(\frac{1}{n^2})$.
c. $P(X > 2\lg n) \le P(X_1 > 2\lg n) + \cdots + P(X_n > 2\lg n) \le n \cdot \frac{1}{n^2} = \frac{1}{n}$.
d.

$$\mathbb{E}[X] = \sum_{k=1}^{n} kP(X=k) = \sum_{k=1}^{\lceil 2\lg n\rceil} kP(X=k) + \sum_{k=\lceil 2\lg n\rceil+1}^{n} kP(X=k)$$

$$\le \sum_{k=1}^{\lceil 2\lg n\rceil} \lceil 2\lg n\rceil P(X=k) + \sum_{k=\lceil 2\lg n\rceil+1}^{n} nP(X=k)$$
$$= \lceil 2\lg n\rceil P(X \le \lceil 2\lg n\rceil) + nP(X \ge \lceil 2\lg n\rceil + 1)$$
$$\le \lceil 2\lg n\rceil \cdot 1 + n \cdot \frac{1}{n} = O(\lg n).$$

11-2.
a. The probability that a particular key is hashed to a particular slot is $\frac{1}{n}$. The probability that k keys are inserted into the same slot and all other keys are inserted elsewhere is $(\frac{1}{n})^k(1-\frac{1}{n})^{n-k}$. Since there are $\binom{n}{k}$ ways to choose such k keys,

$$Q_k = (\frac{1}{n})^k(1-\frac{1}{n})^{n-k}\binom{n}{k}.$$

b. Let X_i be the random variable denoting the number of keys that hash to slot i. Then

$$P_k = P(M = k) = P(\max_{1\le i\le n} X_i = k) \le P(\exists i \text{s.t.} X_i = k)$$

$$\le P(X_1 = k) + \cdots + P(X_n = k) = nQ_k.$$

c.

$$Q_k = (\frac{1}{n})^k(1-\frac{1}{n})^{n-k}\binom{n}{k} < \frac{n!}{n^k k!\,(n-k)!} < \frac{1}{k!} < \frac{e^k}{k^k}.$$

d. We assume $n \ge 3$. Since part c. holds for any k, it holds for k_0. Thus, it suffices to show that $\frac{e^{k_0}}{k_0^{k_0}} < \frac{1}{n^3}$. Taking logarithms of both sides gives

$$3 < \frac{k_0}{\lg n}(\lg k_0 - \lg e) = c(1 + \frac{\lg c - \lg e}{\lg\lg n} - \frac{\lg\lg\lg n}{\lg\lg n}).$$

If we let $x = (1 + \frac{\lg c - \lg e}{\lg\lg n} - \frac{\lg\lg\lg n}{\lg\lg n})$, then we have $\lim_{n\to\infty} x = 1$, so there is n_0 such that $x \ge \frac{1}{2}$ for all $n \ge n_0$. Therefore, any constant $c > 6$ works for $n \ge n_0$. To handle values of n such that $3 \le n < n_0$, since there are only finite number of such n, we can evaluate each expression x and determine a value c_0 for which $3 < cx$ for all values of such n. The final value of c would be $\max(6, c_0)$. Therefore we see $Q_{k_0} < \frac{1}{n^3}$.
To see that $P_k < \frac{1}{n^2}$ for $k \ge k_0$, since $P_k \le nQ_k$, $P_{k_0} \le nQ_{k_0} < \frac{1}{n^2}$. For $k > k_0$, let c be large enough that $k_0 > 3 > e$, then $\frac{e}{k} < 1$ for all $k \ge k_0$, so $\frac{e^k}{k^k}$ decreases as k increases. Then for all $k \ge k_0$, $Q_k < \frac{e^k}{k^k} \le \frac{e^{k_0}}{k^{k_0}} < \frac{1}{n^3}$, therefore $P_k < \frac{1}{n^2}$.
e.

$$\mathbb{E}[M] = \sum_{k=0}^{n} kP(M=k) = \sum_{k=0}^{k_0} kP(M=k) + \sum_{k=k_0+1}^{n} kP(M=k)$$

$$\le \sum_{k=0}^{k_0} k_0 P(M=k) + \sum_{k=k_0+1}^{n} nP(M=k) = k_0 P(M \le k_0) + nP(M > k_0)$$

$$\le k_0 \cdot 1 + n\sum_{k=k_0+1}^{n} P_k < k_0 + n \cdot n \cdot \frac{1}{n^2} = k_0 + 1 = O(\frac{\lg n}{\lg\lg n}).$$

11-3.
a. The offset of i-th probe is $\frac{i(i+1)}{2} = \frac{i^2+i}{2}$. Therefore, the probe sequence is

$$h'(k,i) = (h(k) + \frac{1}{2}i + \frac{1}{2}i^2) \mod m,$$

which is a special case of quadratic probing.
b. It suffices to show that for the worst case, each of the first m probes hashes to distinct values. Assume that there is a key k and probe numbers $i < j$ for which $h'(k,i) = h'(k,j)$. Then $\frac{i(i+1)}{2} \equiv \frac{j(j+1)}{2} \mod m$, which is equivalent to $\frac{(j-i)(j+i+1)}{2} \equiv 0 \mod m$. Since m is a power of 2, $(j-i)(j+i+1)$ is a multiple of a power of 2. But since the parity of $j-i$ and $j+i+1$ must be different, $2m$ must divide one of the them. Since $j-i < m$, 2m cannot divide $j-i$, but $j+i+1 < 2m$, so 2m cannot divide $j+i+1$ as well, a contradiction.

11-4.
a. For each pair of distinct keys $k, l \in U$, the number of hash functions $h \in \mathcal{H}$ for which $h(k) = h(l)$ is $\frac{m}{m^2}|\mathcal{H}| = \frac{1}{m}|\mathcal{H}|$, therefore the family is universal.
b. For $x = (0, \cdots 0)$, all hash functions in $\mathcal{H}$ produce the same value $h_a(x) = 0$, so $(h_a(x), h_a(y)) = (0, h_a(y))$, Therefore $\mathcal{H}$ cannot be 2-universal, because not all sequences $(h_a(x), h_a(y))$ are equally likely to occur in $\mathbb{Z}_p \times \mathbb{Z}_p$. $\mathcal{H}$ is still universal, because if we choose a random hash function from $\mathcal{H}$, the probability that two distinct keys x and y collide is the probability such that $\sum_{j=0}^{n-1} a_j(y_j - x_j) \equiv 0 \mod p$. Since x and y are distinct, there is at least one nonzero element in $y_j - x_j$. Since p is prime, the resulting product $a_j(y_j - x_j)$ is equally likely to be any value of $\mathbb{Z}_p$. This holds for any nonzero entry in $y_j - x_j$, hence $\mathcal{H}$ is universal.
c. Let $x, y \in U$ be fixed, distinct keys. We have $x_i \neq y_i$ for some i. Without loss of generality, assume $i = 0$. Then we have $h'_{ab}(x) - h'_{ab}(y) \equiv a_0(x_0 - y_0) + \sum_{j=1}^{n-1} a_j(x_j - y_j) \mod p$, which is equivalent to $a_0 \equiv (x_0 - y_0)^{-1}(h'_{ab}(x) - h'_{ab}(y) - \sum_{j=1}^{n-1} a_j(x_j - y_j)) \mod p$. Since p is prime, $(x_0 - y_0)^{-1}$ exists and unique. Therefore we can make $h'_{ab}(x) - h'_{ab}(y)$ whatever we want by choosing a particular a_0, having done so, we can also make $h'_{ab}(x)$ whatever we want by choosing a particular b. This applies for every i such that $x_i \neq y_i$.
Therefore, for any given a, we may find a hash function h'_{ab} which generates any possible element in $\mathbb{Z}_p \times \mathbb{Z}_p$ by choosing the right a_0 and b. Since there

are p^2 possible choices for a_0 and b, and also p^2 possible choices for $\mathbb{Z}_p \times \mathbb{Z}_p$, each element in $\mathbb{Z}_p \times \mathbb{Z}_p$ is generated by exactly one choice of a_0 and b. Since this is true for all p^{n-1} choices of $a_1, \cdots, a_{n-1}$, there are exactly p^{n-1} functions h'_{ab} which generate each value pair in $\mathbb{Z}_p \times \mathbb{Z}_p$, and all value pairs in $\mathbb{Z}_p \times \mathbb{Z}_p$ are then equally likely when h'_{ab} is chosen randomly, so $\mathcal{H}'$ is 2-universal.
d. Since $\mathcal{H}$ is 2-universal, for any key pair (m, m'), all hash value pairs $(h(m), h(m'))$ are equally likely when $h \in \mathcal{H}$ is chosen at random. So even if the adversary knows $\mathcal{H}$, knowing (m, t) gives no information about $h(m')$. Since the probabilities for all possible $h(m')$s have probability $\frac{1}{p}$ equally likely, the adversary cannot do better than just guessing.