SOFTENG 250

HWK7

William Shin 884819645

1. There are two situations that can arise when element *a* is added to the array.

Case 1: index of *a* < T.length, which means *a* can be added in constant time, where array T does not have to grow in size. Each insertion takes takes constant time. If our initial T.length is greater than *m*, then we would have *m* operations that take constant time and so our running time would be O(*m*).

Case 2: index of a ≥ T.length, which means array T has to double in size, creating T'. When all elements of T is placed contiguously at the start of T', T.length constant operations when copying elements into T'. This takes O(T.length) time. T.length is bounded by m, so O(T.length) = O(*m*).

So if the total running of these operations is O(*m*) and there are *m* insertions, each insertion on average takes O(1) time.

2.

Chaining

h(x)	0	1	2	3	4	5	6	7	8
x		177					198		

h(x)	9	10	11	12	13	14	15
	9 57 201 89		43 59	204		142 158	15

Linear Probing

h(x)	0	1	2	3	4	5	6	7	8
x	89	15	177	59			198		

h(x)	9	10	11	12	13	14	15
	9	57	43	204	201	142	158

Double hashing

h(x)	0	1	2	3	4	5	6	7	8
x	177	158		89		15	198	59	

h(x)	9	10	11	12	13	14	15
	9		43	204	201	142	57

3.

Although calculating the hash value of a substring should take O(*L*) time, we can assume that it takes constant time by using the previous hash value when calculating the hash value of the next substring. So calculating the hash value of the first substring will take O(*L*) time and calculating all subsequent substrings will take O(1) (constant) time. There are n substrings so the total running time for this operation is O(*L*+n).

Calculating the hash value of the pattern will take O(*L*) time, where *L*, is the length of the pattern.

Comparing the hash values of 1 substring to the hash value of the pattern takes constant time. There are n comparisons so the total running time for this operation is O(n).

When the hash values are equal, checking individual characters of the pattern to the substring takes O(*L*) times. This operation is carried out a minimal number of times if a good hashing function is used. This operation will result in the substring being returned if it matches the pattern or being refuted if it doesn't match the pattern.

So the total running time of this algorithm is:

T(n) = T(calculating h(pattern)) + T(calculating n * h(substring)) + T(comparing hash value of pattern to hash value of substring) + T(comparing the individual characters of substring to pattern, returning it if true or rejecting it if false)

= O(*L*) + O(n+*L*) + O(n) + O(*L*) + time refuting false matches

= O(n+*L*) + time refuting false matches.

4.

Let event B = "train arriving at Britomart on time"

Let event M = "train departing Manukau on time"

We know that P(B) = 0.8, P(M) = 0.9 and P(B ∩ M) = 0.75

a) P(B | M) = $\frac{\mathrm{P(B \cap M)}}{P(M)} = \frac{0.75}{0.9} = 0.833$

b) $P(M \mid B) = \frac{P(M \cap \mathrm{B})}{P(B)} = \frac{0.75}{0.8} = 0.9375$

c) Not independent because:

$P(B|M) \neq P(B)$ and $P(M|B) \neq P(M)$ and $P(B) * P(M) = 0.72 \neq P(\mathrm{B} \cap \mathrm{M})$