7.1.1)

It is possible to swap the values of u and v without using extra storage. The following steps are required:

U = U + V

V = U - V

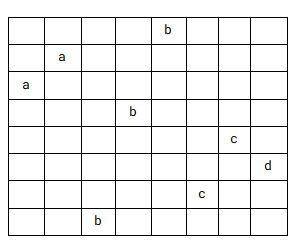
U = U - V

This swaps u and v without using extra storage.

7.1.3)

In order to sort the set { b, c, d, c, b, a, a, b } in alphabetical order, do the following steps in the distribution counting algorithm.

|  | **A** | **B** | **C** | **D** |
| --- | --- | --- | --- | --- |
| **Frequencies** | 2 | 3 | 2 | 1 |
| **Distribution values** | 2 | 5 | 7 | 8 |



| b | 2 | **5** | 7 | 8 |
| --- | --- | --- | --- | --- |
| a | **2** | 4 | 7 | 8 |
| a | **1** | 4 | 7 | 8 |
| b | 0 | **4** | 7 | 8 |
| c | 0 | 3 | **7** | 8 |
| d | 0 | 3 | 6 | **8** |
| c | 0 | 3 | **6** | 7 |
| b | 0 | **3** | 5 | 7 |

7.1.8)

For each of the statues, count the number of statues that are shorter than it through counting sort. Use the resulting values to sort the status to their target positions. For the first statue that is not in the correct position, swap it to the target position. Repeat this until there are no statues left in a spot that is not their target position. With all statues moved to their respective positions, the use of the algorithm minimized the total number of moves required to get the statues in place.

7.2.1)

| B | A | O | ... |
| --- | --- | --- | --- |
| 2 | 1 | 3 | 6 |

BESS\_KNEW\_ABOUT\_BAOBABS

BAOBAB K not in table shift 6

BAOBAB A in table shift 1

BAOBAB O in table shift 3

BAOBAB Match

7.2.3

1000 zeros

a) 00001

| 0 | 1 |
| --- | --- |
| 1 | 5 |

1 shifting comparison until at end

996 comparisons

b) 10000

| 0 | 1 |
| --- | --- |
| 1 | 4 |

4 useless comparisons then 1 shifting comparison

5 \* 996 = 4980

c) 01010

| 0 | 1 |
| --- | --- |
| 2 | 1 |

1 useless comparison then 1 shifting comparison but shifts 2

996 \* 2 / 2 = 996

7.2.7)

a) 00001

| 0 | 1 |
| --- | --- |
| 1 | 5 |

| 1 | 00001 | 5 |
| --- | --- | --- |
| 2 | 00001 | 5 |
| 3 | 00001 | 5 |
| 4 | 00001 | 5 |

Will only make bad comparisons

996

b) 10000

| 0 | 1 |
| --- | --- |
| 1 | 4 |

| 1 | 10000 | 3 |
| --- | --- | --- |
| 2 | 10000 | 2 |
| 3 | 10000 | 1 |
| 4 | 10000 | 5 |

Will make four good comparisons then one bad and shift five

1000 \* 5 / 5 = 1000

c) 01010

| 0 | 1 |
| --- | --- |
| 2 | 1 |

| 1 | 01010 | 4 |
| --- | --- | --- |
| 2 | 01010 | 4 |
| 3 | 01010 | 2 |
| 4 | 01010 | 2 |

Will make one good comparison then one bad and shift 4

996 \* 2 / 4 = 498

7.2.9)

a)

Through horspool’s algorithm, the remaining m-1 characters can be compared from left to right, since the shift is the pattern is based solely on the text’s alignment with the last character of the required pattern.

b)

The Boyer-Moore algorithm can be used to compare the remaining m - 1 characters in the pattern from right to left because of the shift table.

7.3.1)

A)

Keys: 30, 20, 56, 75, 31, 19

h(K): K mod 11

Addresses:

| k | 30 | 20 | 56 | 75 | 31 | 19 |
| --- | --- | --- | --- | --- | --- | --- |
| h(K) | 8 | 9 | 1 | 9 | 9 | 8 |

Open table:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| null | 56 | null | null | null | null | null | null | 30 -> 19 | 20 -> 75 -> 31 | null |

B)

The maximum number of comparisons on the key in a successful search of this table should be 3, when looking for a key of 31

C)

By assuming that all keys are equally likely and using the average number of comparisons, the result is roughly 1.7, which was calculated by.

⅙ + ⅙ + ⅙ + 2/6 + 3/6 + 2/6 = ~1.7

7.3.2

A)

Keys: 30, 20, 56, 75, 31, 19

h(K): K mod 11

Addresses:

| k | 30 | 20 | 56 | 75 | 31 | 19 |
| --- | --- | --- | --- | --- | --- | --- |
| h(K) | 8 | 9 | 1 | 9 | 9 | 8 |

Open table:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 56 | 19 | null | null | null | null | null | 30 | 20 | 75 |

B) The maximum number of comparisons on the key in a successful search of this table is 6, when looking for K = 19

C) By assuming that all keys are equally likely and using the average number of comparisons, the result is roughly 2.3 which was calculated by:

7.3.3)

The variation in the number of values from a function like that would be limited solely by the size of the alphabet. Usually there is not an equal probability of a word starting with any letter.

7.3.7)

FIrst insert a successive number of elements from the list into a hash table until a matching element is found or the list is emptied. The worst case efficiency for this should be theta of n^2 since all distinct keys would be hashed to the same address allowing for the number of comparisons to be in the domain of theta n^2. The average case would be theta of n when all keys are distributed evenly.

7.4.1) A dictionary with the first and last word on the page labeled. Example: (A - As) this idex contain all words between a and as

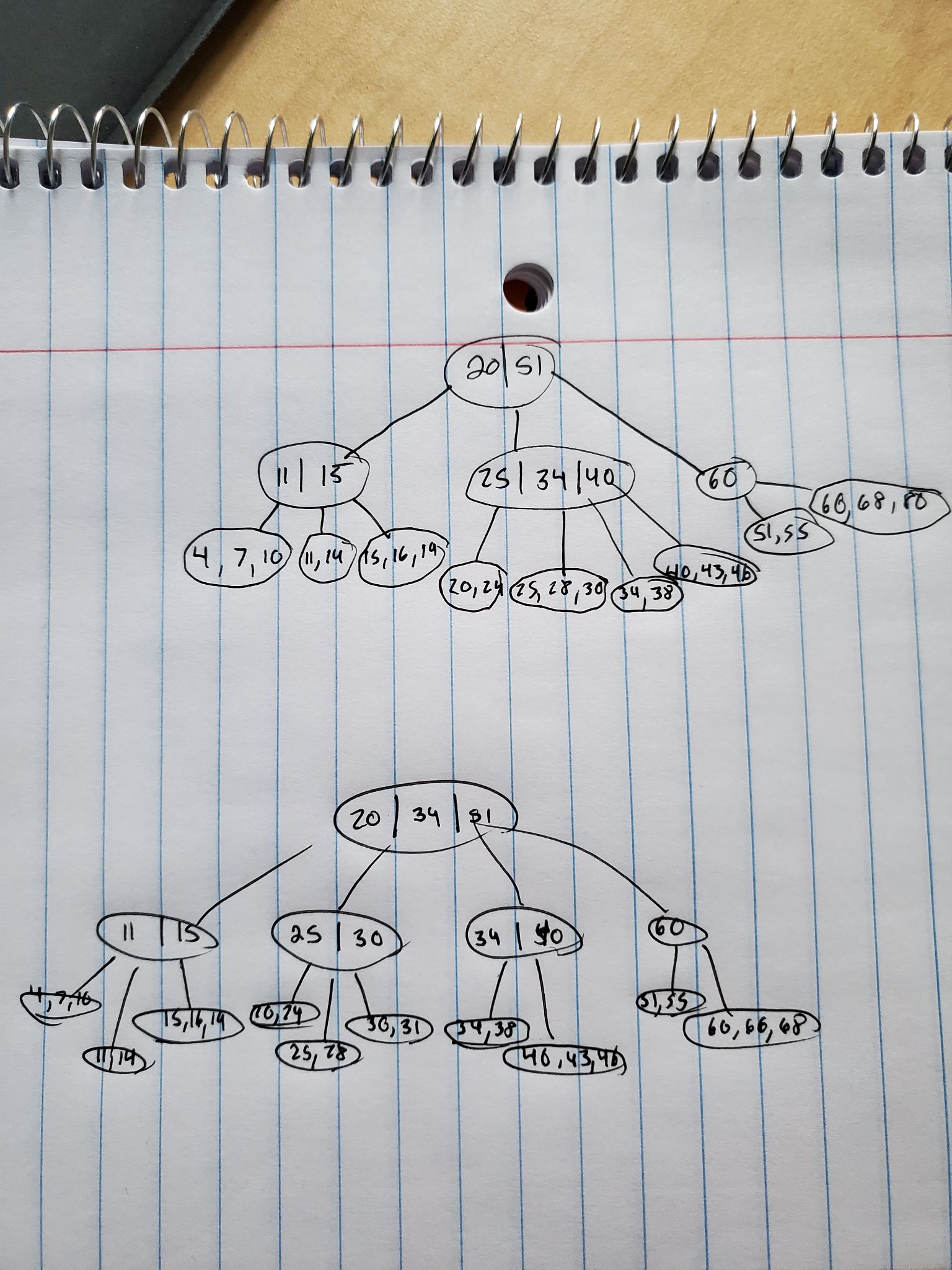
7.4.3)

log(m/2) + 1 3

The smallest values that satisfies this is 585

7.4.4)

Insert 30 into 7.8



Then insert 31

