Assignment 4

Sullivan Frazier

December 1, 2021

1 Question 1

The classic sequential search looks for a given pattern, and once it is found it is stopped. While on the other hand, the non-classic search returns the number of occurrences of a key/pattern within a given text. This has a time efficiency of O(m log n) due to the algorithm needing to repeatedly evaluate the text at different positions throughout. The classic search has a time efficiency of O(nm) due to it only reading through the text until the search pattern is found.

2 Question 2 - Part A

There is a chance you will select two gloves of the same color. Best Case:2

Worst case implies the selection of wrong gloves every draw until there is no option but a solution. Although we cannot see the gloves, we can tell which one is left handed versus right handed. The worst case entails pulling 5 left-handed red gloves, 4 left-handed yellow gloves, and 2 left-handed pairs of green gloves. Then your next draw would be a matching pair. So the worst case means you would have 5+4+2+1=12 total draws from the drawer.

3 Question 2 - Part B

Best case versus worst case. There is a 3:1 ratio, the odds of the missing socks being from two different pairs are high. In retrospect, the odds of the socks that are missing and matching are low. The number of pairs of socks on average would be three due to it's probability leaning towards the missing socks being mismatching.

4 Question 3

In order to split the pattern like requested -
for i = 0; i ¡ n;++i **do**
    for j = 0; j ¡ n*2-1;++j **do**
        $Left = disk[j]//setleft$

```
        Right = disk[j + 1]//setright
        if Left != Right then
            disk[j] = Right
            disk[j + 1] = Left
            ++count
        end if
    end for
end for
```

5 Question 4

Brute force is unfortunately rather lengthy and inefficient. The time efficiency is O(m*n) due to it having to compare each letter of the pattern, with each letter of the text. So if the length of the text is 47, and the pattern is 6 letters, (47*6)=282 comparisons.

6 Question 5

The first move will be the two boys taking the boat to the other side, one boy stays on the island and the other returns with the boat. Then a soldier comes across and stays with the first boy. These four trips designate the amount of total trips required while also meeting the problems requirements. The one on the boat ferries across the n amount of soldiers, allowing for 4n trips to be made.

7 Question 6

An efficient manner in my opinion, is to clean up and/or sort the given data, and then compare. This includes removing punctuation, spaces, commas, and other sorts of characters. Once you have the two strings cleaned up and sorted, just compare the values of the letters. Or the prime number trick (come back and look at this)

8 Question 7

Ideally, and in most (theoretically all) situations, horspool will always make LESS comparisons than brute force, due to it's shift table attribute. This allows for an increase of efficiency all around. This algorithm skips comparisons that are "known" to be mismatches, preventing wasted computational power and time.

9 Question 8

If horspool finds a matching substring, the shift would be determined by the shift table. If the entirety of the pattern matches in a portion of the text, the shift will be the size of the pattern.

10 Question 9 - Part A

Input	Hash(mod 11)
30	8
20	9
56	1
75	9
31	9
19	8

11 Question 9 - Part B

Having the load factor further away from 1, implies an inefficiency in the table (wasted space), while on the other hand having it too long would mean longer lists. If the load factor is close to 1, the hash table is fully optimized, making it possible to find a key with a cost of 1-2 comparisons.

12 Question 9 - Part C

The efficiency of operations such as insertion and deletion are identical to that of searching, on average they are O(1) if the number of keys is about equivalent to the hash table's size.

13 Question 10

Algorithm 1 greedyAssignment(A[n][n])

```
selected[n]
for i = 0; i < n; i++ do
    selected[i] = 0
end for
assignments[n]
for i = 0; i < n; i++ do
    assignments[i] = −1
end for
for i = 1; i < n; i++ do
    minCost = INTMAX, selectedJob = −1
    for j = 1; j < n; j++ do
        if A[i][j] < minCostandselected[j] == 0 then
            selected[j] = j
            minCost = A[i][j]
        end if
        selected[selectedJob[j]] = 1
        assignments[i] = selectedJob
    end for
end for
returnassignments
```

This algorithm does not yield an optimal solution, although there is a chance it may. This is only a feasible solution to get you in the ballpark of what is optimal.