Mert Kilic

100334228

Lab 2

CPSC 2150

1)

A picture containing text, clock

Description automatically generated

N2

N

Text, letter

Description automatically generated

N3

N2

N

Text

Description automatically generated

N(N-1) / 2

Time Complexity = N2 + N3 + N2 / 2 - N / 2 = O(N3)

Since there are no simultaneous calls or array initialization it the stack, space complexity is O(1).

2)

a) Time Complexity = 1 + 2 + 4 + …. + N / 4 + N / 2 + N = 2N -1 f(n) = 2n -1

Space Complexity = 1

b) Time Complexity = 1 + 2+ …. + n – 2 + n-1 + n = N(N+1) / 2 T(n) = n(n+1) / 2

Space Complexity = 1

c) Time Complexity = n\*log2n T(n) = nlog2n

Space Complexity = 1

3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 Second | 1 Hour | 1 Month | 1 Century |
|  | ≈ 10300000 | 101,080,000,000 | 10788,400,000,000 | 10946,080,000,000,000 |
|  | 90,000,000,000 | 11664 \* 1014 | 788,400,000,0002 | 946,080,000,000,0002 |
|  | 300000 | 1,080,000,000 | 788,400,000,000 | 946,080,000,000,000 |
|  | 62550 | 133,000,000 | 72,500,000,000 | 68,500,000,000,000 |
|  | 548 | 32863 | 887919 | 30758413 |
|  | 67 | 1026 | 9238 | 98169 |
|  | 18 | 30 | 39 | 50 |
| n! | 9 | 12 | 15 | 17 |

4)

a) My algorithm uses 3 helper dynamic arrays to store indexes of colors from the actual array. The time complexity of this part is O(N) and space complexity is O(M + P + K). Then, it starts sorting reds by comparing array of red indexes to array of blue indexes and white indexes. This operation is performed in O(N) time. Lastly, since all the whites are supposed to be at the end, it checks the array in reverse order all the way up to size – numOfWhites. While checking, if the current index does not contain white, then it goes back to the array of white indexes and performs a swap from there. This operation is again, performed in O(N) time. To conclude, the program sorts the array in O(N) time by using helper dynamic arrays.

Time Complexity = O(N)

Space Complexity = O(M + P + K)