CS325

Guangyu Zhang

1.

a).

Solve this problem in divide and conquer way can follow the steps: first, we divide the whole array into 2 parts as subproblems, and recursively divide the array. Then, conquer, for each subproblem, find its max or min value. Finally, combination, compare the results of each subproblem to find the max or min.

Pseudo-code:

Find max\_or\_min(array[n])

{

if (len(array[n])) = 1

return array[0]

else

divide\_to\_2(array[n])

sub\_a=array[0, n/2]

sub\_b=array[n/2, n]

Find max\_or\_min(sub\_a)

Find max\_or\_min(sub\_b)

Compare (Find max\_or\_min(sub\_a), Find max\_or\_min(sub\_b))

return max, min

}

b).

c).

The iterative algorithm should be:

For each number, compare to n – 1 others numbers, so it has 2 for loops.

Therefore, the recursive min\_and\_max algorithm has less time consumption than iterative algorithm.

2.

a).

Mergesort3(array[n])

{

m = n/3

array\_a = array[0, m]

array\_b = array[m, 2m]

array\_c = array[2m, n]

Mergesort3(array\_a)

Mergesort3(array\_b)

Mergesort3(array\_c)

}

Merge3()

for (i=0; i < m; i++){

result[] = array[i]

}

for (i=0; i < 2m-m; i++){

result[] = array[i]

}

for (i=0; i < n; i++){

result[] = array[i]

}

b).

c).

3. submitted as code

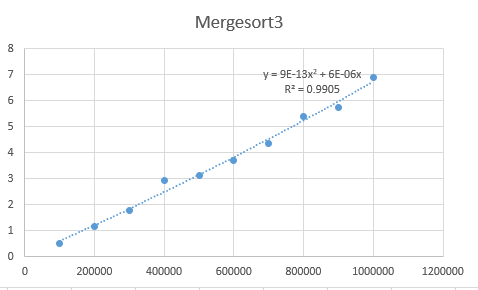
4.

a). submitted as code

b).

|  |  |
| --- | --- |
| Number(n) | merge sort 3 |
| 100000 | 0.51265549659729 |
| 200000 | 1.1659111976623535 |
| 300000 | 1.7732818126678467 |
| 400000 | 2.9261486530303955 |
| 500000 | 3.12766432762146 |
| 600000 | 3.706111192703247 |
| 700000 | 4.340397119522095 |
| 800000 | 5.410558223724365 |
| 900000 | 5.754785060882568 |
| 1000000 | 6.912974834442139 |

c).



d).

3-way merge sort runs a little faster than 2-way merge sort, but generally they have similar time consumption. The theoretical running times of 3-way and 2-way merge sort are same: . However, in my experiment, 3-way merge sort is a little faster than 2-way merge sort. One possibility is that my part causes this difference.

