@SE221 (Lab explanation)

ID: 21301289

Name: Ishtiaq Ahme

see: 23 (lab)

#Task-1: 21301289

1. The "merge" function takes two sorted arrays, "a" and "b", and merge them into a single sorted array "c" using a two-pointer approach.

2. The "mergeSort" function implements the merge sort algorithm. It recursively divides the input array "arr" into two halves until the base case is reached. It them merges the sorted halves back together using the "merge" function.

3. The code open two files, "input1.txt" for reading ("r") and "output1.txt" for writing ("w"). It reads an integer "n" from the input file, which represents the number of elements in the array, and then reads the space-separated array elements into the list "arr".

4. The "mergeSort" function is called to sort the array "arr", resulting in "sorted_arr".

5. The sorted array "sorted_arr" is converted to a space-separated string and written to the output file "output1.txt" using the "write" method.

6. The "close" method using for the close the input and output.

#Task-2: 21301289

1. The "find_max" function is a recursive function that uses a divide and conquer approach to find the maximum value within a given array. It takes an array "arr", a start index "start", and an end index "end". If "start" is equal to "end", it means there is only one element in the subarray, and it returns that element as the maximum. Otherwise, it divides the subarray into two halves, recursively finds the maximum value in the left and right halves and returns the maximum of these two values.

2. The code opens two files, "input2.txt" for reading and "output2.txt" for writting.

3. It reads the first line of "input2.txt" to get the value of "n", which represent the number of elements in the array. Then, it reads the second line of "input2.txt" to get the space separated list of integers, which is stored in the "arr" list.

4. The code calls the "find_max" function to find the maximum value in the "arr" list. The maximum value is stored in the variable "result".

5. The maximum value is written to "output2.txt" using the write method.

6. finally print(".") use for handleing return functi

7. the "close" method using for the close input and output.

Task-3

21301289

1. "merge_and_count" function:

This function takes two sorted arrays, "left" and "right", and returns a merged list along with the count of inversions. It combines the two array into one while counting the inversions. Inversions are counted whenever an element in the "right" list is smaller than an element in the "left" array.

2. "merge_sort_and_count" function; This function recursively sorts and counts inversions in an input list "arr". It divides the list into two halves, sort each half, and then use the "merge_and_count" function to merge and count inversions in the combined array.

3. The code opens two files, "input3.txt" for reading and "output3.txt" for writting.

4. input, it reads the first line of "input3.txt" to get the value of "n", which represents the number of elements in the array. Then it reads the second line of "input3.txt" to get a space separated list of integers, which is stored in the "heights" array.

8
27913684

5. The code call "merge_sort_and_count" functi to find the number of inversions in the "heigh array. The count of inversions is stored in the variable "inversions".

6. Writes the ~~code~~ count inversions to "output3.tx using the write method.

7. Finally print("") are for handleing the return function.

8. The "close" method using for the close input and output.

Sample Input 3 Explanation:

In the ~~above, co~~ code, we are taking input in the format ~~description~~described described in the problem statement. The second line of input contain N integers representing the height of the aliens.

for the 3rd input.: 8
2 7 4 1 3 6 8 3

where 8 is the number of aliens. and 2 7 4 1 3 6 8 3 is the permutation of their heights.

2130128J.

Task-4:

1. "max_sum_of_squares, this function takes an array of integers "arr" as input and calculates the maximum value of A[i] + A[j]^2 by itering through all possible pairs of elements in the array.

2. n = len(arr) which calculates the length of "arr".

3. arr.sort(), The input ~~list~~ array "arr" is sorted in ascending order to facilate the efficient calculation of the maximum value.

4. max_sum, A variable is initialized with negative infinity to store the maximum value.

5. Nested loop:
   (i) The code uses nested loops to iterate through all pairs of elements in the sorted list.
   (ii) The outer loop iterates over the elements

from the beginning to the second to last element.

(iii) The inner loop iterates over the elements from the next element to the last element.

(iv) For each pair of element (i, j), it calculates the sum of the current pair's values plus the square of the second element (A[i] + A[j]^2)

(v) The maximum value is updated if the current sum is greater than the current maximum.

6. ~~for writen~~ calculate, max value is written to the "output4.txt"

7. Close method using for close input and output.

#Task 5: 21301289

1. "Partition()", This function takes an array "arr" and two indices "low" and "high" as parameters. It chooses the rightmost element as the pivot and rearranges the elements such that elements less than the pivot are on the left, and elements greater than the pivot are on the right. It returns the pivot elements.

2. "quick-sort()", This is the main quick sort function. It recursively sorts the subarray before and after the pivot element. The base case is when "low" is no longer less than "high".

3. The code opens "input5.txt" for reading. It reads the number of elements "N" from the first line and the list of integers from the second line.

4. Sorting, it applies the quick sort algo to sort the list of numbers by calling "quick sort".

5. The sorted list of number is written to "output5.txt" in space-separated format.

6. for print() function its handling for return.

7. Reader method using for the close input and output.

Task-6:

2130128 J

The target of partitions is to place the pivot at its correct position in the sorted array and put all smaller elements to the left of the pivot, and all greater elements to the right of the pivot. Partion is done recursively on each side of the pivot is placed in its correct position. Then finally it sorts the array.