**Assignment 3**

1. **What is the time complexity and space complexity of the code you wrote (question 2). Paste your code and explain them for each line in the pages/word “Assignment 3”.(3 scores).**

Problem A:

//define the properties of this class

**double** sa; //s1(n)=k

**double** hrs,hr1=36,hr2=41,hr3=48; //s2(n)=4k,t1(n)=3

**double** r1=15.0,r2=15\*1.5,r3=15\*2; //S3(n)=3k,t2(n)=3

**public** **double** employee\_Salary(**double** hrs) { //s4(n)=2k

//three different conditions according to the description

**if**(hrs<=hr1) { //t3(n)=1

**return** hrs\*r1;

}**else** **if**(hrs<=hr2) { //t4(n)=1

**return** hr1\*r1+(hrs-hr1)\*r2;

}**else** **if**(hrs<=hr3) { // t5(n)=1

**return** hr1\*r1+(hr2-hr1)\*r2+(hrs-hr2)\*r3;

}**else**

**return** -1.0;

}

So, the time complexity and space complexity of this function is:

T(n)=9≈O(1)

S(n)=10k≈O(1)

Problem B:

**public** **static** List<List<Integer>> findTri(**int**[] nums) { //s1(n)=2k

Arrays.*sort*(nums); //t1(n)=1

List<List<Integer>> list = **new** ArrayList<List<Integer>>();//s2(n)=k

**for**(**int** i = 0; i < nums.length-2; i++) { //s3(n)=k,t2(n)=nums.length-2

**if**(i > 0 && (nums[i] == nums[i-1])) **continue**;//t3(n)=3\*(nums.length-2)

**for**(**int** j = i+1, k = nums.length-1; j<k;) {

//s4(n)=k\*(nums.length-2),t4(n)=(nums.length-1)\*(nums.length-2)

**if**(nums[i] + nums[j] + nums[k] == 0) {

//t5(n)=t4(n)\*1

list.add(Arrays.*asList*(nums[i],nums[j],nums[k]));

// t6(n)=t4(n)\*1

j++; // t7(n)=t4(n)\*1

k--; // t8(n)=t4(n)\*1

**if**((j < k) && (nums[j] == nums[j-1]))j++;// t9(n)=t4(n)\*4

**if**((j < k) && (nums[k] == nums[k+1]))k--;// t10(n)=t4(n)\*4

}**else** **if**(nums[i] + nums[j] + nums[k] > 0) k--; // t11(n)=t4(n)\*2

**else** j++; // t12(n)=t4(n)\*1

}

}

**return** list;

}

For this function, the time and space complexity is:

T1(n)=15(nums.length-1)\*(nums.length-2)+4\*(nums.length-1)+1

≈O(n2)

S1(n)=k(nums.length+2)≈O(n)

**private** **static** **void** printArray(**int**[] arr) { // s1(n)=2k

System.***out***.print("[");

**for**(**int** i=0;i<arr.length-1;i++) { // s2(n)=k,t1(n)=arr.length-1

System.***out***.print(arr[i]+", ");

}

System.***out***.println(arr[arr.length-1]+"]");

}

For this function, the time and space complexity is:

T2(n)≈O(n)

S2(n)≈O(1)

**public** **static** **void** main(String[] args) { //s1(n)=2k

**int**[] arr= {-1, 0, 1, 2, -1, -4}; //s2(n)=k,t1(n)=1

System.***out***.println("Array:");

*printArray*(arr); //t2(n)=T2(n),s3(n)=S2(n)

System.***out***.println();

System.***out***.println("find triplets:");

List<List<Integer> > triplets = *findTri*(arr);//s4(n)=k\*S1(n),t3(n)=T1(n)

**if** (!triplets.isEmpty()) { //t4(n)=1

System.***out***.println(triplets);

} **else** {

System.***out***.println("No triplets can be found");

}

For the main and whole solution, the time and space complexity is:

T(n)=2+T2(n)+T1(n)≈O(n2)

S(n)=3k+S2(n)+S1(n)≈O(n)

Problem C:

**public** **static** **void** longestSub(String str) { //s1(n)=2k

**int** maxLen = 0; //s2(n)=k,t1(n)=1

**int** index = 0; //s3(n)=k,t2(n)=1

**int** j = 0; //s4(n)=k,t3(n)=1

HashMap<Character, Integer> map = **new** HashMap<Character, Integer>();

//s5(n)=k

**for** (**int** i=0; i<str.length(); i++) { //s6(n)=k,t4(n)=str.length()

map.clear(); //t5(n)=t4(n)

map.put(str.charAt(i), 1); //t6(n)=t4(n)

**for** (j=i+1; j<str.length(); j++) { //t7(n)=str.length()\*str.length()

**if** (map.get(str.charAt(j)) != **null**) {// t8(n)=t7(n)

**break**;

}

map.put(str.charAt(j), 1); // t9(n)=t7(n)

}

**if** (j-i > maxLen) { // t10(n)=t4(n)

maxLen = j-i; // t11(n)=t4(n)

index = i; // t12(n)=t4(n)

}

}

System.***out***.println(str.substring(index, index+maxLen)+","+maxLen);

// t13(n)=1

}

For this function, the time and space complexity is:

T1(n)=4+6t4(n)+3t7(n)

=3\*str.length()\*str.length()+6\*str.length()+4

≈O(n2)

S1(n)=7k≈O(1)

**public** **static** **void** main(String[] args) //s1(n)=2k

{

String str = "bbbbbbb"; //s2(n)=k,t1(n)=1;

System.***out***.println("The input string is "+str);

System.***out***.println("Substring and it's length:");

*longestSub*(str); //s3(n)=S1(n),t2(n)=T1(n)

}

So, for this function and the whole solution to this problem, the time and space complexity is:

T(n)=T1(n)+1≈ O(n2)

S(n)=3k+S1(n)≈O(1)