Hw2_Report_120090272_Feng_Chuqiao

Question 1

1.The possible solutions for this problem

To find the indices of two elements such that add up to t, the possible solutions can be various.

1: traverse the input array one by one, compare one data with all other data in the input array. However the time complexity of this method is $O(n^2)$.

2: using a dictionary to record the data, traverse the number array, use the sum as the key of the dictionary. Check if the key is already in dictionary, so that we can get the value of the key. If there isn't, take the position of the number as the value of the dictionary, and store the set of data in the dictionary. The time complexity of this method is $O(n)$.

2.Solution of this problem

In mu solution, I first create a set of new number list and the original input number set, because the numbers in set cannot be the same. If there are two same numbers valued $\frac{t}{2}$ then the program return directly. Next, I use the numbers i in the input array with $i - \frac{t}{2}$ and adding into the new set. If the number already in number list, then set the value as the target number, and compare with the index in the number list, then the problem can be solved very quickly.

3.Why the solution is better

Because this solution can be with low time complexity, and easy to understand and completed, so this solution must be better for the question.

Question 2

1.The possible solutions for this problem

1: Insertion sort, with should tranverse the input data array, and compare with the previous numbers, and find a correct space to insert. This solution has a time complexity of $O(n^2)$.

2: Merge sort, which follows a process of divide and conquer, finally combine the information of divided list. First we divide the array into the half by half, with a final state that each list contains 2 numbers. We compare these lists with 2 numbers, and count if there are any inversions. Then we combine 2 number lists into 4 number lists, and so on, and compare them one by one to count the inversions of the total number list. This solution only has a time complexity of $O(nlogn)$.

2.Solution of this problem

In this question, we choose merge sort as one of the best solutions.

3.Why the solution is better

Because the solution divide the whole process into different easy part, which finally reduce the time complexity of the algorithm.

Question 3

1.The possible solutions for this problem

The possible good solution for this problem is using doubly linked list. So that we can easily insert or pop out the numbers at any position in the array. As for the instructions of this question, we should locate the position of the input number and then insert in. If we strictly tranverse through the linked-list, the time complexity will be very large.

2.Solution of this problem

In this solution, I use a dictionary to store the position and corresponding value of node, so that is will be easier for program to locate the number. In doubly-linked list method, the time comlexity for this problem is $O(1)$。

3.Why the solution is better

Because doubly linked list can help us quickly insert the number, and locate the position, so the time complexity is very small.