CSE 321 Homework 5 Report

Question 1

In the function I get the two input parameter calls city1 and city2 that are integer arrays. Then I initialize the 2 array PN(length=city1) for city1 PS(length=city2) for city2. After that I assign the first element of PN,PS to city1,city2's first elements. Then in the for loop(in range length of city) I assign the PN and PS element to minimum of the costs with changing cost like this PS[i]=city2[i]+min(PS[i-1],10+PN[i-1]). Finally I return the carried minimum of the last item in the PS,PN array.

Complexity Analysis:

Attention those **n** is city1 or city2's length and I ignore the complexity of min because of the constant 2 element size.

2*O(n) → comes from PN and PS creation.

O(n)→ comes from process loop

So the total complexity is $\boldsymbol{W(n) = 3(O(n)) = O(n)}$.

Question 2

In the function I get only one 2d array for includes start times and finish time for symposiums. Then I always select the first activity for apply the greedy concept. After that for the remain symposiums: I choose the symposium if the symposiums start time greater or equal the previous symposiums finish time. Thus I add the list of these, valid symposium by using indexes.

Complexity Analysis:

Attention that **n** is number of symposiums

The control loop's length is number of symposiums

So the total complexity is $\boldsymbol{W(n) = O(n)}$.

Question 3

In this question I use the kind of different function. My function finds the total number of 0 sum's subset by using dynamic programming. And then if the result is greater then 1 returns true(the key value is 1 because my function counts the empty subset)

The recurrence relation becomes dp[i][s] = dp[i+1][s+arr[i]] + dp[i+1][s]

The recurrence turns n times(total number of elements in the given list) and because of the continuously calculate subset sum then the complexity is **O(n*s)** s is sum of array

Question 4

In this problem I find the maximum cost by using mismatch=-1 gap=-1 and match=2(it can be changed in function parameter). For the solve this problem derive the basically this recurrence relation:

dp[i][j] = max(dp[i-1][j-1]+mismatch,dp[i-1][j]+gap,dp[i][j-1]+gap)

For apply this problem's solution Firstly I construct the 2d matrix by using gap values. After that I fill the 2d array to values. And then I construct the string that are matching. Then calculate the value of cost. Finally I print it these strings.

In the algorithm I use dp as 2d m*n array that states us the complexity **O(m*n)** because in the algorithm the loops are traverse whole 2d array.

Question 5

In this algorithm I get the numbers list as parameter to add operation. Firstly, my adding mechanism is like this:

Assume the numbers are these: **[10,20,40,50]**

Then the number of operation calculates like this:

10+20=**30 (30 is both subtotal and operation count in this add)**
30+40=**70 (70 is both subtotal and operation count in this add)**
70+50=**120 (120 is both total in all of adds and operation count in this add)**

Thus the total operation count is sum of subtotals: **30+70+120 = 220**

I apply the min for the greedy acceptance rule. For implementing this mechanism I hold two variables call opInOneStage and totalOperations in a while loop that has length of the number's list. Thus All of stages in while loop I add the 2 of numbers that selecting and deleting the minimum number in each cycle, in list and save the opInOneStage and on the other hand I add the totalOperations and opInOneStage. So at the end of the loop, my variable totalOperations became the desired result.

Complexity Analysis:

Attention that **n** is length of number array.

The while loop's complexity is: **O(n)** →because all of element should be deleted in the end.

Min() function in while loop is: **O(n)** → Although the number of elements decreases as the loop passes, it does not change the total result.

So the total complexity is $\boldsymbol{W(n) = O(n\text{^}2)}$.