Operating Systems and Algorithms

Asssignment-2

Name:Ajay Ray Roll:2021102032

Q.1. Explanation: Given an array of integers . if we see the the problem closely here we need to find two parts of the subarray such that the differece of the two parts is minimum.

algorithm: every element has two choices either it will be part of s1 or s2 . so accordingly i will cal the recursive call and try to minimise the sum and total_sum - sum.

```
This is the recursive call
    int find(vector<int>&A,vector<vector<int>>&dp,int ind,int sum,int tsum)

and two options
int take=find(A,dp,ind+1,sum + A[ind],tsum);
  int n_take=find(A,dp,ind+1,sum ,tsum);
```

Time complexcity: O(N*total_sum) . Basically I run two nested for loop that is defining the time complexcity.

Space complexcity:O(N*total_sum) . this is the total space for the size of dp matrix.

Q.2. This is a classical knapsack problem. implementation is just take the item or not and according try all the combination to maximize the values.

Algorithm: I implementated usinng dynamic programming where dp[i][j] basically denoting the maximum value obtained up to the i th index of the array with j weight.

```
this is the main working of the code anc according to it the dp states are updated.
if(wieght[i-1]<=j){
          dp[i][j]=max(value[i-1]+dp[i-1][j-wieght[i-1]],dp[i-1][j]);
        }
else{
          dp[i][j]=dp[i-1][j];

        }
```

Time complexcity: O(N*wight) . Basically I run two nested for loop that is defining the time complexcity.

Space complexcity:O(N*total_sum) . this is the total space for the size of dp matrix.

Q.4.

Explanation: Here we need to find minimum wastage of area . wastage is defined as any rectanguler piece which is not listed.

Algorithm: I used a recursion with memorization algoritihm to solve the question. for a given w and h i am initilizing the dp array with maximum wastage that is w*h . Then for all the valued of w and h i am calling the recursive function and update the dp state whenever a less wastage is obtained.

dp state: dp[w][h] what is the minimum wasteage with rectangle of width w and hieght h.

```
main algorithm:
for (int i = 1; i <= h / 2; i++) {
        temp = find(w, i);
        if (temp + find(w, h - i) < dp[w][h]) {
            dp[w][h]= temp + find(w, h - i);
        }
    }
    for (int i = 1; i <= w / 2; i++) {
        temp = find(i, h);
        if (temp + find(w - i, h) < dp[w][h]) {
            dp[w][h] = temp + find(w - i, h);
```

```
        }
    }
```

Time complexcity: O(w*h) . maximum recursive call will be w* h times with using the memorization for subproblems.

space complexcity is O(w*h) storing the value in dp state.