Date:

Sun Mon Tue Wed Thu Fri Sat

ASSIGNMENT # 03
DESIGN AND ANALYSIS OF ALGORITHMS : CS 302

Name of Group members:

Syed M. Hammad 18K-0237

Hiba Zubair 18K-1361

Section: F

Due Date: 16th Dec 2020

Group question allocated: 9

PROBLEM STATEMENT: Using dynamic programming understand and solve partition-problem

$S = \{3, 1, 1, 2, 2, 1\}$

- Identifying and explaining the problem statement further:

  Divide the given array into two subset that the sum of both subsets is equal.

  ↳ now the partition can be of any size

  ↳ furthermore elements in subset must be unique i.e the element in one subset shouldn't belong to the other subset as well.

Date:

Sun Mon Tue Wed Thu Fri Sat

↳ This is basically an extension of subset sum problem in which the sum is already given and we need to partition the array into subset such that both subset equal to that sum.

——• Similarity of this problem with knapsack problem solution:

If we recall what knapsack was, we were given an item array which was divided into weight array and value array. And a weight was given which was equal to the weight of the knapsack.

Here we would say that the given array is weight array, and we will see further that by even/odd sum of array technique we will get the sum and hence Array given is analogous to weight array sum is analogous to weight.

Furthermore, in knapsack we do is on each item we check whether to pick that item and put it in the knapsack or not, in short words yes or no / True or false which is very much similar to our "Equal subset problem" in which we first see we pick elemet or not if sum equal we return true.

Date:

Sun Mon Tue Wed Thu Fri Sat

- Solution approach explained:

The array given

arr={ 3, 1, 1, 2, 2, 1 }.

The odd-even sum approach.

The sum of the given array will be:

sum = 3 + 1 + 1 + 2 + 2 + 1

sum = 10

$\frac{sum}{2} = \frac{10}{2} = 5.$

Now here notice the sum of the array is even so naturally we get the assurance that this array can be divided into or partitioned into subsets equally so that both of the subsets have equal sum. is possible

a counter example:

If the given array would be like

arr={ 3, 1, 1, 2, 2, 1, 1 }

The sum of this array will be:

sum = 3 + 1+1+2+2+1+1

sum = 11

$\frac{sum}{2} = \frac{11}{2} = 5.5$

This means it is odd hence this array CANNOT

be partitioned such that subsets have equal sum.
In this case False will be returned

Now for solving this the approach we going to use is "Topdown Knapsack 01" approach.

——• The code / Algorithm of the equal Partition - Problem or simply Partition - Problem.

4 Code / Algo for array sum:

```
int sum=0;                    // size of array.
  for (int i=0; i<size; i++)
  {
      sum = sum + arr[i];
  }
```

4 Main code / Algo

```
if (sum % 2 != 0)
{                          -> This condition means if
  return false;              array sum is odd.
}
```

Date:

Sun Mon Tue Wed Thu Fri Sat

else if (sum % 2 == 0) — the even sum

{ — ↑ size of array

return subset sum (arr, sum/2, n)

↙ function which will implement topdown approach.

↓ the given array

}

Subsetsum function:

t[n+1][sum+1] — true or false returned

Initialization code (main)

```
for (i=0 ; i<n+1 ; i++)
for (j=0; j< size+1 ; j+1)
{
    if (i==0)
        t[i][j] = False;
    if (j==0)
        t[i][j] = true;
```

Note:- we want one subset which equals to the even sum automatically it implies the remaining elements will sum up to the same sum hence both equal sum.

∴ i = size of array

j = item/sum.

=> we make t[n+1][W+1]

↑ size of array ↑ sum.

t[n+1][sum+1] array/2d array/matrix is made.

Date:

Sun Mon Tue Wed Thu Fri Sat

used OR because in boolean max value doesn't OR · make sense.

```
if (arr[i-1]!=j)
{
 t[i][j] = t[i][j-arr[i-1]] || t[i-1][j]
}
else
{
  t[i][j]= t[i-1][j]
}

return ( t[n][sum] )
{
```

wt array i.e given array

↳ last block of table either T/F return.

——• Dry run to further clearly explain our approach:

sum = 10

sum/2 = 5

sum % 2 == 0.

subset sum function.

Date:

Sun Mon Tue Wed Thu Fri Sat

n+1 sum+1

t[7][6]

	0	1	2	3	4	5	sum
0	T	F	F	F	F	F	arr: {3, 1, 1, 2, 2, 1}
1	T	F	F	T	F	F	sum:
2	T	T	F	F	T	F	
3	T	T	T	F	T	T	
4	T	F	T	T	T	T	
5	T	F	T	T	T	T	
6	T	T	T	T	T	T	

7

size of array

Initialization.

Consider arr[0][0] arr[]: 2 elmts.

means arr: empty sum: 0.

sum: 0 possible.

partition { } { } possible.

and if

arr empty

sum: 1 be

can't possible.

COPY

Date:

Sun Mon Tue Wed Thu Fri Sat

- i = 1 , j = 1

arr[0] = 3 < = 1 ×

t[1][1] = t[0][1] =

t[1][1] = F

- i = 1 , j = 2

arr[0] = 3 < = 2 ×

t[1][2] = t[0][2]

t[1][2] = F

- i = 1 , j = 3

arr[0] = 3 < = 3 ✓

t[1][3] = t[1][3-3] OR t[0][3]

t[1][3] = t[1][0] OR t[0][3]

t[1][3] = T || F

t[1][3] = T

- i = 1 , j = 4

arr[0] = 3 < = 4

t[1][4] = t[1][4-3] OR t[0][4]

= t[1][1] || t[0][4]

= F || F

= F

COPY