Date   Page

⇒ if end-start > diff
⇒ ↑ start.

Day - 39

Prefix and Suffix

⇒ Prefix: From starting.
⇒ Suffix: From ending.

6	4	5	-3	2	8

⇒ Prefix sum

Sum the elements from starting to end.

6	10	15	12	14	22

⇒ Suffix Sum

Sum the elements from ending to start.

22	16	12	7	10	8

Code

```
⇒ vector <int> prefix (n);
prefix [0] = arr[0];
for( int i = 1; i < n; i++){
        prefix[i] = prefix [i-1] + arr[i];
};
```

Date ______ Page ______

* <u>Sub-array:</u>

=) Some contagious part of the array.

$\boxed{1\,|\,2\,|\,3\,|\,4} \rightarrow n$

=) $\boxed{1\,|\,2}$ $\boxed{2\,|\,3\,|\,4}$ $\boxed{2\,|\,3}$ $\boxed{3\,|\,4}$

=) All are sub array of the given array.

1-size: $\boxed{1}$ $\boxed{2}$ $\boxed{3}$ $\boxed{4} \rightarrow n$

2-size: $\boxed{1\,|\,2}$ $\boxed{2\,|\,3}$ $\boxed{3\,|\,4} \rightarrow n-1$

3-size: $\boxed{1\,|\,2\,|\,3}$ $\boxed{2\,|\,3\,|\,4} \rightarrow n-2$

4-size: $\boxed{1\,|\,2\,|\,3\,|\,4} \rightarrow 1$

* <u>Divide array in 2 sub array with equal sum:</u>

$\boxed{3\,|\,4\,|\,-2\,|\,5\,|\,8\,|\,20\,|\,-10\,|\,8}$

=) First, we start with first element then check the sum of both subarray's.

=) If it is equal then return the index.

=) Otherwise, increase the pointer to the next element and do the same thing.

<u>Code</u>

array.

3 | 4

given array.

→ n

→ n-1

→ n-2

₹ 1

with

8

ment then
array's.
the index.
n to the
me thing.

```
for(i=0; i<n-1; i++){
    int sum1 = 0, sum2=0;
    for(j=0; j<=i; j++)          } n
        sum1 += arr[j];                  } n
    for(j=i+1; j<n; j++)  } n
        sum2 += arr[j];
    if (sum1 == sum2)
        return 1;
}
return 0;
```

Second Approach

3	4	-2	5	8	20	-10	8
↑ 3	↑ 7						

Total Sum = 36

⇒ First, we calculate total sum of the array.
⇒ After that, we start from 3 and check the differen b/w 3 & total sum.
⇒ If it equal that means subarray present.
⇒ If not then calculate the sum of 3 with 4 and do the same task.

⇒ $36-3 = 33 \neq 3$
⇒ $36-(3+4) = 29 \neq \cancel{3}\,7$
⇒ $36-(3+4-2) = 31 \neq \cancel{3}\,5$
⇒ $36-(3+4-2+5) = 26 \neq \cancel{3}\,12$
⇒ $36-(3+4-2+5+8) = \underline{18 = 18}$

Date ______ Page ______

Code

```
Total_Sum =0;
for (i=0; i<n ;i++)
        Total_Sum +=arr[i];
int prefix=0;
for( i=0; i<n-1; i++)
        prefix+= arr[i];
        ans= Total_sum -prefix;
        if (ans == prefix)
            return 1;
}
```

* Largest Sum Contiguous Subarray:

3	4	-5	8	-12	7	6	-2

⇒ For understanding, let's take a small example:

0	1	2	3
4	-6	2	8

The subarray are —

⇒ {0} {0, 1} {0, 1, 2} {0, 1, 2, 3}

⇒ {1}, {1, 2}, {1, 2, 3}

⇒ {2}, {2, 3}

⇒ {3}

⇒ So, we will calculate the prefix sum as — from 0 - 3, 1-3, 2-3, 3.

Date. Page

Code

```
int max = INT_MIN;
for( i=0; i<n; i++){
      prefix =0;
      for (j= i; j<n; j++){
            prefix += arr[j];
            maxi = max(maxi, prefix);
      }
}
return max;
```

=) There is another better approach to solve this question in O(N) time.

=) First we take two variables - prefix & max.

=) Now, we calculate prefix, & check ~~who is bigger between prefix & max~~ if prefix is bigger than max then update the max with prefix.

=) If we get any -ve value in prefix then update the prefix with zero (0).

=) This is Kadane's algo..

=)

4	-6	2	8

prefix = ~~4~~ ~~-2~~ ~~0~~ ~~2~~ 10   maxi = ~~4~~ 10

=) We convert -ve prefix to 0 because adding -ve no. to the prefix, decrease the prefix.

Date   Page

Code

```
Fmax = INT_MIN;
prefix = 0;
for( i=0; i<n; i++){
    prefix += arr[i];
    max = max( max, prefix);
    if (prefix <0)
        prefix =0;
}
```

* Max. difference b/w 2 element:

9	5	8	12	2	3	7	4

⇒ In this question, we have to return max. difference b/w 2 element, but difference is only calculated of ~~the~~ those elements that are come after the smaller element.

⇒ In Brute force approach, we ~~can eat~~ can select the first then find the greater & no. than that and calculate the diff. & store the max. difference.

⇒ $O(N^2) \rightarrow$ T.C.

⇒ In Second approach, we use suffix max to solve the problem because here we have to find the difference of the max. element from that element to the

Right. $\longrightarrow$ O(N) $\rightarrow$ T.C.

⇒ We can also do this in one traversal by ~~st calculating suffix~~ taking a variable & start with end then do the difference with suffix and store the max difference.

⇒ After ~~[illegible]~~ doing this return the max difference.

max.
erence
ts that
lement.

~~each~~
e
lculate
rence.

max
ere
of the
to the