Form the largest number

$n = 4$

arr = | 4 | 46 | 8 | 9 | (int)

ans = 98464

a = "4", b = "46"

case 1 , a+b = "446"

case 2 , b+a = "464"

n = 4

arr = | 4 | 46 | 8 | 9 | (int)

// input

arr = | "4" | "46" | "8" | "9" | (String)

// convert all int into String

(gmp)

a = "4"

b = "46"

a+b // "446"

b+a // "464"

// sort based on a+b & b+a

arr1 =

0	1	2	3
"9"	"8"	"46"	"4"

(String)

String ans = arr[0] + arr[1] + arr[2] + arr[3]

= "98464"

// make an answer as string

↳ Inbuilt f^n used to convert int into String

1) String.valueOf(value); → int

2) String.parseInt(value); → int

Code

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    String ans = formLargestNumber(arr, n);
    System.out.println(ans);
}
public static String formLargestNumber(int[] arr, int n) {
    // convert all int values to String
    String[] arr1 = new String[n];
    for (int i = 0; i < n; i++) {
        arr1[i] = String.valueOf(arr[i]);
    }

    // sort the array based on a+b and b+a
    Arrays.sort( arr1, (a, b) -> {
        String str1 = a + b;
        String str2 = b + a;

        return str2.compareTo(str1);    // largest number
        // return str1.compareTo(str2);    // smallest number
    } );

    // make a answer of String type
    String ans = "";
    for (int i = 0; i < n; i++) {
        ans = ans + arr1[i];
    }
    return ans;
}
```

str2 - str1;

str1 - str2;

sys.(ans[i]);

ans = ans + arr1[i] // ans = ""

i = 0 // ans = "" + "9" = "9"

i = 1 // ans = "9" + "8" = "98"

i = 2 // ans = "98" + "46" = "9846"

i = 3 // ans = "9846" + "4" = "98464"

⇒ Subarray (subpart of an array which is continuous in nature)

arr =

0	1	2	3
1	5	3	2

all subarrays

1
1 5
1 5 3
1 5 3 2

5
5 3
5 3 2

3
3 2

2

Note :-

Subarrays does not include repetation

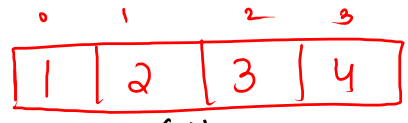
Print All Subarrays

$T.C = O(N^2 * N) \cong O(N^3)$

$S.C = O(1)$

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    printSubarrays(arr, n);
}
public static void printSubarrays(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            print(arr, i, j);
        }
    }
}
public static void print(int[] arr, int si, int ei) {
    for (int i = si; i <= ei; i++) {
        System.out.print(arr[i] + " ");
    }
    System.out.println();
}
```



(si) (ei)

i=0, j=0 → 1

i=0, j=1 → 1 2

i=0, j=2 → 1 2 3

i=0, j=3 → 1 2 3 4

i=1, j=1 → 2

i=1, j=2 → 2 3

i=1, j=3 → 2 3 4

i=2, j=2 → 3

i=2, j=3 → 3 4

i=3, j=3 → 4

Sum Equals Zero

arr =

5	-2	3	-1	4

all subarrays

sum	
5	5
3	5 -2
6	5 -2 3
5	5 -2 3 -1
9	5 -2 3 -1 4
-2	-2
1	-2 3
0	-2 3 -1
	-2 3 -1 4
	3
	3 -1
	3 -1 4
	-1
	-1 4
	4

return true → 0

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    boolean ans = findSum(arr, n);
    System.out.println(ans);
}
public static boolean findSum(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            int sum = checkSum(arr, i, j);
            if ( sum == 0 ) {
                return true;
            }
        }
    }
    return false;
}
public static int checkSum(int[] arr, int si, int ei) {
    int sum = 0;
    for (int i = si; i <= ei; i++) {
        sum += arr[i];
    }
    return sum;
}
```

$T.C = O(N^3)$

where N is size of array

$S.C = O(1)$

⇒ Kadane's algorithm

↳ used to find "maximum sum subarray"

arr =

5	-2	3	-1	4

all subarrays

sum	
5	5
3	5 -2
6	5 -2 3
5	5 -2 3 -1
9	5 -2 3 -1 4
-2	-2
1	-2 3
0	-2 3 -1
4	-2 3 -1 4
3	3
2	3 -1
6	3 -1 4
-1	-1
3	-1 4
4	4

maximum

Brute force :- $O(N^3)$

Kadane's algo :- $O(N)$