Part 4 week 4

23. Subset

```
# include<stdio.h>
int main()
{
   int n1,n2,i,j,count=0;
   printf("\nEnter the number of elements in array 1 : ");
   scanf("%d", &n1);
   printf("\nEnter the number of elements in array 2 : ");
   scanf("%d", &n2);
   int arr1[n1],arr2[n2];
   printf("\nEnter the %d positive elements in array 1 : ",n1);
   for(i=0;i<n1;i++)
   {
      scanf("%d",&arr1[i]);
   }
   printf("\nEnter the %d positive elements in array 2 : ",n2);
   for(i=0;i<n2;i++)
   {
      scanf("%d",&arr2[i]);
   }
   for (i = 0; i < n2; i++)
   {
      for (j = 0; j < n1; j++)
      {
         if(arr2[i] == arr1[j])
         {
                                    count++;
                                    arr1[j]=-1;
                              break;
         }
      }
   }
   if (count==n2)
      printf("\nArray 2 is a subset of Array 1\n ");
   else
      printf("\nArray 2 is not a subset of Array 1\n");
   return 0;
}
```

24. Frequency of al elements:

```
#include <stdio.h>

void findFrequency(int A[], int n)
{
    // create a count array of size `n` to store the count of all array elements
    int freq[n];

    for (int i = 0; i < n; i++) {
        freq[i] = 0;
    }

    // update frequency of each element
    for (int i = 0; i < n; i++) {
        freq[A[i]]++;
    }

    // iterate through the array to print frequencies
    for (int i = 0; i < n; i++)
    {
        if (freq[i]) {
            printf("%d appears %d times\n", i, freq[i]);
        }
    }
}

int main(void)
{
    int A[] = { 2, 3, 3, 2, 1 };
    int n = sizeof(A) / sizeof(A[0]);

    findFrequency(A, n);

    return 0;
```

25. find pair.

```
include <stdio.h>

void findPair (int nums[], int n, int target)
{
  int flag = 0;
```

```
  for (int i = 0; i < n; i++)
    {
      for (int j = i + 1; j < n; j++)
           {
             if (nums[i] + nums[j] == target)
               {
                 printf ("Pair found (%d, %d) \n", nums[i], nums[j]);
                 flag = 1;
               }
           }
    }
  if (flag == 0)
    {
      printf ("Pair not found");
    }
}

int main ()
{
  int nums[] = {5, 2, 3, 4, 1, 6, 7};
  int target = 7;

  int n = sizeof (nums) / sizeof (nums[0]);

  findPair (nums, n, target);

  return 0;
}
```

26.triplet sum

```
include <stdio.h>

bool find3Numbers(int A[], int arr_size, int sum)
{

    int l, r;


    // Fix the first element as A[i]

    for (int i = 0; i < arr_size - 2; i++) {
```

```
        // Fix the second element as A[j]
        for (int j = i + 1; j < arr_size - 1; j++) {

            // Now look for the third number
            for (int k = j + 1; k < arr_size; k++) {
                if (A[i] + A[j] + A[k] == sum) {
                    printf("Triplet is %d, %d, %d",
                           A[i], A[j], A[k]);
                    return true;
                }
            }
        }
    }

    // If we reach here, then no triplet was found
    return false;
}

int main()
{
    int A[] = { 1, 4, 45, 6, 10, 8 };
    int sum = 22;
    int arr_size = sizeof(A) / sizeof(A[0]);
    find3Numbers(A, arr_size, sum);
```

```
    return 0;
}
```

27. 4 sum. C

```
#include <stdio.h>
#include <stdlib.h>
int* sort_array(int* nums, int numsSize) {
    for (int i = 0; i < numsSize - 1; i++) {
        for (int j = 0; j < numsSize - i - 1; j++) {
            if (nums[j] > nums[j + 1]) {
                int temp = nums[j];
                nums[j] = nums[j + 1];
                nums[j + 1] = temp;
            }
        }
    }
    return nums;
}

int** fourSum(int* nums, int numsSize, int target, int* returnSize, int** returnColumnSizes)
 {

    nums = sort_array(nums, numsSize);

    *returnSize = 0;


    int **result = (int**)malloc(numsSize * numsSize * sizeof(int*));
    *returnColumnSizes = (int*)calloc(numsSize * numsSize, sizeof(int));

    for (int i = 0; i < numsSize - 3; i++) {
        // Skip duplicate elements
        if (i > 0 && nums[i] == nums[i - 1]) continue;

        for (int j = i + 1; j < numsSize - 2; j++) {
            // Skip duplicate elements
            if (j > i + 1 && nums[j] == nums[j - 1]) continue;

            int k = j + 1, l = numsSize - 1;
            while (k < l) {
                int current_sum = nums[i] + nums[j] + nums[k] + nums[l];
                if (current_sum == target) {
```

```
                result[*returnSize] = (int*)malloc(4 * sizeof(int));
                result[*returnSize][0] = nums[i];
                result[*returnSize][1] = nums[j];
                result[*returnSize][2] = nums[k];
                result[*returnSize][3] = nums[l];
                (*returnColumnSizes)[*returnSize] = 4;
                (*returnSize)++;


                while (k < l && nums[k] == nums[k + 1]) k++;
                while (k < l && nums[l] == nums[l - 1]) l--;

                k++;
                l--;
            } else if (current_sum > target) {
                l--;
            } else {
                k++;
            }
        }
    }
    return result;
}
```

28.. triplet in zero

```
#include <stdbool.h>
#include <stdio.h>

// Prints all triplets in arr[] with 0 sum

void findTriplets(int arr[], int n)
{

    bool found = false;

    for (int i = 0; i < n - 2; i++) {

        for (int j = i + 1; j < n - 1; j++) {
```

```
            for (int k = j + 1; k < n; k++) {
                if (arr[i] + arr[j] + arr[k] == 0) {
                    printf("%d %d %d\n", arr[i], arr[j],
                            arr[k]);
                    found = true;
                }
            }
        }
    }

    if (found == false)
        printf(" not exist \n");
}

int main()
{
    int arr[] = { 0, -1, 2, -3, 1 };
    int n = sizeof(arr) / sizeof(arr[0]);
    findTriplets(arr, n);
    return 0;
}
```

29. Count triplets

```
#include <algorithm>
#include <iostream>
#include <vector>
```

```
using namespace std;

int count_Triplets(int A[], int N){
     int count = 0;
     sort(A, A + N);
     for(int i = 0; i < N; i++){ // for first number
       for(int j = i + 1; j < N; j++){ // for second number
          for(int k = j + 1; k < N; k++){  // for third number
              if(A[i] + A[j] == A[k]){
                   count++; // increment count
              }
          }
       }
     }
  return count;
}

int main() {

           int A[] = {5,7,12,3,2};
           int N = 5;
           cout << count_Triplets(A, N);
           return 0;
}
```

30. Union and intersection :

```
#include <stdio.h>

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void findUnion(int arr1[], int size1, int arr2[], int size2) {
    int i = 0, j = 0;

    while (i < size1 && j < size2) {
        if (arr1[i] < arr2[j]) {
       printf("%d ", arr1[i++]);
        } else if (arr2[j] < arr1[i]) {
           printf("%d ", arr2[j++]);
```

```
        } else {
            printf("%d ", arr1[i++]);
            j++;
        }
    }

    while (i < size1) {
    printf("%d ", arr1[i++]);
    }

    while (j < size2) {
        printf("%d ", arr2[j++]);
    }
    printf("\n");
}

void findIntersection(int arr1[], int size1, int arr2[], int size2) {
    int i = 0, j = 0;

    while (i < size1 && j < size2) {
        if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr2[j] < arr1[i]) {
            j++;
        } else {
            printf("%d ", arr1[i++]);
            j++;
        }
    }
    printf("\n");
}

int main() {
    int arr1[] = {1, 3, 5, 7};
    int arr2[] = {2, 4, 6, 7};

    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    printf("Array 1: ");
    printArray(arr1, size1);

    printf("Array 2: ");
    printArray(arr2, size2);
```

```
    printf("Union: ");
    findUnion(arr1, size1, arr2, size2);

    printf("Intersection: ");
    findIntersection(arr1, size1, arr2, size2);

    return 0;
}
```

32. remove duplicate (different from already done)

```
#include <stdio.h>

int remove_duplicate(int A[], int N) {
    if (N == 0 || N == 1) {
        return N;
    }

    int X = 1;  // Initial count of distinct elements
    for (int i = 1; i < N; i++) {
        if (A[i] != A[i - 1]) {
            A[X++] = A[i];  // Update array with distinct elements
        }
    }

    return X;
}

int main() {
    int A[] = {1, 2, 2, 3, 4, 4, 5};
    int N = sizeof(A) / sizeof(A[0]);

    printf("Original Array: ");
    for (int i = 0; i < N; i++) {
        printf("%d ", A[i]);
    }

    int distinctCount = remove_duplicate(A, N);

    printf("\nArray after removing duplicates: ");
    for (int i = 0; i < distinctCount; i++) {
        printf("%d ", A[i]);
```

```
    }

    printf("\nNo. of distinct elements: %d\n", distinctCount);

    return 0;
}
```

33. K th element in array:

```
#include <stdio.h>

int kthElement(int arr1[], int arr2[], int N, int M, int K) {
    // Merge the arrays while maintaining sorted order
    int merged[N + M];
    int i = 0, j = 0, k = 0;

    while (i < N && j < M) {
        if (arr1[i] < arr2[j])
            merged[k++] = arr1[i++];
        else
            merged[k++] = arr2[j++];
    }


    while (i < N)
        merged[k++] = arr1[i++];


    while (j < M)
        merged[k++] = arr2[j++];
    return merged[K - 1];
}


int main() {
    int arr1[] = {2, 3, 4, 8, 12};
    int arr2[] = {1, 5, 7, 10};
    int N = sizeof(arr1) / sizeof(arr1[0]);
    int M = sizeof(arr2) / sizeof(arr2[0]);
    int K = 5;

    int result = kthElement(arr1, arr2, N, M, K);

    printf("The element at position %d is: %d\n", K, result);
```

```
    return 0;
}
```

34. length of largest sub array:

```
#include <stdio.h>

int lenOfLongSubarr(int A[], int n, int K) {
    int maxLength = 0;
    int currentSum = 0;
    int start = 0;

    for (int end = 0; end < n; ++end) {
        currentSum += A[end];

        while (currentSum > K && start <= end) {
            currentSum -= A[start];
            start++;
        }

        int currentLength = end - start + 1;
        if (currentLength > maxLength) {
            maxLength = currentLength;
        }
    }

    return maxLength;
}

// Example usage
int main() {
    int A[] = {1, 2, 3, 4, 5};
    int n = sizeof(A) / sizeof(A[0]);
    int K = 11;

    int result = lenOfLongSubarr(A, n, K);

    printf("The length of the longest subarray with sum at most %d is: %d\n", K, result);

    return 0;
}
```

35.