1. Hash tables, collisions, implement in  Java using Arrays

2. Dynamic programming, DP for fibonacci

3. Sorrting algorithms, implement in Java

class HashMapArr {

    String[] key;

    int[] val;

    int index = 0;

    HashMapArr(int n) {

        this.key = new String[n];

        this.val = new int[n];

    }

    void put(String key, int val) {

        this.key[index] = key;

        this.val[index] = val;

        index++;

    }

    int get(String key) {

        for (int i = 0; i < key.length(); i++) {

            if (this.key[i] == key) {

                return this.val[i];

            }

        }

        return -1;

    }

}

public class Main {

    public static void main(String[] args) {

        HashMapArr h = new HashMapArr(5);

        h.put("one", 1);

        h.put("two", 2);

        System.out.println(h.get("two"));

        System.out.println(h.get("one"));

        System.out.println(h.get("three"));

    }

}

//Hash Table

class HashTable {

    String[] keys;

    int[] values;

    int size;

    HashTable(int n) {

        keys = new String[n];

        values = new int[n];

        size = n;

    }

    int hash(String key){

        return Math.abs(key.hashCode())%size;

    }

    void put(String key, int value){

        int index = hash(key);

        if(keys[index]==null){

            keys[index]=key;

            values[index]=value;

        }

        else {

            int currIndex = index;

            do {

                currIndex = (currIndex+1)%size;

                if(keys[currIndex]==null){

                    keys[currIndex] = key;

                    values[currIndex] = value;

                    return;

                }

            } while(currIndex!=index);

            System.out.println("Cannot afford new values, hash table is full");

        }

    }

    int get(String key) {

        int index = hash(key);

        if(keys[index]==key) {

            return values[index];

        }

        else {

            int currIndex = index;

            do {

                currIndex = (currIndex+1)%size;

                if(keys[currIndex]==key){

                    return values[currIndex];

                }

            } while(currIndex!=index);

            return -1;

        }

    }

}

public class Main{

    public static void main(String args[]) {

       HashTable ht = new HashTable(10);

       ht.put("a",1);

       ht.put("b",2);

       ht.put("c",3);

       ht.put("d",4);

       System.out.println(ht.get("a"));

       System.out.println(ht.get("d"));

       System.out.println(ht.get("k"));

    }

}

import java.util.Arrays;

public class DP {

    public static void main(String[] args) {

        int[] arr = new int[10];

        arr[0] = 0;

        arr[1] = 1;

        for(int i = 2; i < arr.length; i++) {

            arr[i] = arr[i-1] + arr[i-2];

        }

        System.out.println(Arrays.toString(arr));

    }

}

//Selection sort

public class Main{

    public static void main(String args[]) {

        int[] arr = {10,8,5,15,3};

        int temp;

        for(int i=0;i<arr.length;i++){

            int minIndex = i;

            for(int j=i+1;j<arr.length;j++) {

                if(arr[j]<arr[minIndex]){

                    minIndex = j;

                }

            }

            temp = arr[minIndex];

            arr[minIndex] = arr[i];

            arr[i] = temp;

        }

        for(int i=0;i<arr.length;i++) {

            System.out.print(arr[i] + " ");

        }

    }

}

//bubble sort

public class Main{

    public static void main(String args[]) {

        int[] arr = {10,8,5,15,3};

        int temp;

        for(int i=0;i<arr.length;i++){

            for(int j=0;j<arr.length-1;j++){

                if(arr[j]>arr[j+1]) {

                    temp = arr[j+1];

                    arr[j+1] = arr[j];

                    arr[j]=temp;

                }

            }

        }

        for(int i=0;i<arr.length;i++) {

            System.out.println(arr[i]);

        }

    }

}

//insertion sort

public class Main{

    public static void main(String args[]) {

        int[] arr = {10,8,5,15,3};

        int temp;

        for(int i=0;i<arr.length;i++){

            int key = arr[i];

            int j = i-1;

            while(j>=0 && arr[j]>key) {

                arr[j+1] = arr[j];

                j = j-1;

            }

            arr[j+1] = key;

        }

        for(int i=0;i<arr.length;i++) {

            System.out.println(arr[i]);

        }

    }

}

//merge sort

public class Main{

    public static void mergeSort(int[] arr) {

        if(arr.length<=1) {

            return;

        }

        int mid = arr.length/2;

        int[] left = new int[mid];

        int[] right = new int[arr.length - mid];

        for(int i=0;i<mid;i++){

            left[i] = arr[i];

        }

        for(int i=mid;i<arr.length;i++){

            right[i-mid] = arr[i];

        }

        mergeSort(left);

        mergeSort(right);

        merge(arr,left,right);

    }

    static void  merge(int[] arr, int[] left, int[] right) {

        int i=0,j=0,k=0;

        while(i<left.length && j<right.length) {

            if(left[i]<=right[j]){

                arr[k] = left[i];

                i++;

            }

            else {

                arr[k] = right[j];

                j++;

            }

            k++;

        }

        while(i<left.length){

            arr[k] = left[i];

            i++;

        }

        while(j<right.length){

            arr[k] = right[j];

            j++;

        }

    }

    public static void main(String args[]) {

        int[] arr = {10,8,5,15,3};

        mergeSort(arr);

        for(int i=0;i<arr.length;i++) {

            System.out.println(arr[i]);

        }

    }

}

//quick sort

import java.util.Arrays;

public class Main{

    static void quickSort(int[] array, int low, int high){

        if(low<high) {

            int partitionIndex = partition(array, low, high);

        quickSort(array,low,partitionIndex-1);

        quickSort(array,partitionIndex+1,high);

        }

    }

    static int partition(int[] array, int low, int high) {

        int pivotIndex = low + (high-low)/2;

        int pivot = array[pivotIndex];

        array = swap(array,pivotIndex,high);

        int i = low;

        int j = high-1;

        while(i<=j){

            while(array[i]<pivot){

                i++;

            }

            while(array[j]>pivot){

                j--;

            }

            if(i<=j){

                swap(array,i,j);

                i++;

                j--;

            }

        }

        array = swap(array,i,high);

       return i;

    }

    static int[] swap(int[] arr, int i, int j){

        int temp = arr[i];

        arr[i] = arr[j];

        arr[j] = temp;

        return arr;

    }

    public static void main(String args[]) {

        int[] arr = {10,8,5,15,3,19,21};

        quickSort(arr,0,6);

        for(int i=0;i<arr.length;i++) {

            System.out.println(arr[i]);

        }

    }

}

