**Demo Exam 2023 Algorithms and Data structure (Usman A):**

* NB ! Q1 vil mest sannsynlig komme på eksamen (lurt å kunne dette!!)
* Write the step-by-step algorithm when the question (write an algoritm) is given

**Q1:** **Write a step-by-step algorithm that finds the largest number in a list (an array) of n numbers.**

public void largestNumberInArray(){  
 int [] arr = {1,2,3,4,5,32,7,8,9,11};  
 int n = arr.length;  
 int max = arr[0];  
 for (int i = 0; i <n ; i++) {  
 if (arr[i]>max){  
 max=arr[i];  
  
 }  
 }  
 System.*out*.println("Largest number in array:"+max);  
}

**Step by Step algorithm:**

1. Initialize a variable in the type of an int and call it: “max” and set the max variable equal to arr[0] as default value.
2. Create a loop (for loop) that goes through each element in the array starting from the first element.
3. Create an if statement inside the loop, if element is greater than the max value, set the max value equal to the element.
4. Print the max value using System.out.println statement outside the loop and the answer should be displayed 😊

**Q2:**

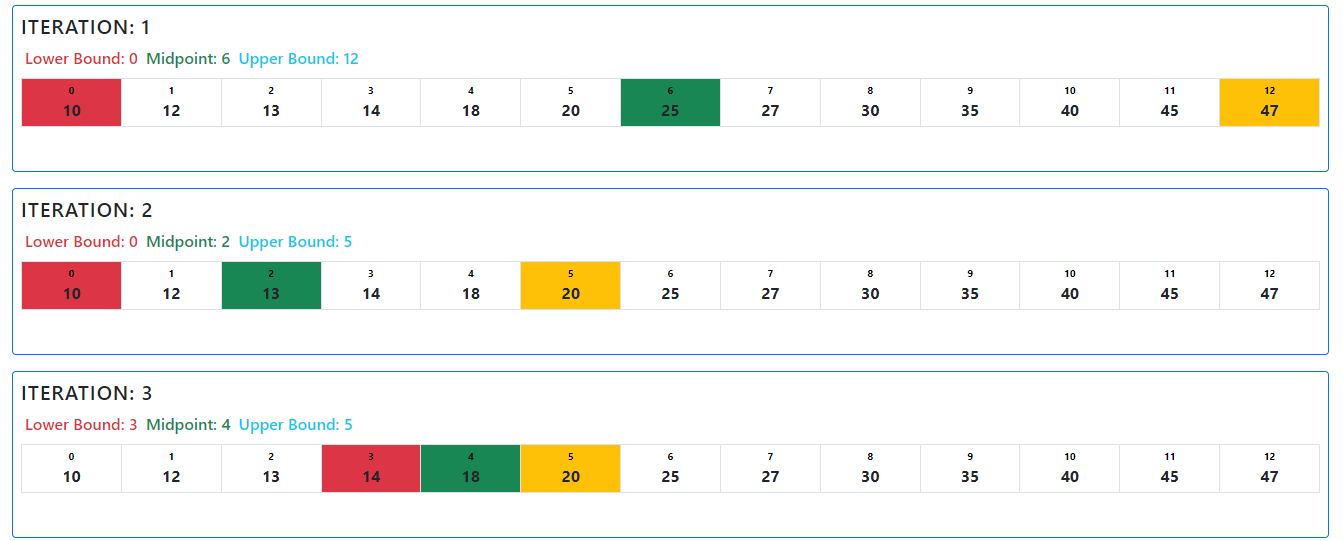
**Binary search:**

**Q3: i) Write an algorithm and solution to Binary Search with a Recursive version (10 points) Suppose x = 18, and we have the following array: [10,12,13,14,18,20,25,27,30,35,40,45,47]**

package Algoritmer.exercises;  
  
import java.util.Arrays;  
  
public class BinarySearch {  
  
 public void search(){  
 int [] arr ={1,2,3,4,5,6,7,8,9,10};  
 System.*out*.println(Arrays.*toString*(arr));  
 int key =9;  
 System.*out*.println("Key to be searched:"+key);  
  
 int low =0;  
 int high = arr.length-1;  
  
 int mid = (low+high)/2;  
  
 while (low <=high){  
 if (arr[mid]==key){  
 System.*out*.println("Element is found at index:" + mid);  
 break;  
 }if (arr[mid] < key) {  
 low = mid+1;  
 } else {  
 high=mid-1;  
 }  
 mid = (low+high)/2;  
  
 if (high>low){  
 System.*out*.println("Key is not found!");  
 }  
  
 }  
 }  
  
 public static void main(String[] args) {  
 BinarySearch bs = new BinarySearch();  
 bs.search();  
 }  
}

**Step by Step Binary Search Algorithm:**

1. Initial the variables low, high and key in the type of “int”.
2. Set low equal to 0 and high equal to ‘arr. length-1”.
3. Create a loop (While loop) to divide the list or array in half. While the low index is less or equal to the “high” index the binary search should be performed.
4. Use the formula (low+high)/2 to calculate the middle element from the array/list.
5. Compare the key element with the middle element.
6. Create an if statement inside the loop, if key is equal to middle element, then return the index position and the key is found!
7. Create an else-if statement, if key is greater than the middle element, then the key is located at the right subarray, repeat step 1-3 to find the key’s location.
8. Create a the last else-if statement, if key is lower than the middle element, choose the left subarray. Repeat step 1-3 to find it’s location 😊
9. Create an else statement, if the key is not found it’s the message (“Key’s not found”) should be displayed or return index-1.



18 is located on index x=4

**ii) When not to use Divide and Conquer (5 points) Short answer 3-4 sentences**

**Q4:**

**(i) Why has Bubble sort proved inefficient compared to other sorting algorithms? (4 points)**

**(ii) Sort the following list with Bubble Sort, Merge Sort and Quick Sort 123 34 189 56 150 12 9 240 (Skal man bruke java kode?? Send mail til foreleseren)**

**Bubble Sort:**

class BubbleSort {  
  
 void bubbleSort(int [] arr){  
 int n = arr.length;  
 for (int i = 0; i <n ; i++) {  
 for (int j = 0; j < n-i-1; j++) {  
 if (arr[j]>arr[j+1]){  
 int temp = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1]=temp;  
 }  
 }  
 }  
 }  
 void printArray(int [] arr){  
 int n = arr.length;  
 for (int i = 0; i <n ; i++) {  
 System.*out*.println(arr[i]);  
 }  
 }  
  
 public static void main(String[] args) {  
 int [] arr = {123, 34, 189, 56, 150 ,12 ,9 ,240};  
 BubbleSort ob = new BubbleSort();  
 System.*out*.println("Unsorted:");  
 ob.printArray(arr);  
 ob.bubbleSort(arr);  
 System.*out*.println("Sorted:");  
 ob.printArray(arr);  
 }  
}

**Quick Sort:**

package Algoritmer.exercises;  
  
import java.io.\*;  
 class QuickSort {  
  
 static void swap(int [] arr , int i , int j){  
 int temp = arr[i];  
 arr[i]=arr[j];  
 arr[j]=temp;  
 }  
   
 static int partition(int [] arr , int low , int high){  
 int pivot = arr[high];  
 int i = (low-1);  
  
 for (int j = low; j <=high-1 ; j++) {  
 if (arr[j]<pivot){  
 i++;  
 *swap*(arr , i, j);  
 }  
 }  
 *swap*(arr , i+1 ,high);  
 return (i+1);  
 }  
   
 static void quickSort(int [] arr , int low , int high){  
 if (low <high){  
 int pi =*partition*(arr , low ,high);  
   
 *quickSort*(arr , low ,pi-1);  
 *quickSort*(arr , pi+1 ,high);  
   
   
   
 }  
 }  
 static void printArray(int [] arr, int size){  
 for (int i = 0; i <size ; i++) {  
 System.*out*.println(arr[i]);  
 }  
 }  
  
 public static void main(String[] args) {  
 int [] arr = {123, 34, 189, 56, 150 ,12 ,9 ,240};  
 int n = arr.length;  
   
 System.*out*.println("Unsorted list:");  
 *printArray*(arr,n);  
 *quickSort*(arr, 0 , n-1);  
 System.*out*.println("Sorted list:");  
 *printArray*(arr , n);  
   
 }  
  
  
}

**Merge Sort:**

package Algoritmer.exercises;  
  
/\* Java program for Merge Sort \*/  
public class MergeSort {  
 void merge (int arr [] , int l , int m , int r){  
  
 int n1= m-l +1;  
 int n2 = r-m;  
  
 int L[] = new int [n1];  
 int R[] = new int[n2];  
  
 for (int i = 0; i < n1; i++) {  
 L[i]=arr[i+1];  
 }  
 for (int j = 0; j <n2 ; j++) {  
 R[j] = arr[m+1+j];  
 }  
  
 int i = 0 , j =0;  
  
 int k =l;  
  
 while (i <n1 && j < n2){  
  
 if (L[i] <=R[j]) {  
 arr[k] = L [i];  
  
 i++;  
 } else {  
 arr[k] = R [j];  
 j++;  
 }  
 k++;  
 }  
 while (i<n1){  
 arr[k] = L[i];  
 i++;  
 k++;  
 }  
  
 while (j <n2){  
 arr[k] = R[j];  
 j++;  
 k++;  
 }  
 }  
  
 void sort( int arr [] , int l , int r){  
  
 if ( l < r){  
 int m = l + (r-l)/2;  
  
 sort(arr , l , m);  
 sort(arr , m+1 , r);  
  
 merge(arr , l , m , r);  
 }  
 }  
  
 static void printArray( int [] arr){  
 int n = arr.length;  
 for (int i = 0; i <n ; i++) {  
 System.*out*.println(arr [i] + " ");  
  
 System.*out*.println();  
 }  
 }  
  
 public static void main(String[] args) {  
  
 int [] arr = {123, 34, 189, 56, 150 ,12 ,9 ,240};  
 int n = arr.length;  
 System.*out*.println("Unsorted list:");  
 *printArray*(arr);  
 System.*out*.println("Sorted list:");  
 MergeSort ms = new MergeSort();  
 ms.sort(arr , 0 ,n-1 );  
 *printArray*(arr);  
 }  
}

**(iii) provide a graphical representation of solving each of the sorting algorithms.**

(Se SkriveBoka for graphical representation av de ulike sorting algoritmene)

**Merge Sort Table format:**

**Arr = [123 34 189 56 150 12 9 240]**

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **U** | **V** | **Sorted (Result)** |
| **1** | 123,34,189,56 | 150, 12, 9, 240 | **9** |
| **2** | 123,34,189,56 | 150, 12, 9, 240 | **9,12** |
| **3** | 123,34,189,56 | 150, 12, 9, 240 | **9,12,34** |
| **4** | 123,34,189,56 | 150, 12, 9, 240 | **9,12,34,56** |
| **5** | 123,34,189,56 | 150, 12, 9, 240 | **9,12,34,56,123** |
| **6** | 123,34,189,56 | 150, 12, 9, 240 | **9,12,34,56,123,150** |
| **7** | 123,34,189,56 | 150, 12, 9, 240 | **9,12,34,56,123,150,189** |
| **----------------** | 123,34,189,56 | 150, 12, 9, 240 | **9,12,34,56,123,150,189,240**  **(Final values)** |

**Q6 : Short answers (3-4 sentences) are required for these 10 sub-questions (mangler) !!!**