bfs.java 1

C: > Users > naima > Desktop > algo > bfs.java > bfs > bfs(ArrayList<ArrayList<Integer>> adj)

```java
import java.util.*;

public class bfs {

    // bfs for  single connected component
    static ArrayList<Integer> bfs(ArrayList<ArrayList<Integer>> adj) {
        ArrayList<Integer> result = new ArrayList<>();
        boolean[] visited = new boolean[adj.size()];
        Queue<Integer> queue = new LinkedList<>();

        queue.add(0);
        visited[0] = true;

        while (!queue.isEmpty()) {
            int node = queue.poll();
            result.add(node);

            for (int neighbor : adj.get(node)) {
                if (!visited[neighbor]) {
                    visited[neighbor] = true;
                    queue.add(neighbor);
                }
            }
        }
        return result;
    }

    Run main | Debug main
    public static void main(String[] args) {
        int V = 5;
        ArrayList<ArrayList<Integer>> adj = new ArrayList<>();
        for (int i = 0; i < V; i++) {
            adj.add(new ArrayList<>());
        }


        adj.get(0).add(1);
        adj.get(1).add(0);

    }}
```

heapsortJava

C: > Users > naima > Desktop > algo > heapsortJava

```java
public class heapsort  {


    static void heapify(int[] arr, int n, int i) {
        int largest = i;
        int l = 2 * i + 1;//left
        int r = 2 * i + 2;
        if (l < n && arr[l] > arr[largest])
            largest = l;

        if (r < n && arr[r] > arr[largest])
            largest = r;
        if (largest != i) {
            int temp = arr[i];
            arr[i] = arr[largest];
            arr[largest] = temp;


            heapify(arr, n, largest);
        }
    }
    static void heapSort(int[] arr) {
        int n = arr.length;
        for (int i = n / 2 - 1; i >= 0; i--)
            heapify(arr, n, i);
         for (int i = n - 1; i > 0; i--) {

            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;

            heapify(arr, i, 0);}}

    public static void main(String[] args) {
        int[] arr = { 9, 4, 3, 8, 10, 2, 5 };

        heapSort(arr);

        for (int i = 0; i < arr.length; ++i)
            System.out.print(arr[i] + " ");
    }
}
```

huffman.java 1

C: > Users > naima > Desktop > algo > huffman.java > ...

```java
import java.util.*;

public class huffman {

    static class Node {
        int freq;
        char ch;
        Node left, right;

        Node(int f, char c) {
            freq = f;
            ch = c;
        }

        Node(int f, Node l, Node r) {
            freq = f;
            left = l;
            right = r;
        }
    }


    static void generateCodes(Node root, String code, ArrayList<String> codes) {
        if (root.left == null && root.right == null) {
            codes.add(code);
            return;
        }
        generateCodes(root.left, code + "0", codes);
        generateCodes(root.right, code + "1", codes);
    }

    Run main | Debug main
    public static void main(String[] args) {
        String s = "abcdef";
        int[] freq = {5, 9, 12, 13, 16, 45};
        int n = s.length();


        PriorityQueue<Node> pq = new PriorityQueue<>(Comparator.comparingInt(a -> a.freq));
        for (int i = 0; i < n; i++)
            pq.add(new Node(freq[i], s.charAt(i)));


        while (pq.size() > 1) {
            Node left = pq.poll();
            Node right = pq.poll();
            pq.add(new Node(left.freq + right.freq, left, right));
        }

        Node root = pq.poll();


        ArrayList<String> codes = new ArrayList<>();
        generateCodes(root, "", codes);

        System.out.println("Huffman Codes:");
        for (int i = 0; i < n; i++)
            System.out.println(s.charAt(i) + ": " + codes.get(i));
    }
}
```

```java
public class InsertionSort {

    void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++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;
        }
    }

    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");

        System.out.println();
    }

    public static void main(String args[])
    {
        int arr[] = { 12, 11, 13, 5, 6 };

        InsertionSort ob = new InsertionSort();
        ob.sort(arr);

        printArray(arr);
    }
}
```

```java

class knapsack {

    static int knapsackRec(int W, int[] val, int[] wt, int n) {

        if (n == 0 || W == 0)
            return 0;

        int pick = 0;

        if (wt[n - 1] <= W)
            pick = val[n - 1] + knapsackRec(W - wt[n - 1], val, wt, n - 1);

        int notPick = knapsackRec(W, val, wt, n - 1);

        return Math.max(pick, notPick);
    }

    static int knapsack(int W, int[] val, int[] wt) {
        int n = val.length;
        return knapsackRec(W, val, wt, n);
    }

    Run main | Debug main
    public static void main(String[] args) {
        int[] val = {1, 2, 3};
        int[] wt = {4, 5, 1};
        int W = 4;

        System.out.println(knapsack(W, val, wt));
    }
}
```

kruskal.java

C: > Users > naima > Desktop > algo > kruskal.java > kruskal > kruskal(int V, int[][] edges)

```java
public class kruskal {

    // find  parent
    static int find(int[] parent, int x) {
        if (parent[x] != x)
            parent[x] = find(parent, parent[x]);
        return parent[x];
    }


    static void union(int[] parent, int x, int y) {
        int px = find(parent, x);
        int py = find(parent, y);
        parent[py] = px;
    }

    static int kruskal(int V, int[][] edges) {


        for (int i = 0; i < edges.length - 1; i++) {
            for (int j = i + 1; j < edges.length; j++) {
                if (edges[i][2] > edges[j][2]) {
                    int[] temp = edges[i];
                    edges[i] = edges[j];
                    edges[j] = temp;
                }
            }
        }

        int[] parent = new int[V];
        for (int i = 0; i < V; i++)
            parent[i] = i;

        int cost = 0;
        int edgeCount = 0;


        for (int i = 0; i < edges.length && edgeCount < V - 1; i++) {
            int u = edges[i][0];
            int v = edges[i][1];
            int w = edges[i][2];


            if (find(parent, u) != find(parent, v)) {
                union(parent, u, v);
                cost += w;
                edgeCount++;
                System.out.println(u + " - " + v + " : " + w);
            }
        }

        return cost;
    }

    Run main | Debug main
    public static void main(String[] args) {
        int V = 4;
        int[][] edges = {
            {0, 1, 10},
            {0, 2, 6},
            {0, 3, 5},
            {1, 3, 15},
            {2, 3, 4}
        };

        int mstCost = kruskal(V, edges);
        System.out.println("Total cost of MST = " + mstCost);
    }
}
```

linersearch.java

C: > Users > naima > Desktop > algo > linersearch.java > linersearch

```java
class linersearch {
    public static int search(int arr[], int N, int x)
    {

        for (int i = 0; i < N; i++) {
            if (arr[i] == x)
                return i;
        }
        return -1;
    }

    Run main | Debug main
    public static void main(String args[])
    {
        int arr[] = { 2, 3, 4, 10, 40 };
        int x = 10;

        int result = search(arr, arr.length, x);
        if (result == -1)
            System.out.print(
                "Element is not present in array");
        else
            System.out.print("Element is present at index "
                             + result);
    }
}
```

linersearch.java | nthFibonacci.java

C: > Users > naima > Desktop > algo > nthFibonacci.java > nthFibonacci

```java
class nthFibonacci {



    static int nthFibonacci(int n){

        if (n <= 1) {
            return n;
        }

        return nthFibonacci(n - 1) + nthFibonacci(n - 2);
    }

    Run main | Debug main
    public static void main(String[] args){
        int n = 5;
        int result = nthFibonacci(n);
        System.out.println(result);
    }
}
```

quicksort.java

C: > Users > naima > Desktop > algo > quicksort.java > quicksort > quickSort(int[] a

```java
import java.util.Arrays;

class quicksort {

    static int partition(int[] arr, int low, int high) {
        int pivot = arr[low];
        int i = low + 1;

        for (int j = low + 1; j <= high; j++) {
            if (arr[j] < pivot) {
                swap(arr, i, j);
                i++;
            }
        }


        swap(arr, low, i - 1);
        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 swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    Run main | Debug main
    public static void main(String[] args) {
        int[] arr = {9, 3, 7, 1, 8};
        quickSort(arr, 0, arr.length - 1);
        System.out.println(Arrays.toString(arr));
    }
}

```

selectionsort.Java

C: > Users > naima > Desktop > algo > selectionsort.Java

```java
class selectionsort {

    static void selectionSort(int[] arr) {
        for (int i = 0; i < arr.length - 1; i++) {

            int minIndex = i;

            for (int j = i + 1; j < arr.length; j++) {
                if (arr[j] < arr[minIndex]) {
                    minIndex = j;
                }
            }

            int temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
        }
    }

    static void printArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }

    public static void main(String[] args) {
        int[] arr = {64, 25, 12, 22, 11};

        System.out.print("Original: ");
        printArray(arr);

        selectionSort(arr);

        System.out.print("Sorted: ");
        printArray(arr);
    }
}
```

TowerOfHanoi Java

C: > Users > naima > Desktop > algo > TowerOfHanoi Java

```java
class TowerOfHanoi {

    static void hanoi(int n, char from, char to, char aux) {
        if (n == 0)
            return;

        hanoi(n - 1, from, aux, to);
        System.out.println("Move disk " + n + " from " + from + " to " + to);
        hanoi(n - 1, aux, to, from);
    }

    public static void main(String[] args) {
        int n = 3;
        hanoi(n, 'A', 'C', 'B');
    }
}
```

linersearch.java | nthFibonacci.java | prims.java

> Users > naima > Desktop > algo > prims.java > ...

```java
import java.util.*;

public class prims {


    private int minKey(int[] key, boolean[] mstSet) {
        int min = Integer.MAX_VALUE, minIndex = -1;
        for (int v = 0; v < key.length; v++) {
            if (!mstSet[v] && key[v] < min) {
                min = key[v];
                minIndex = v;
            }
        }
        return minIndex;
    }

    private void printMST(int[] parent, int[][] graph) {
        System.out.println("Edge \tWeight");
        for (int i = 1; i < graph.length; i++)
            System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
    }


    public void primMST(int[][] graph) {
        int V = graph.length;
        int[] parent = new int[V];
        int[] key = new int[V];
        boolean[] mstSet = new boolean[V];

        Arrays.fill(key, Integer.MAX_VALUE);
        key[0] = 0;
        parent[0] = -1;

        for (int count = 0; count < V - 1; count++) {
            int u = minKey(key, mstSet);
            mstSet[u] = true;

            for (int v = 0; v < V; v++) {
                if (graph[u][v] != 0 && !mstSet[v] && graph[u][v] < key[v]) {
                    parent[v] = u;
                    key[v] = graph[u][v];
                }
            }
        }

        printMST(parent, graph);
    }

    Run main | Debug main
    public static void main(String[] args) {
        prims mst = new prims();
        int[][] graph = {
            {0, 2, 0, 6, 0},
            {2, 0, 3, 8, 5},
            {0, 3, 0, 0, 7},
            {6, 8, 0, 0, 9},
            {0, 5, 7, 9, 0}
        };

        mst.primMST(graph);
    }
}
```