

Assignment 2

Code:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.PriorityQueue;

class Node implements Comparable<Node> {
    private static int idCounter = 0;
    public int id;

    public Node parent = null;

    public List<Edge> neighbors;

    public double f = Double.MAX_VALUE;
    public double g = Double.MAX_VALUE;
    // Hardcoded heuristic
    public double h;

    Node(double h){
        this.h = h;
        this.id = idCounter++;
        this.neighbors = new ArrayList<>();
    }

    @Override
    public int compareTo(Node n) {
        return Double.compare(this.f, n.f);
    }
}
```

```
public static class Edge {  
    Edge(int weight, Node node){  
        this.weight = weight;  
        this.node = node;  
    }  
  
    public int weight;  
    public Node node;  
}
```

```
public void addBranch(int weight, Node node){  
    Edge newEdge = new Edge(weight, node);  
    neighbors.add(newEdge);  
}
```

```
public double calculateHeuristic(Node target){  
    return this.h;  
}
```

```
public static Node aStar(Node start, Node target) {  
  
    PriorityQueue<Node> openList = new PriorityQueue<>();  
    PriorityQueue<Node> closedList = new PriorityQueue<>();  
  
    start.f = start.g + start.calculateHeuristic(target);  
    openList.add(start);  
}
```

```

while(!openList.isEmpty()){

    Node n = openList.peek();

    if(n == target) {
        return n;
    }

    for(Node.Edge edge : n.neighbors) {
        Node m = edge.node;

        double totalWeight = n.g + edge.weight;

        if(!openList.contains(m) && !closedList.contains(m)) {
            m.parent = n;

            m.g = totalWeight;

            m.f = m.g + m.calculateHeuristic(target);

            openList.add(m);

        }

        else {

            if(totalWeight < m.g) {

                m.parent = n;

                m.g = totalWeight;

                m.f = m.g + m.calculateHeuristic(target);

                if(closedList.contains(m)) {

                    closedList.remove(m);

                    openList.add(m);

                }

            }

        }

        System.out.println(" At Node :"+ m.id);

        System.out.println("g(x) :"+ m.g + " " +"h(x) :"+ m.h + " " +"f(x) :"+ m.f);

        System.out.println();
    }
}

```

```

        }

        openList.remove(n);
        closedList.add(n);

    }

    return null;
}

public static void printPath(Node target){
    Node n = target;
    if(n==null)
        return;

    List<Integer> ids = new ArrayList<>();
    while(n.parent != null){
        ids.add(n.id);
        n = n.parent;
    }
    ids.add(n.id);
    Collections.reverse(ids);

    for(int id : ids){
        System.out.print(id + " ");
    }
    System.out.println("");
}

public static void main(String[] args) {
    Node head = new Node(3);

    head.g = 0;

    Node n1 = new Node(2);
    Node n2 = new Node(2);
    Node n3 = new Node(2);

    head.addBranch(1, n1);
    head.addBranch(5, n2);

```

```

head.addBranch(2, n3);

n3.addBranch(1, n2);

Node n4 = new Node(1);

Node n5 = new Node(1);

Node target = new Node(6);

n1.addBranch(7, n4);

n2.addBranch(4, n5);

n3.addBranch(6, n4);

n4.addBranch(3, target);

n5.addBranch(1, n4);

n5.addBranch(3, target);

Node res = aStar(head, target);

System.out.println("Best path to node "+ target.id + " is ");

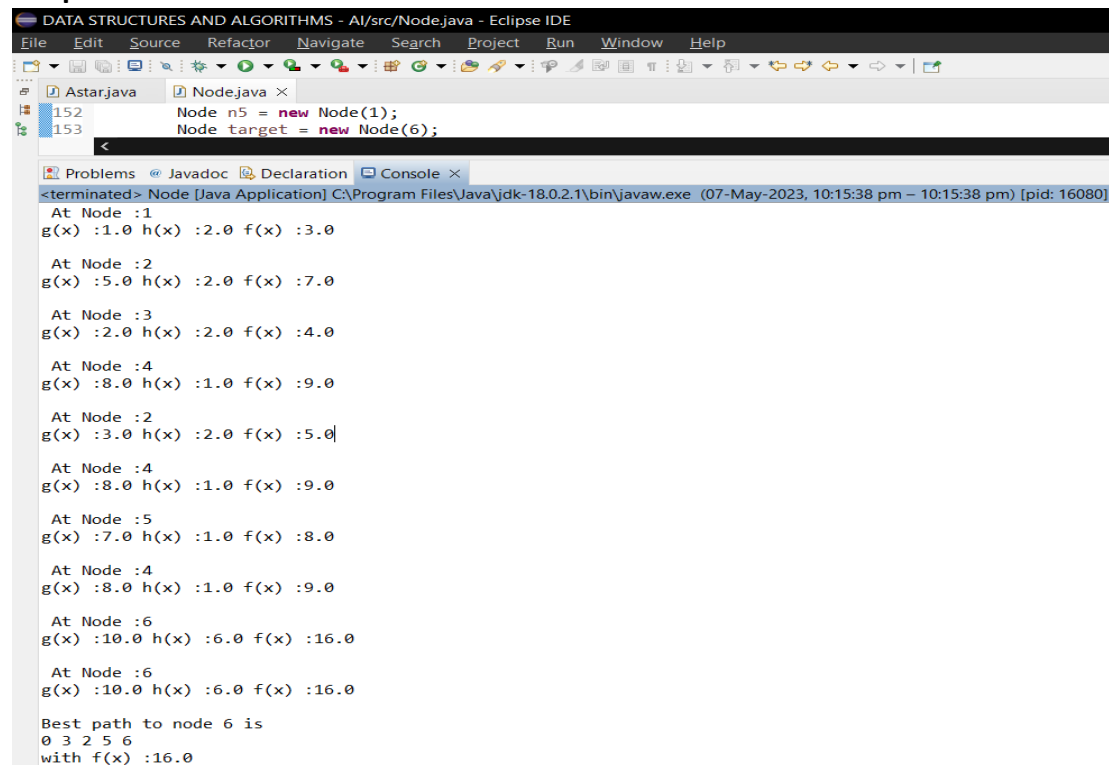
printPath(res);

System.out.println("with f(x) :"+ target.f);

}}

```

Output:



```

DATA STRUCTURES AND ALGORITHMS - AI/src/Node.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Astarjava Node.java x
152 Node n5 = new Node(1);
153 Node target = new Node(6);
<
Problems Javadoc Declaration Console x
<terminated> Node [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (07-May-2023, 10:15:38 pm - 10:15:38 pm) [pid: 16080]
At Node :1
g(x) :1.0 h(x) :2.0 f(x) :3.0

At Node :2
g(x) :5.0 h(x) :2.0 f(x) :7.0

At Node :3
g(x) :2.0 h(x) :2.0 f(x) :4.0

At Node :4
g(x) :8.0 h(x) :1.0 f(x) :9.0

At Node :2
g(x) :3.0 h(x) :2.0 f(x) :5.0

At Node :4
g(x) :8.0 h(x) :1.0 f(x) :9.0

At Node :5
g(x) :7.0 h(x) :1.0 f(x) :8.0

At Node :4
g(x) :8.0 h(x) :1.0 f(x) :9.0

At Node :6
g(x) :10.0 h(x) :6.0 f(x) :16.0

At Node :6
g(x) :10.0 h(x) :6.0 f(x) :16.0

Best path to node 6 is
0 3 2 5 6
with f(x) :16.0

```