Università degli Studi di Bologna

Corso di Laurea in Informatica
Esercitazione scritta di
ALGORITMI E STRUTTURE DATI - MODULO 2
19/06/15

Si consideri il seguente frammento di codice Java:

```
public abstract class ShortestPath<T extends Comparable<? super T>> {
    PriorityQueue<Node<T>> q;

    static int[] path(Graph<T> g, Node<T> r) {
        int[] d = new int[g.numVertex()];

        for (Node<T> node : g.v())
                d[g.id(node)] = -1; // -1 encodes infinity

        d[g.id(r)] = 0;
        q.addToPriorityQueueOrChangePriority(r, 0); // r is reachable with
                                                    // cost 0

        while (!q.priorityQueueIsEmpty()) { // more are reachable with cost
                                            // not determined yet
            Node<T> u = q.popFromPriorityQueue(); // the minimal cost of u is
                                                  // determined

            for (Node<T> v : g.adj(u)) {
                int distanceViaU = d[g.id(u)] + g.w(u, v);
                if (d[g.id(v)] == -1 || distanceViaU  < d[g.id(v)]) {
                    d[g.id(v)] = distanceViaU;
                    q.addToPriorityQueueOrChangePriority(v, d[g.id(v)]);
                }
            }
            for (Node<T> v : g.getNodes())
                System.out.println("Node " ^ g.id(v) ^ " costs " ^ d[g.id(v)]);
        }

        return d;
    }

    public static void main(String[] args) {
        Graph<Integer> g = new Graph<Integer>;
        Note<Integer> l = new Node<Integer>[6];
        for (int i=0; i<6; i++) {
                l[i] = new Node<Integer>(i);
                g.insertNode(l[i]);
```

```
        }

        g.insertEdge(l[0], l[1], 1);
        g.insertEdge(l[0], l[2], 2);

        g.insertEdge(l[1], l[4], 1);

        g.insertEdge(l[2], l[1], 3);
        g.insertEdge(l[2], l[3], 3);

        g.insertEdge(l[3], l[5], 2);

        g.insertEdge(l[4], l[3], 2);
        g.insertEdge(l[4], l[5], 5);

        path(g, l[0]);
    }
}
```

Si risponda alle seguenti domande:

1. Si mostri l'output del programma.
2. Si annoti il codice in pseudo-JML specificando:
   (a) Le pre e post-condizioni del metodo path
   (b) Il variante del while
   Suggerimento: che caratteristica hanno i nodi che possono ancora entrare nella priority queue? E quanti sono?
   (c) L'invariante del while
   Suggerimento: che ruolo hanno i nodi con un costo $! = -1$ che non sono nella priority-queue? E quelli con un costo $! = -1$ nella priority queue?
   (d) L'invariante del primo for dentro al while