

Resumen de algoritmos para maratones de programación

Diego Alejandro Martínez - Manuel Felipe Pineda

1 de octubre de 2012

Índice

1. Plantilla

2. Grafos

2.1. Dijkstra	3
2.2. Bellman-Ford	3
2.3. Floyd-Warshall	3
2.4. Johnson	3
2.5. Minimum Spanning Tree: Kruskal	3
2.6. Minimum Spanning Tree: Prim	3
2.7. Breadth First Search	3
2.8. Depth First Search	3
2.9. Strongly Connected Components	3
2.10. Puntos de articulación	3
2.11. 2-SAT	3
2.12. Maximum bipartite matching	3
2.13. Flujo Máximo	3
2.14. Lowest Common Ancestor: TarjanOLCA	3

3. Matemáticas

4. Geometría

5. Strings

6. Estructuras de Datos

7. Hashing

8. Miseláneo

1. Plantilla

```
1 import java.io.BufferedOutputStream;
import java.io.BufferedReader;
3 import java.io.InputStreamReader;
import java.io.PrintStream;
3 import java.util.StringTokenizer;

public class Template {

    //Scanner creado por Santiago Gutierrez
    static class Scanner {

        BufferedReader br;
        StringTokenizer st;

        public Scanner() {
            System.setOut(new PrintStream(new \
                BufferedOutputStream(System.out), true));
            br = new BufferedReader(new InputStreamReader(System.in));
        }

        public String next() {

            while (st == null || !st.hasMoreTokens()) {
                try {
                    st = new StringTokenizer(br.readLine());
                } catch (Exception e) {
                    throw new RuntimeException();
                }
            }
            return st.nextToken();
        }
    }
}
```

```

    }

    public int nextInt() {
        return Integer.parseInt(next());
    }

    public double nextDouble() {
        return Double.parseDouble(next());
    }

    public String nextLine() {
        st = null;
        try {
            return br.readLine();
        } catch (Exception e) {
            throw new RuntimeException();
        }
    }

    public boolean endLine() {
        try {
            String next = br.readLine();
            while (next != null && next.trim().isEmpty()) {
                next = br.readLine();
            }
            if (next == null) {
                return true;
            }
            st = new StringTokenizer(next);
            return st.hasMoreTokens();
        } catch (Exception e) {
            throw new RuntimeException();
        }
    }
}

public static void main(String []args){
    Scanner sc = new Scanner();
    String cadena = sc.next();
    int integer = sc.nextInt();
    double ddouble = sc.nextDouble();

```

```

        //imprime en una línea
        System.out.println(cadena);
        //imprime con formato (Como en c)
        System.out.printf("%d\n", integer);
        //imprime sin salto de línea
        System.out.print(ddouble);
    }
}

```

2. Grafos

- 2.1. Dijkstra
- 2.2. Bellman-Ford
- 2.3. Floyd-Warshall
- 2.4. Johnson
- 2.5. Minimum Spanning Tree: Kruskal
- 2.6. Minimum Spanning Tree: Prim
- 2.7. Breadth First Search
- 2.8. Depth First Search
- 2.9. Strongly Connected Components
- 2.10. Puntos de articulación
- 2.11. 2-SAT
- 2.12. Maximum bipartite matching
- 2.13. Flujo Máximo
- 2.14. Lowest Common Ancestor: TarjanOLCA

3. Matemáticas

4. Geometría

5. Strings

6. Estructuras de Datos

7. Hashing

8. Miseláneo