

Resumen de algoritmos para maratones de programación

Diego Alejandro Martínez - Manuel Felipe Pineda

30 de septiembre de 2012

Índice

1. Plantilla

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. Misceláneo

1. Plantilla

```
1  #include <cstdio>
   #include <cmath>
2  #include <iostream>
   #include <string>
   #include <vector>
   #include <queue>
   #include <stack>
   #include <list>
   #include <map>

   using namespace std;

   #define rep(i,a,b) for(int i=a;i<b;i++)
   #define REP(i,n) rep(i,0,n)
   #define foreach(x, v) for (typeof (v).begin() x = (v).begin(); \
x != (v).end(); ++x)
   #define D(x) cout << #x " = " << x << endl;

   typedef long long int lld;
   typedef pair<int,int> pii;
   typedef vector<int> vi;
   typedef vector<pii> vpri;

2  int main(){

2      return 0;

2  }

2  .....
```

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. Miseláneo