

# Resumen de algoritmos para maratones de programación

Diego Alejandro Martínez - Manuel Felipe Pineda

2 de octubre de 2012

## Índice

<b>1. Plantilla</b>	<b>2</b>	4.7. Intersección de dos rectas . . . . .	3
<b>2. Grafos</b>	<b>3</b>	4.8. Intersección de dos segmentos . . . . .	3
2.1. Dijkstra . . . . .	3	4.9. Determinar si dos segmentos se intersectan o no . . . . .	3
2.2. Bellman-Ford . . . . .	3	4.10. Centro del círculo que pasa por tres puntos . . . . .	3
2.3. Floyd-Warshall . . . . .	3	4.11. Par de puntos más cercanos . . . . .	3
2.4. Johnson . . . . .	3	4.12. Par de puntos más alejados . . . . .	3
2.5. Minimum Spanning Tree: Kruskal . . . . .	3	4.13. Área de un polígono . . . . .	3
2.6. Minimum Spanning Tree: Prim . . . . .	3	4.14. Convexhull . . . . .	3
2.7. Breadth First Search . . . . .	3	<b>5. Strings</b>	<b>3</b>
2.8. Depth First Search . . . . .	3	5.1. Knuth-Morris-Pratt KMP . . . . .	3
2.9. Strongly Connected Components . . . . .	3	5.2. Aho-Corasick . . . . .	3
2.10. Puntos de articulación . . . . .	3	5.3. Suffix Array . . . . .	3
2.11. 2-SAT . . . . .	3	<b>6. Teoría de Juegos</b>	<b>3</b>
2.12. Maximum bipartite matching . . . . .	3	<b>7. Estructuras de Datos</b>	<b>3</b>
2.13. Flujo Máximo . . . . .	3	7.1. Prefix Tree - Trie . . . . .	3
2.14. Lowest Common Ancestor: TarjanOLCA . . . . .	3	7.2. Fenwick Tree . . . . .	3
<b>3. Matemáticas</b>	<b>3</b>	7.3. Interval Tree . . . . .	3
3.1. Algoritmo de Euclides y extendido . . . . .	3	<b>8. Hashing</b>	<b>3</b>
3.2. Potencia modular . . . . .	3	8.1. FNV Hash . . . . .	3
3.3. Criba de Eratóstenes . . . . .	3	8.2. JSW Hash . . . . .	3
<b>4. Geometría</b>	<b>3</b>	<b>9. Misceláneo</b>	<b>3</b>
4.1. Utilidades Geometría . . . . .	3	9.1. Bitwise operations . . . . .	3
4.2. Distancia mínima: Punto-Segmento . . . . .	3		
4.3. Distancia mínima: Punto-Recta . . . . .	3		
4.4. Determinar si un polígono es convexo . . . . .	3		
4.5. Determinar si un punto está dentro de un polígono convexo .	3		
4.6. Determinar si un punto está dentro de un polígono cualquiera	3		

# 1. Plantilla

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

using namespace std;

#define all(x) x.begin(),x.end()
#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> vpii;

int main(){

    return 0;
}
```

.....

## 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

- 3.1. Algoritmo de Euclides y extendido
- 3.2. Potencia modular
- 3.3. Criba de Eratóstenes

## 4. Geometría

- 4.1. Utilidades Geometría
- 4.2. Distancia mínima: Punto-Segmento
- 4.3. Distancia mínima: Punto-Recta
- 4.4. Determinar si un polígono es convexo
- 4.5. Determinar si un punto está dentro de un polígono convexo
- 4.6. Determinar si un punto está dentro de un polígono