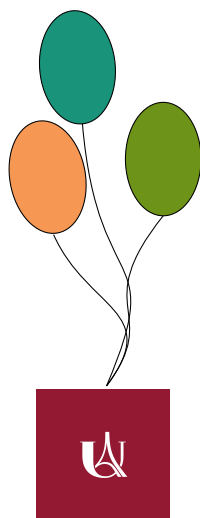


SWERC NOTEBOOK

ÉQUIPE NOM

Nom 1, Nom 2, Nom 3



Université Paris Cité
UFR de Mathématiques et Informatique

Contents

1	Setup	1
2	Mathématiques	2
2.1	Analyse	2
2.2	Combinatoire	2
2.3	Distances	2
2.4	Géométrie	2
2.5	Optimisation et approximations	2
2.6	Probabilités et statistiques	2
2.7	Trigonométrie	2
3	Algorithmes sur les graphes	2
3.1	Graphes et utilitaires	2
3.2	Parcours de graphes	2
3.2.1	Depth First Search	2
3.2.2	Breadth First Search	2
3.2.3	Composante connexes dans un graphe orienté – Algorithme de Tarjan	2
3.2.4	MST dans un graphe orienté – Algorithme de Prim	2
3.2.5	Chemin le plus courts dans un graphe pondéré non orienté – Dijkstra	2
3.3	Algorithmes sur les flots	2

1 Setup

Les paramètres de compilations sont les suivants :

```
g++ -o -Wall -O2 NomExec Nom.cpp
```

Paramètres de base du programme :

```

1  #include <bits/stdc++.h>
2  #define LL long long
3  #define PB push_back
4  #define vi vector<int>
5  #define F(i,a,x) for(i = a; i<x;++i)
6  #define FE(a,x) for(int i = a; i<x; ++i)
7  using namespace std;
8  void solve(){
9      //si plusieurs elements
10     int n; cin >> n;
11     FE(0,n){
12         //affectations
13     }
14     /*code*/
15 }
16 int main(void){
17     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0); //PEUT FAIRE BUGUER ! (
        potentiellement ios_base::)
18     int t; cin >> t;
19     while(t--){
20         solve(),
21     }
22     return 0;
23 }
```

2 Mathématiques

2.1 Analyse

2.2 Combinatoire

2.3 Distances

2.4 Géométrie

2.5 Optimisation et approximations

2.6 Probabilités et statistiques

2.7 Trigonométrie

3 Algorithmes sur les graphes

3.1 Graphes et utilitaires

```

1 typedef unordered_map<int,vector<int> > graph;
2 int visited[MAX_SIZE];
3 queue<int> qe;

```

3.2 Parcours de graphes

3.2.1 Depth First Search

Complexité moyenne : $\mathcal{O}(|V| + |E|)$ Complexité maximale : $\mathcal{O}(|V| \cdot |E|)$

```

1 void DFS(graph G, int x){
2     visited[x] = 1;
3     for (auto const& [key,value] : graph){
4         if (visited[value] != 1){
5             DFS(G,value);
6         }
7     }
8 }

```

3.2.2 Breadth First Search

Complexité moyenne : $\mathcal{O}(|V| + |E|)$ Complexité maximale : $\mathcal{O}(|V| \cdot |E|)$

```

1 void BFS(graph G, int x){
2     qe.push_back(x);
3     visited[x] = 1;
4     while(!qe.empty()){
5         int v = qe.front();
6         qe.pop_front();
7         for(auto value : G.at(v)){
8             if (visited[value] != 1){
9                 qe.push_back(value);
10                visited[value] = 1;
11            }
12        }
13    }
14 }

```

3.2.3 Composante connexes dans un graphe orienté – Algorithme de Tarjan

3.2.4 MST dans un graphe orienté – Algorithme de Prim

3.2.5 Chemin le plus courts dans un graphe pondéré non orienté – Dijkstra

3.3 Algorithmes sur les flots