

## Contents

1	語法
1.1	c++ . . . . .
1.2	python . . . . .
2	Graph
2.1	Bellman-Ford . . . . .
2.2	Dijkstra . . . . .
2.3	Floyd-Warshall . . . . .
2.4	SPFA . . . . .
2.5	SPFA . . . . .
3	Other
3.1	thm . . . . .

## 1 語法

### 1.1 c++

```

1 // c++ code
2 #include <bits/stdc++.h>
3 lower_bound(a, a + n, k); //最左邊 ≥ k 的位置
4 upper_bound(a, a + n, k); //最左邊 > k 的位置
5 upper_bound(a, a + n, k) - 1; //最右邊 ≤ k 的位置
6 lower_bound(a, a + n, k) - 1; //最右邊 < k 的位置
7 [lower_bound, upper_bound) //等於 k 的範圍
8 equal_range(a, a+n, k);
9
10 // 從小到大
11 priority_queue<int, vector<int>, greater<int>> > pq
12
13 insert(it,x)//向vector的任意迭代器it處插入一個元素x
14 erase(it)//刪除迭代器為it處的元素, erase(first,last)
15 //刪除一個區間[first,last)內的所有元素, 時間複雜度均為O(N)
16
17 set
18 insert(x) //將x插入set中 O(log(n))
19 count(x) //回傳x是否存在於set中() O(log(n))
20 erase(x) //刪除在set中的x O(log(n))
21 clear() //刪除set中所有元素 O(n)
22 empty() //回傳是否為空 O(1)
23 size() //回傳共有幾個元素 O(1)
24
25 map
26 insert(x) //將x這個pair插入map中 O(log(n))
27 count(x) //回傳x這個key是否在map中 O(log(n))
28 erase(x) //刪除在map中key為x的 O(log(n))
29
30 #include <bits/stdc++.h>
31 using namespace std;
32
33 int main(){
34     set<int> s;
35     for(int i = 0; i < 10; i++){
36         s.insert(i);
37     }
38     cout << "lower bound: " << *s.lower_bound(5) <<
39         '\n'; // 5
40     cout << "upper bound: " << *s.upper_bound(5) <<
41         '\n'; // 6
42     if(s.lower_bound(20) == s.end()){
43         cout << "all elements are less than 20\n";
44     }
45 }
```

### 1.2 python

```

1 sorted((4,1,9,6),reverse=True)
2 fruits = ['apple', 'watermelon', 'pear', 'banana']
3 a = sorted(fruits, key = lambda x : len(x))
4 print(a)
5 # 輸出: ['pear', 'apple', 'banana', 'watermelon']
6 divmod(a,b)
7 把除數和餘數運算結果結合起來,
8 返回一個包含商和餘數的元組(a // b, a % b)
9
10 pow(base, exp[, mod])
11 >>> pow(38, -1, mod=97)
12 23
13 >>> 23 * 38 % 97 == 1
14 True
15
16 eof 寫法
17 try:
18     while True:
19         s = input()
20 except EOFError:
21     pass
22
23 eval(expression, globals=None, locals=None)
24
25 list(map(int, input().split()))
26 L.append(r)
27 my_list = ['This', 'is', 'a', 'string', 'in',
28            'Python']
29 my_string = " ".join(my_list)
30 #This is a string in Python
31 test = [[0 for j in range(m)] for i in range(n)]
```

## 2 Graph

### 2.1 Bellman-Ford

```

1 #include<iostream>
2 using namespace std;
3 const int INF = 1e9;
4 const int MAXN = 1000;
5 const int MAXM = 1000;
6 struct Edge {
7     int u;
8     int v;
9     int w;
10 };
11
12 int n, m;
13 Edge edges[MAXN];
14 int dis[MAXN];
15
16 // s是起點
17 bool bellman(int s) {
18     for (int i = 0; i < n; i++) {
19         dis[i] = INF;
20     }
21     dis[s] = 0;
22     bool relax;
23     // 做 n 輪
24     for (int i = 0; i < n; i++) {
25         relax = false;
26         for (int j = 0; j < m; j++) {
27             int u = edges[j].u;
28             int v = edges[j].v;
29             int w = edges[j].w;
30             if (dis[u] == INF) {
31                 continue;
32             }
33             if (dis[v] > dis[u] + w) {
34                 dis[v] = dis[u] + w;
35                 relax = true;
36             }
37         }
38     }
39 }
```

```

38     if (!relax) {
39         break;
40     }
41 }
42 return relax;
43 }
44
45
46 int main(){
47
48 }

```

## 2.2 Dijkstra

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define M 100005
4 #define INF 1e9
5 struct Edge{
6     int v, w;
7     Edge(int a, int b):v(a), w(b){};
8 };
9 struct node{
10     int u, dis;
11     node(){};
12     node(int a, int b):u(a), dis(b){};
13     bool operator<(const node &r)const{
14         return dis > r.dis;
15     }
16 };
17 int dis[M]; //距離
18 vector<Edge> G[M];
19 void init(){
20     fill(dis, dis+M, INF);
21     for(int i = 0; i < M; i++){
22         G[i].clear();
23     }
24 }
25 void dijkstra(int start){
26     dis[start] = 0;
27     priority_queue<node> pq;
28     pq.push(node(start, 0));
29     while(!pq.empty()){
30         node now = pq.top();
31         pq.pop();
32         if(now.dis > dis[now.u]) continue;
33         for(Edge i : G[now.u]){
34             if(dis[i.v] > now.dis + i.w){
35                 dis[i.v] = now.dis + i.w;
36                 pq.push(node(i.v, dis[i.v]));
37                 // printf("push(%d, %d)\n", i.v,
38                     //         dis[i.v]);
39             }
40         }
41     }
42 }
43 int main(){
44     int point, side;
45     cin >> point >> side;
46     init();
47     for(int i = 0; i < side; i++){
48         int s, t, w;
49         cin >> s >> t >> w;
50         G[s].push_back(Edge(t, w));
51         G[t].push_back(Edge(s, w));
52     }
53     dijkstra(1);
54     for(int i = 2; i <= point; i++){
55         cout << dis[i] << '\n';
56     }
57 }
58 }

```

## 2.3 Floyd-Warshall

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define M 1005
4 #define INF 1e9
5
6 int dis[M][M];
7 // int G[M][M];
8 void init(int n){
9     for(int i = 0; i <= n; i++){
10         for(int j = 0; j <= n; j++){
11             dis[i][j] = INF;
12             if(i == j) dis[i][j] = 0;
13         }
14     }
15 }
16 void Floyd(int n){
17     for(int k = 1; k <= n; k++){
18         for(int i = 1; i <= n; i++){
19             for(int j = 1; j <= i; j++){
20                 dis[i][j] = min(dis[i][k]+dis[k][j],
21                                 dis[i][j]);
22             }
23         }
24     }
25 }
26 void printarr(int r, int c){
27     for(int i = 1; i <= r; i++){
28         for(int j = 1; j <= c; j++){
29             if(dis[i][j] == INF) cout << "INF ";
30             else cout << dis[i][j] << ' ';
31         }
32         cout << '\n';
33     }
34 }
35 int main(){
36     int point, side;
37     cin >> point >> side;
38     init(point);
39     for(int i = 0; i < side; i++){
40         int s, t, w;
41         cin >> s >> t >> w;
42         dis[s][t] = w;
43         dis[t][s] = w;
44     }
45     Floyd(point);
46     int Cas;
47     cin >> Cas;
48     while(Cas--){
49         int i, j;
50         cin >> i >> j;
51         cout << dis[i][j] << '\n';
52     }
53     // printarr(point, point);
54 }

```

## 2.4 SPFA

```

1 const int INF = 1e9;
2 const int MAXN = 1000;
3 struct Edge {
4     int v;
5     int w;
6 };
7 int n, m;
8 vector<Edge> G[MAXN]; //向量記圖
9 int dis[MAXN];
10 void SPFA(int s) {
11     // 記錄目前的點是否在 queue 中
12     bool inq[n];
13     for (int i = 0; i < n; i++) {
14         dis[i] = INF;

```

```

15     inq[i] = false;
16 }
17 dis[s] = 0;
18 inq[s] = true;
19 queue<int> q;
20 q.push(s);
21 while (!q.empty()) {
22     int u = q.front();
23     q.pop();
24     inq[u] = false;
25     for (Edge e : G[u]) {
26         if (dis[e.v] > dis[u] + e.w) {
27             dis[e.v] = dis[u] + e.w;
28             if (!inq[e.v]) {
29                 inq[e.v] = true;
30                 q.push(e.v);
31             }
32         }
33     }
34 }
35 }

```

## 2.5 SPFA

```

1  const int INF = 1e9;
2  const int MAXN = 1000;
3  struct Edge {
4      int v;
5      int w;
6  };
7  int n, m;
8  vector<Edge> G[MAXN]; //向量記圖
9  int dis[MAXN];
10 void SPFA(int s) {
11     // 記錄目前的點是否在 queue 中
12     bool inq[n];
13     for (int i = 0; i < n; i++) {
14         dis[i] = INF;
15         inq[i] = false;
16     }
17     dis[s] = 0;
18     inq[s] = true;
19     queue<int> q;
20     q.push(s);
21     while (!q.empty()) {
22         int u = q.front();
23         q.pop();
24         inq[u] = false;
25         for (Edge e : G[u]) {
26             if (dis[e.v] > dis[u] + e.w) {
27                 dis[e.v] = dis[u] + e.w;
28                 if (!inq[e.v]) {
29                     inq[e.v] = true;
30                     q.push(e.v);
31                 }
32             }
33         }
34     }
35 }

```

## 3 Other

### 3.1 thm