Graph Algorighm - Elementary Graph Algorithm

實作

可以用 Adjacency List 或 Adjacency Matrix 表示。空間複雜度各是 $O(V+E)$ 與 $O(V^2)$。雖然 Adjacenct Matrix 很肥，但是有可以任意存取某個邊的好處，而且邊有權重時不用另外實作邊的資料結構。

Adjacency Matrix

概念上來說利用一個矩陣:

$$w_{ij} = \begin{cases} 1 & \text{if } (\mathrm{u_i}, \mathrm{u_j}) \in E \\ 0 & \text{otherwise} \end{cases}$$

實作上來說像是:

```
#define N_MAX 20
int W[N_MAX][N_MAX];
int main()
{
    int i, j, weight;
    while(scanf("%d%d%d", &i, &j, &weight) != EOF) {
        W[i][j] = 1;
    }
    return 0;
}
```

雖然感覺很肥，很多 Adjacency List 中 $O(V+E)$ 的演算法這邊會變成 $O(V)$，但是如果邊沒有權重的話，可以用 bit field 壓縮空間。而且如果圖很稠密，那麼 Adjacency List 跟 Adjacency Matrix 的複雜度相去不遠，但實作上相較之下較便利。

Adjacency List

概念上來說是一個 List 的 List：

$$(u,v) \in E \iff v \in Adj[u]$$

實作上可以用 `vector` 陣列，或是 `vector<<vector>>`：

```
/* include stuffs */

# define N_MAX 1000
vector <int> W[N_MAX];

int main()
{
    int i, j;
    while(scanf("%d%d%d", &i, &j) != EOF) {
        W[i].push_back(j);
    }
    return 0;
}
```

然後發現如果有邊的話不知道要怎麼做。自己刻一個資料結構，或是開一個陣列存權重：

```
struct edge{
    int i;
    int j;
    int w;
    edge(int _i, int _j, int _w):i(_i),j(_j),w(_w){}
};

```

```
vector <edge> W[N_MAX];

int main()
{
    int i, j, w;
    while(scanf("%d%d%d", &i, &j, &w) != EOF) {
        W[i].push_back(edge(i, j, w));
    }
    return 0;
}
```

Trasversal

DFS

在 CLRS 提供的 pseudo code 長這樣:

```
DFS_VISIT(G, u)
time = time + 1
u.d = time //紀錄進入 u 的時間
u.color = GRAY
for all v in G.adj[u]
    if v.color == WHITE
        v.pi = u //紀錄 v 的父節點是 u
        DFS_VISIT(G, u)
u.color = BLACK
time = time + 1
u.f = time //u pop 出來的時間


DFS(G)
for each vertex u in G.V
    u.color = WHITE
    u.pi = NIL
time = 0
for all vertex u in G.V
    if u.color == WHITE
        DFS_VISIT(u)
```

實際上使用的實作可能像這樣:

```
#define N_MAX 1000

char vis[N_MAX];
vector<int> W[N_MAX];

/* DFS_VISIT() function */
void dfs(int n, vector<int> *G)
{
    vis[n] = 1;
    /* GRAY area for u after this line */
    for (auto &i in G[n]) {
        if (!vis[i]) {
            /* WHITE area for i*/
            dfs(i, G);
        }
    }
    /* BLACK area for u after this line */
}

int main()
{
    /* skip input */
    /* The DFS() */
    fill(vis, vis + NODE_NUM);
    for(int i = 0; i < N_MAX; i++) {
        if (!vis[i])
            dfs(i, &W);
    }

}
```

在 CLRS 中的 DFS 比較高級一點，會把進入時間跟結束時間一併紀錄。可以多新增`int d[N_MAX]`, `int pi[N_MAX]` , `int f[N_MAX]` 等等分別紀錄第 i 個點的結束時間，或是直接當作 `vis[i]` 的根據。

BFS

CLRS 給的：

```
BFS(G, V)
    for all vertex u in G.V - {s}
        u.color = WHITE
        u.d = INF
        u.pi = NIL
    s.color = GRAY
    s.d = 0
    s.pi = NIL
    Q = Ø
    ENQUEUE(Q, s)
    while Q != Ø
        u = DEQUEUE(Q);
        for all v in G.adj[u]
            if v.color = WHITE
                v.color = GRAY
                u.d = u.d. + 1
                v.pi = u
                ENQUEUE(Q, v)
        u.color = BLACK
RETURN
```

```
void bfs(int start)
    fill(vis, vis + NODE_NUM, 0);
    queue<int> q;
    q.push(start);
    vis[start] = 1; /* GRAY starting point */
    while(!q.empty()) {
        int cur = q.front();
        q.pop();
        for (&i in W[cur])
            if (!vis[i]) {
                vis[i] = 1;
                /* GRAY AREA */
                q.push(i);
            }
        /* BLACK AREA */
    }
}
```

類似地，在 CLRS 的程式碼中，多紀錄了進入與離開的 time stamp，以及遍歷時的父節點。在實作上可以跟 DFS 時多開幾個紀錄這些資訊的陣列代替。

另外，CLRS 用白灰黑 3 種狀態表示點在 BFS 中的狀態，但實際上裡面有提到「... In fact, as Exercise 22.2-3 shows, we would get the same result even if we did not distinguish between gray and black vertices.」，所以在程式的實作中就不區分灰黑。實際上如果需要也可以用 `vis[i]` 的值代替不同狀態。

Shortest Path

這邊討論的最短路徑是「通過最少數目的邊，到達另外一點」，是以「邊的數目」決定路徑長度，尚沒有權重的概念。

Definition (Shortest-Path Distance)

$G = (V, E)$ 是個圖，$Shortest-Path Distance 定義為：

$$\delta(u, v) : V \times V \to \mathbb{N} \cup \{0\},$$

$$\delta(u, v) = \begin{cases} \infty & \text{if u, v not reachable} \\ \min\{|p| - 1 : p \text{ is a path from u to v }\} & \text{otherwise} \end{cases}$$

Definition (Shortest Path)

$$p \text{ is a shortest past from u to v } \iff \begin{cases} p \text{ is a path from u to v}, \ \mathit{and} \\ \delta(\text{u}, \text{v}) = |p| - 1 \end{cases}$$

Lemma

$G = (V, E)$ 是個圖（有向或無向），則：

$$\forall s \in V. \forall (u, v) \in E. \delta(s, v) \leq \delta(s, u) + 1$$

1. 假定 s, u not reachable，則 $\delta(s,u) = \infty$，命題成立。
2. 假定 s, u reachable，則存在一條長度 $\delta(s,u)$ 的最短路徑 $p' = \langle s \ldots u \rangle$，則路徑 $p = p' \cup \{v\}$ 是一條 u, v 間的路徑（未必是最短），長度為 $\delta(s,u)+1$。因為任何路徑長度都 $\leq$ 最短路徑長度，故：

$$\delta(s,v) \leq \delta(s,u) + 1$$

Online Judge

> *OJ 習題*
>
> *UVA : 336,352,383,532,567,571,601,705, 762,10004,10009,10474, 10505,10592,10603, 10946, 11624*
>
> *POJ : 1129,1154,1416,1606,1753,1915,1979,2243*