# BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

## I. Thuật toán Dijkstra

Áp dụng khi trọng số của tất cả các cạnh không âm:

```
const INF 2000000000;

void Dijkstra (int xp, vector<int> &d, vector<int> &p) {
    d.resize (n + 1, INF);
    p.resize (n + 1, 0);

    set<pair<int, int> > q;
    d[xp] = 0;
    q.insert ({d[xp], xp});
    while (!q.empty()) {
        int u = q.begin()->second;
        q.erase (q.begin());
        for (pair<int,int> x : adj[u]) {
            int v = x.first, L = x.second;
            if (d[u] + L < d[v]) {
                if (d[v] < INF) q.erase ({d[v], v});
                d[v] = d[u] + L;
                p[v] = u;
                q.insert ({d[v], v});
            }
        }
    }
}
```

Output:

- d[u]=khoảng cách từ xp đến đỉnh u
- p[1..n] - mảng để tìm một đường đi ngắn nhất

## II. Thuật toán Bellman-Ford

Áp dụng khi cạnh của đồ thị âm. Hàm dưới đây trả về false nếu như phát hiện được chu trình âm, ngược lại trả về true. Khi trả về true thì ý nghĩa các vector d, p giống như trong thuật toán Dijkstra:

```
const INF 2000000000;

bool Bellman_Ford (int xp, vector<int> &d, vector<int> &p) {
    d.resize (n + 1, INF);
    p.resize (n + 1, 0);
    vector<bool> inqueue (n + 1, false);
    vector<int> cnt (n + 1, 0);
    queue<int> q;
    d[xp] = 0;
    q.push (xp);
    inqueue[xp] = true;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        inqueue[u] = false;
        for (pair<int, int> x : adj[u]) {
```

```
            int v = x.first, L = x.second;
            if (d[u] + L < d[v]) {
                d[v] = d[u] + L;
                p[v] = u;
                if (!inqueue[v]) {
                    inqueue[v] = true;
                    ++cnt[v];
                    q.push (v);
                    if (cnt[v] > n)
                        return false;
                }
            }
        }
    }
    return true;
}
```

## III. 0-1 BFS

Thuật toán này sử dụng khi trọng số của cạnh hoặc là 0 hoặc là 1. Khi đó ta có thể sử dụng hàng đợi hai đầu trong C++ để tăng tốc thuật toán (Thay cho Dijkstra truyền thống):

```
conts INF 2000000000;

void BFS01 (int xp, vector<int> &d, vector<int> &p) {
    d.resize (n + 1, INF);
    p.resize (n + 1, 0);
    d[xp] = 0;
    deque<int> q;
    q.push_front (xp);
    while (!q.empty()) {
        int u = q.front();
        q.pop_front();
        for (pair<int, int> x : adj[u]) {
            int v = x.first, L = x.second;
            if (d[u] + L < d[v]) {
                d[v] = d[u] + L;
                p[v] = u;
                if (L == 1)
                    q.push_back (v);
                else
                    q.push_front (v);
            }
        }
    }
}
```

## IV. Thuật toán Floy - Bellman

**Bài toán:** Cho đồ thị có trọng số được biểu diễn bằng ma trận khoảng cách  kích thước $n \times n$ (a[i,j]=độ dài cung trực tiếp nhỏ nhất nối từ i đến j - giá trị bằng INF nếu như không có cạnh trực tiếp nối). Hãy xây dựng ma trận dp[i,j]=độ dài đường đi ngắn nhất từ đỉnh i đến đỉnh j:

```
contst INF 1000000000

void FloyBellman(vector<vector<int> > &f, vector<vector<int> > &p) {
    f.resize(n+1,vector<int>(n+1,INF));
```

```
    p.resize(n+1,vector<int>(n+1,0));
    for(int i=1;i<=n;++i)
    for(int j=1;j<=n;++j) f[i][j]=a[i][j];
    for(int i=1;i<=n;++i) f[i][i]=INF;
    for(int k=1;k<=n;++k)
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
            if (f[i][j]<f[i][k]+f[k][j]) {
                f[i][j]=f[i][k]+f[k][j];
                p[i][j]=k;
            }
}
```

# V. Các ứng dụng của bài toán tìm đường đi ngắn nhất

## 5.1. Tìm một đường đi từ xp đến kt

```
vector<int> path;
for (v = kt; v != 0; v = Prev[v])
    path.push_back (v);
reverse (path.begin(), path.end());
```

## 5.2 Đếm số đường đi ngắn nhất từ xp đến một đỉnh

Giả sử các cạnh của đồ thị là số dương. Ta xây dựng đồ thị con với cạnh (u,v) là cạnh cũ của đồ thị có thêm điều kiện d[v]=d[u]+L(u,v). Đồ thị này là DAG và ta có thể thực hiện qui hoạch động trên đồ thị này để đếm số đường đi ngắn nhất.

Đặt dp[u] là số đường đi ngắn nhất từ xp đến u. vector dp có thể được tính như sau:

```
vector<long long> dp (n + 1, 0);
vector<pair<int, int> > Topo;
for (int u = 1; u <= n; ++u)
    Topo.push_back ({kc[u], u});
sort (Topo.begin(), Topo.end());
for (pair<int, int> x : Topo) {
    int u = x.second;
    for (pair<int, int> e : adj[u]) {
        int v = e.first, L = e.second;
        if (d[v] == d[u] + L)
            dp[v] = dp[v] + dp[u];
    }
}
```

## 5.3 Tìm đường đi có cạnh lớn nhất là nhỏ nhất (sử dụng thuật toán Dijkstra)

```
const INF 2000000000;

void DijMaxMin (int xp, vector<int> &d, vector<int> &p) {
    d.resize (n + 1, INF);
    p.resize (n + 1, 0);

    set<pair<int, int> > q;
    d[xp] = 0;
    q.insert ({d[xp], xp});
    while (!q.empty()) {
        int u = q.begin()->second;
        q.erase (q.begin());
```

```
        for (pair<int,int> x : adj[u]) {
            int v = x.first, L = x.second;
            if (max(d[u], L) < d[v]) {
                if (d[v] < INF) q.erase ({d[v], v});
                d[v] = max(d[u], L);
                p[v] = u;
                q.insert ({d[v], v});
            }
        }
    }
}
```

**Output:**

- **d[u]=cạnh lớn nhất trên đường đi tối ưu đến đỉnh u**
- **p[1..n] - mảng để tìm một đường đi tối ưu**