5.8 節末問題 5.8 的解答

問題 5.8.1

如 5.8.3 項所述，在表示年齡差關係的圖中，令頂點 1 到頂點 i 的最短路徑長度 $dist[i]$，則人 i 的年齡最大$\min(dist[i], 120)$。

因此，對於求出最短路徑長度的程式（程式碼 4.5.3），只變更輸出部分並提出如下程式，即可得到正解。

```
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;

int N, M, A[100009], B[100009];
int dist[100009];
vector<int> G[100009];

int main() {
    // 輸入
    cin >> N >> M;
    for (int i = 1; i <= M; i++) {
        cin >> A[i] >> B[i];
        G[A[i]].pusha_back(B[i]);
        G[B[i]].push_back(A[i]);
    }

    // 廣度優先搜尋的初始化（dist[i]=-1時，為未到達的白色頂點）
    for (int i = 1; i <= N; i++) dist[i] = -1;
    queue<int> Q; // 定義佇列 Q
    Q.push(1); dist[1] = 0; // 將 1 添加到 Q 中（操作 1）

    // 廣度優先搜尋
    while (!Q.empty()) {
        int pos = Q.front(); // 查看 Q 的開頭（操作2）
        Q.pop(); // 取出 Q 的開頭（操作3）
        for (int i = 0; i < (int)G[pos].size(); i++) {
            int nex = G[pos][i];
            if (dist[nex] == -1) {
                dist[nex] = dist[pos] + 1;
                Q.push(nex); // 將 nex 添加到 Q 中（操作 1）
            }
```

```
        }
    }

    // 輸出從頂點 1 到各頂點的最短距離
    // 無法從頂點 1 到達的時候可設為 120 歲
    for (int i = 1; i <= N; i++) {
        if (dist[i] == -1) cout << "120" << endl;
        else cout << min(dist[i], 120) << endl;
    }
    return 0;
}
```

← 從程式碼 4.5.3 變更的部分

※ Python等原始碼請參閱chap5-8.md。

問題 5.8.2

這個問題不容易看出解法，所以讓我們先調查 N 較小的情況吧。

- 當 $N = 2$ 時，以$(P_1, P_2) = (2, 1)$的最大分數為 1
- 當 $N = 3$ 時，以$(P_1, P_2, P_3) = (2, 3, 1)$ 的最大分數為 3
- 當 $N = 4$時，以$(P_1, P_2, P_3, P_4) = (2, 3, 4, 1)$ 的最大分數為 6

（藉由進行全搜尋，用手算也可得知）

$N \geq 5$ 時也一樣，令 $(P_1, P_2, \ldots, P_{N-1}, P_N) = (2, 3, \ldots, N, 1)$，則分數如下（和的公式 → **2.5.10項**）。

$$\sum_{i=1}^{N} (i \bmod P_i) = 1 + 2 + 3 + \cdots + (N-1) + 0 = \boldsymbol{\frac{N(N-1)}{2}}$$

但是，這真的是最大值嗎？答案是肯定的，可以證明如下。

首先，為了簡單起見，證明 $N = 4$ 的最大值為 6。

$$\sum_{i=1}^{4} (i \bmod P_i) = (1 \bmod P_1) + (2 \bmod P_2) + (3 \bmod P_3) + (4 \bmod P_4)$$

雖然分數是計算如上式，但依據 mod 的性質可以說明以下幾點：。

- $1 \bmod P_1$ 為 $P_1 - 1$ 以下
- $2 \bmod P_2$ 為 $P_2 - 1$ 以下
- $3 \bmod P_3$ 為 $P_3 - 1$ 以下
- $4 \bmod P_4$ 為 $P_4 - 1$ 以下

因此，分數是將這些相加的值 $P_1 + P_2 + P_3 + P_4 - 4$ 以下。

但是，(P_1, P_2, P_3, P_4) 是 $(1, 2, 3, 4)$ 的調換排列，因此如下所示：

$$P_1 + P_2 + P_3 + P_4 = 1 + 2 + 3 + 4 = 10$$
$$P_1 + P_2 + P_3 + P_4 - 4 = 1 + 2 + 3 + 4 - 4 = 6$$

可知分數為 6 以下。所以，分數在$(P_1, P_2, P_3, P_4) = (2, 3, 4, 1)$ 時的最大值為 6。

一般情況的證明

可以用和 $N = 4$時相同的方法證明。首先，$i \bmod P_i \leq P_i - 1$ 成立，故分數為 $(P_1 - 1) + (P_2 - 1) + \cdots + (P_N - 1) = P_1 + \cdots + P_N - N$ 以下。

另一方面，$(P_1, P_2, P_3, \ldots, P_N)$ 是 $(1, 2, 3, \ldots, N)$ 的調換排列，因此如下所示：

$$P_1 + P_2 + \cdots + P_N = 1 + 2 + \cdots + N = \frac{N(N+1)}{2}$$
$$P_1 + P_2 + \cdots + P_N - N = \frac{N(N+1)}{2} - N = \frac{N(N-1)}{2}$$

可知分數為 $N(N-1)/2$ 以下。

因此，提出將 $N(N-1)/2$ 輸出的程式即可得到正解。

本問題的限制為 $N \leq 10^9$ 之大，答案可能超過 10^{17}，因此注意使用 `int` 型態等 32 位元整數的話可能會發生溢出。

```cpp
#include <iostream>
using namespace std;

int main() {
    // 輸入
    long long N;
    cin >> N;

    // 輸出
    cout << N * (N - 1) / 2 << endl;
    return 0;
}
```

※ Python等原始碼請參閱 chap5-8.md。

問題 5.8.3

注意：此問題使用4.5.8項「其他代表性的圖演算法」介紹的Dijkstra演算法。初學者無法解決是自然的，不必擔心。

首先，思考如下的加權無向圖。

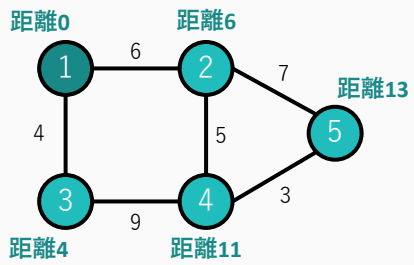
- 有 N 個頂點，從 1 編號到 N 。。
- 有 M 條邊，根據條件$\left|x_{A_i} - x_{B_i}\right| \leq C_i$，添加連接頂點 A_i 和 B_i 的邊，權重為 C_i 。

這時，x_N的最大值為從頂點 1 到頂點 N 的最短路徑長度 $dist[N]$ 。具體例如下，令 $x_i = dist[i]$，則確實滿足 M 個條件式。（本解說不會探討詳細證明，但可使用與無權重圖相同的方法進行證明）

條件

- $|x_1 - x_2| \leq 6$
- $|x_1 - x_3| \leq 4$
- $|x_2 - x_4| \leq 5$
- $|x_2 - x_5| \leq 7$
- $|x_3 - x_4| \leq 9$
- $|x_4 - x_5| \leq 3$

圖



$(x_1, x_2, x_3, x_4, x_5) = (0, 6, 4, 11, 13)$
為最佳解

因此，使用Dijkstra法求出頂點 1 到頂點 N 的最短路徑長度 $dist[N]$，並提出將其輸出的程式如下，即可得到正解。

```
#include <bits/stdc++.h>
using namespace std;

long long N, M;
long long A[500009], B[500009], C[500009];

// 圖
bool used[500009];
long long dist[500009]; // dist[i] 為頂點 1 → 頂點 i 的最短路徑長度
vector<pair<int, long long>> G[500009];
priority_queue<pair<long long, int>, vector<pair<long long, int>>, greater<pair<long
long, int>>> Q;
```

```cpp
// Dijkstra法
void dijkstra() {
    // 陣列的初始化等
    for (int i = 1; i <= N; i++) dist[i] = (1LL << 60);
    for (int i = 1; i <= M; i++) used[i] = false;
    dist[1] = 0;
    Q.push(make_pair(0, 1));

    // 佇列更新
    while (!Q.empty()) {
        int pos = Q.top().second; Q.pop();
        if (used[pos] == true) continue;
        used[pos] = true;
        for (pair<int, int> i : G[pos]) {
            int to = i.first, cost = dist[pos] + i.second;
            if (pos == 0) cost = i.second; // 頂點 0 的時候是例外
            if (dist[to] > cost) {
                dist[to] = cost;
                Q.push(make_pair(dist[to], to));
            }
        }
    }
}

int main() {
    // 輸入、添加圖的邊
    cin >> N >> M;
    for (int i = 1; i <= M; i++) {
        cin >> A[i] >> B[i] >> C[i];
        G[A[i]].push_back(make_pair(B[i], C[i]));
        G[B[i]].push_back(make_pair(A[i], C[i]));
    }

    // Dijkstra法
    dijkstra();

    // 輸出答案
    if (dist[N] == (1LL << 60)) cout << "-1" << endl;
    else cout << dist[N] << endl;
    return 0;
}
```

※ Python等原始碼請參閱 chap5-8.md。