5.8 節末問題 5.8 の解答

問題 5.8.1

5.8.3 項で述べた通り、年齢差の関係を表したグラフの頂点 1 から頂点 i までの最短経路長を $dist[i]$ とすると、人 i の年齢の最大値は $\min(dist[i], 120)$ となります。

したがって、最短経路長を求めるプログラム（コード 4.5.3）の出力部分のみ変更した、以下のようなプログラムを提出すると、正解が得られます。

```
#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() {
    // 入力
    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]);
    }

    // 幅優先探索の初期化（dist[i]=-1 のとき、未到達の白色頂点である）
    for (int i = 1; i <= N; i++) dist[i] = -1;
    queue<int> Q; // キュー Q を定義する
    Q.push(1); dist[1] = 0; // Q に 1 を追加（操作 1）

    // 幅優先探索
    while (!Q.empty()) {
        int pos = Q.front(); // Q の先頭を調べる（操作 2）
        Q.pop(); // Q の先頭を取り出す（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); // Q に nex を追加（操作 1）
            }
```

```
        }
    }

    // 頂点 1 から各頂点までの最短距離を出力
    // 頂点 1 から到達できない場合は 120 歳にできる
    for (int i = 1; i <= N; i++) {
        if (dist[i] == -1) cout << "120" << endl;
        else cout << min(dist[i], 120) << endl;
    }
    return 0;
}
```

← コード 4.3.1 からの変更部分

※ Python などのソースコードは chap5-8.md をご覧ください。

問題 5.8.2

この問題は解法が見えづらいので、N が小さいケースを調べてみましょう。

- $N = 2$ のとき、$(P_1, P_2) = (2, 1)$ で最大スコア 1
- $N = 3$ のとき、$(P_1, P_2, P_3) = (2, 3, 1)$ で最大スコア 3
- $N = 4$ のとき、$(P_1, P_2, P_3, P_4) = (2, 3, 4, 1)$ で最大スコア 6

となります。（全探索をすることで、手計算でも分かります）

$N \geq 5$ の場合も同じように、$(P_1, P_2, \ldots, P_{N-1}, P_N) = (2, 3, \ldots, N, 1)$ とすると、スコアは

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

となります（和の公式 →**2.5.10項**）。しかし、これが本当に最大なのでしょうか。

答えは Yes であり、以下のようにして証明できます。

まずは簡単のため、$N = 4$ の最大値が 6 であることを証明します。スコアは

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

ですが、mod の性質より以下のことがいえます。

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

したがって、スコアはそれらを足した値 $P_1 + P_2 + P_3 + P_4 - 4$ 以下です。

しかし、(P_1, P_2, P_3, P_4) は $(1, 2, 3, 4)$ の並べ替えであるため、

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

となり、スコアが 6 以下であることが分かります。よって、スコアは $(P_1, P_2, P_3, P_4) = (2, 3, 4, 1)$ のとき最大値 6 となります。

一般の場合の証明

$N = 4$ の場合と同じような方法で証明できます。まず、$i \bmod P_i \leq P_i - 1$ が成り立つため、スコアは $(P_1 - 1) + (P_2 - 1) + \cdots + (P_N - 1) = P_1 + \cdots + P_N - N$ 以下であるといえます。

一方、$(P_1, P_2, P_3, \ldots, P_N)$ は $(1, 2, 3, \ldots, N)$ の並べ替えであるため、

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

となり、スコアが $N(N-1)/2$ 以下であることが分かります。

したがって、$N(N-1)/2$ を出力する以下のようなプログラムを提出すると、正解が得られます。

本問題の制約は $N \leq 10^9$ と大きく、答えが 10^{17} を超える可能性があるため、`int` 型などの 32 ビット整数を使うとオーバーフローを起こすことに注意してください。

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

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

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

※ Python などのソースコードは chap5-8.md をご覧ください。

問題 5.8.3

注意：この問題は、4.5.8 項「その他の代表的なグラフアルゴリズム」で紹介されたダイクストラ法を使います。初学者は解けなくて当然ですので、ご安心ください。

まず、以下のような重み付き無向グラフを考えましょう。

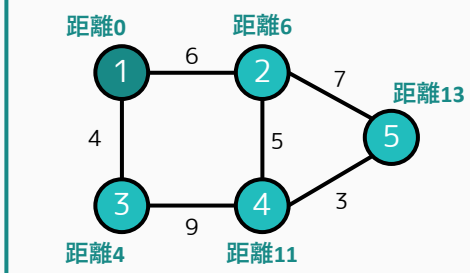
- 頂点は N 個あり、1 から N までの番号が振られている。
- 辺は M 個あり、$|x_{A_i} - x_{B_i}| \leq C_i$ という条件式について、頂点 A_i と B_i を結ぶ重み C_i の辺を追加する。

このとき、x_N の最大値は頂点 1 から頂点 N までの最短経路長 $dist[N]$ となります。具体例は以下の通りであり、$x_i = dist[i]$ とすると確かに M 個の条件式すべてを満たします。（詳しい証明は本解説では扱わないことにしますが、重みなしグラフの場合と同じような方法で証明できます）

条件

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

グラフ



$(x_1, x_2, x_3, x_4, x_5) = (0, 6, 4, 11, 13)$
が最適解

したがって、ダイクストラ法によって頂点 1 から N までの最短経路長 $dist[N]$ を求め、それを出力する以下のようなプログラムを提出すると、正解が得られます。

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

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

// グラフ
bool used[500009];
long long dist[500009]; // dist[i] は頂点 1 → 頂点 i の最短経路長
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
// ダイクストラ法
void dijkstra() {
    // 配列の初期化など
    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));

    // キューの更新
    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; // 頂点 0 の場合は例外
            if (dist[to] > cost) {
                dist[to] = cost;
                Q.push(make_pair(dist[to], to));
            }
        }
    }
}

int main() {
    // 入力・グラフの辺の追加
    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]));
    }

    // ダイクストラ法
    dijkstra();

    // 答えの出力
    if (dist[N] == (1LL << 60)) cout << "-1" << endl;
    else cout << dist[N] << endl;
    return 0;
}
```

※ Python などのソースコードは chap5-8.md をご覧ください。