いろはちゃんコンテスト解説

Day2-G:通学路

@Pro_ktmr

注意

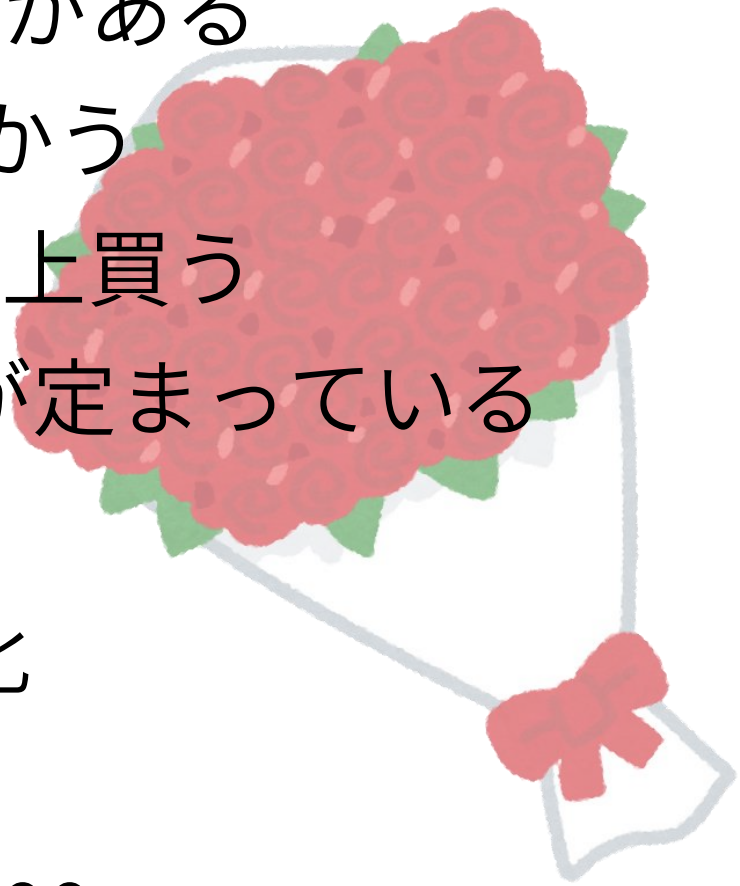
- 内容の正確性には十分注意していますが、間違った情報が含まれることもあります。何かお気づきのことがあればご連絡ください。
- このスライドのサンプルコードはC++で実装されています。

この問題の狙い

- ダイクストラ法が実装できるか
- 頂点を拡張することができるか　など

問題概要

- 駅と鉄道路線(グラフ)がある
- 頂点1から頂点Nへ向かう
- 道中で花を合計K本以上買う
- 路線、花にはコストが定まっている



- コストの合計を最小化

- N,K ≦ 1000　M ≦ 2000

考察

- キーワード「グラフ」「最小」といえばダイクストラ法
- しかし、ダイクストラ法では、何本花を買っているかの情報を持つことができない

考察

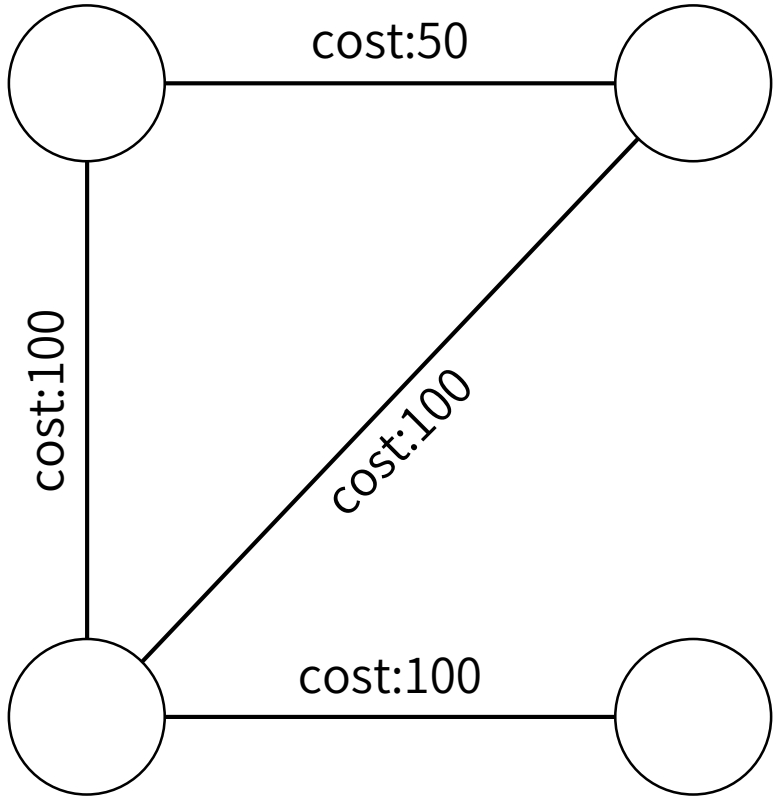
- キーワード「グラフ」「最小」といえばダイクストラ法
- しかし、ダイクストラ法では、何本花を買っているかの情報を持つことができない？

考察

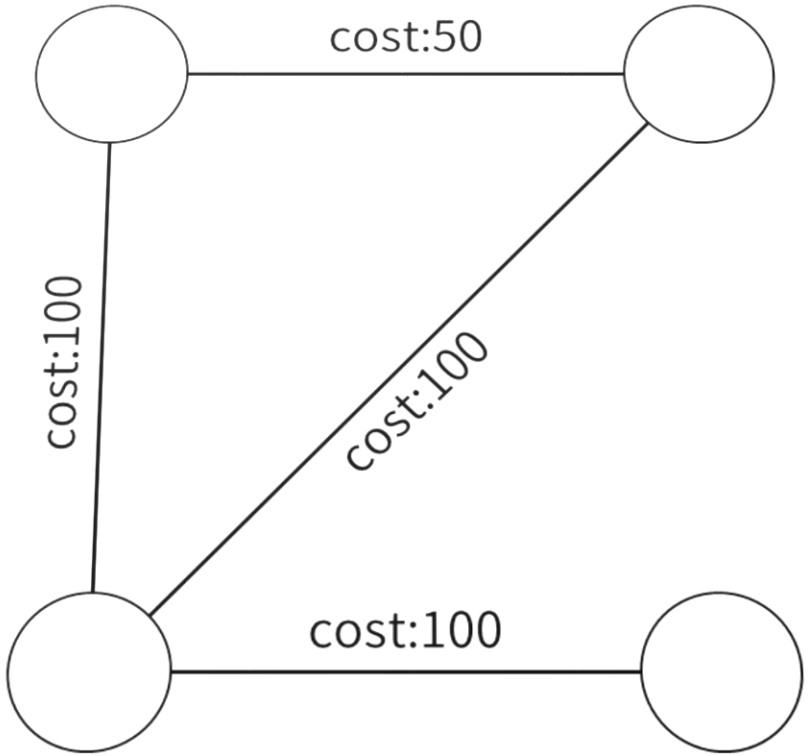
- キーワード「グラフ」「最小」といえばダイクストラ法
- しかし、ダイクストラ法では、何本花を買っているかの情報を持つことができる

- 頂点を増やしてしまえばよい

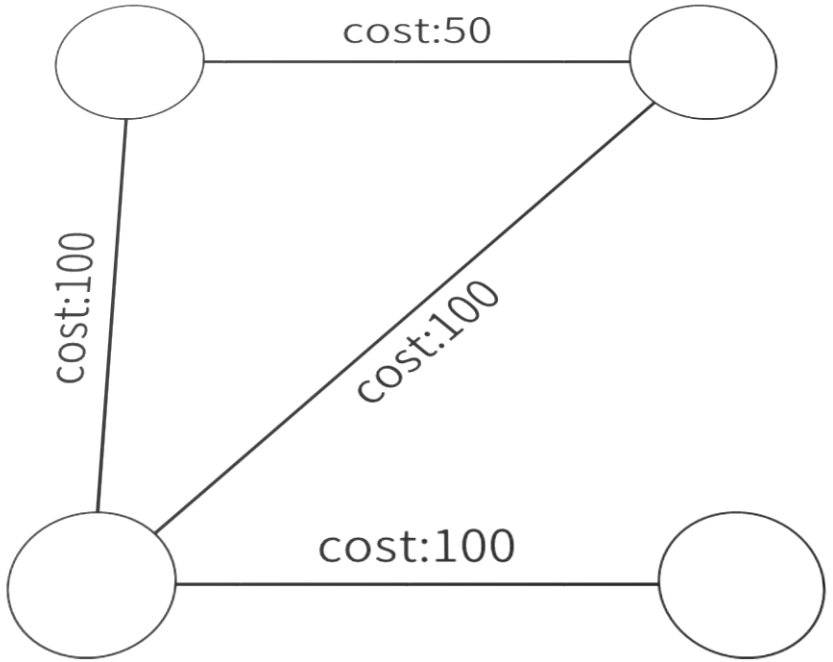
頂点の拡張：入出力例1



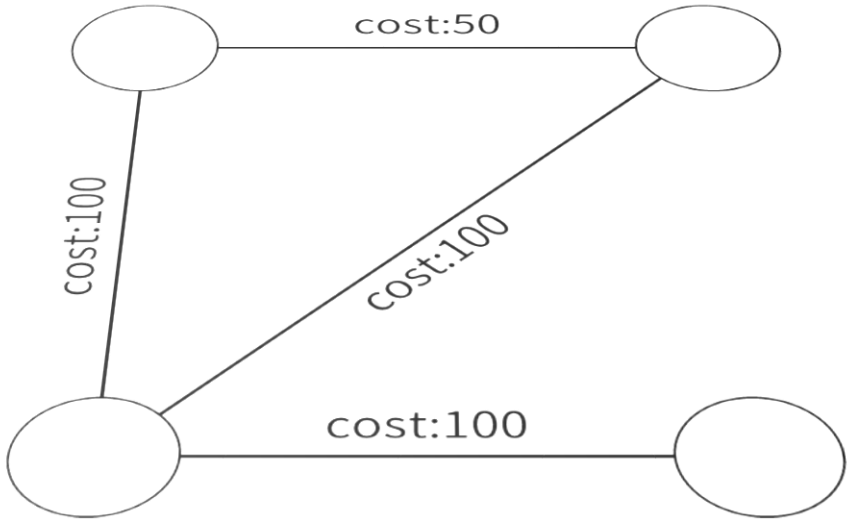
頂点の拡張：入出力例1



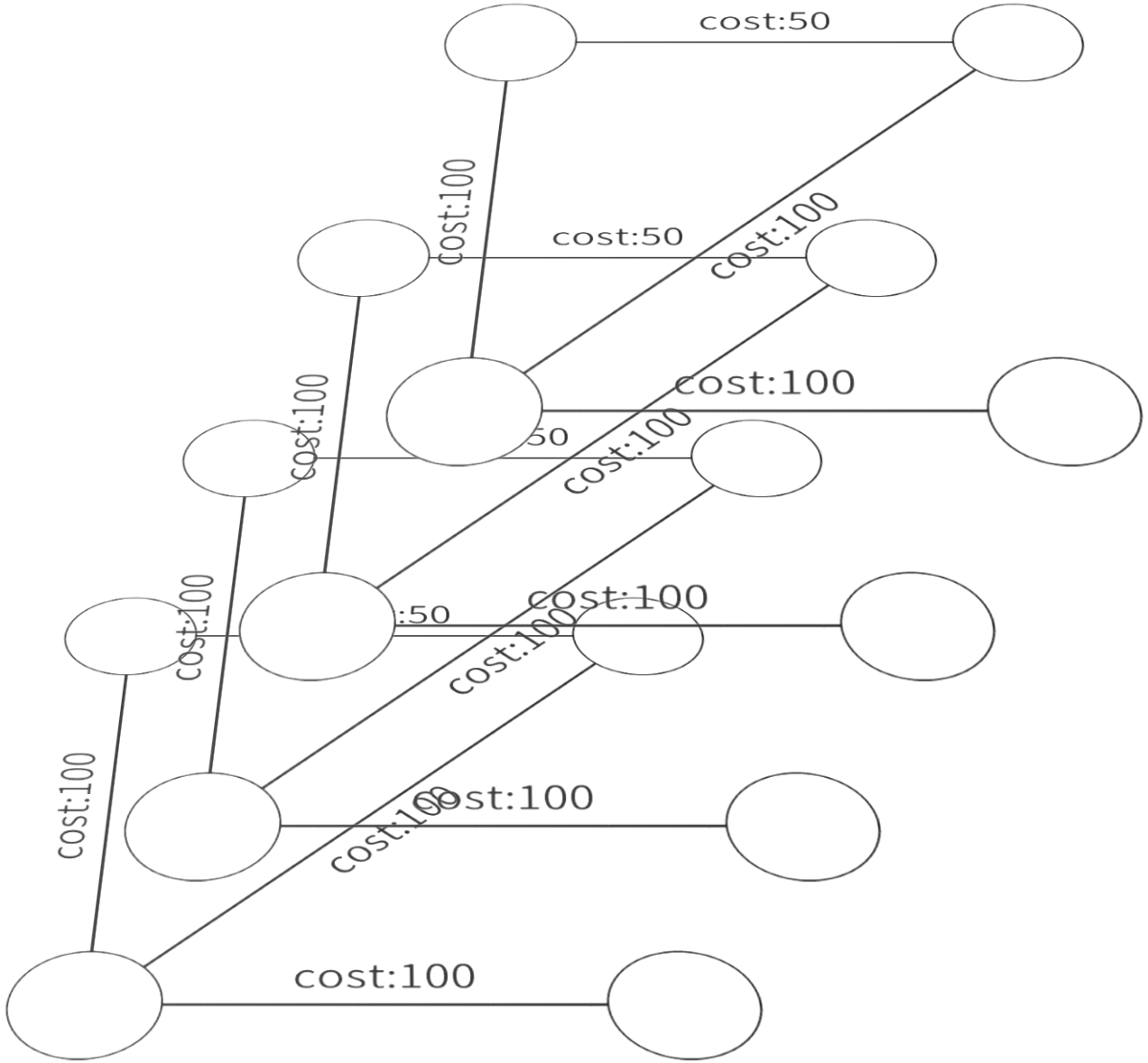
頂点の拡張：入出力例1



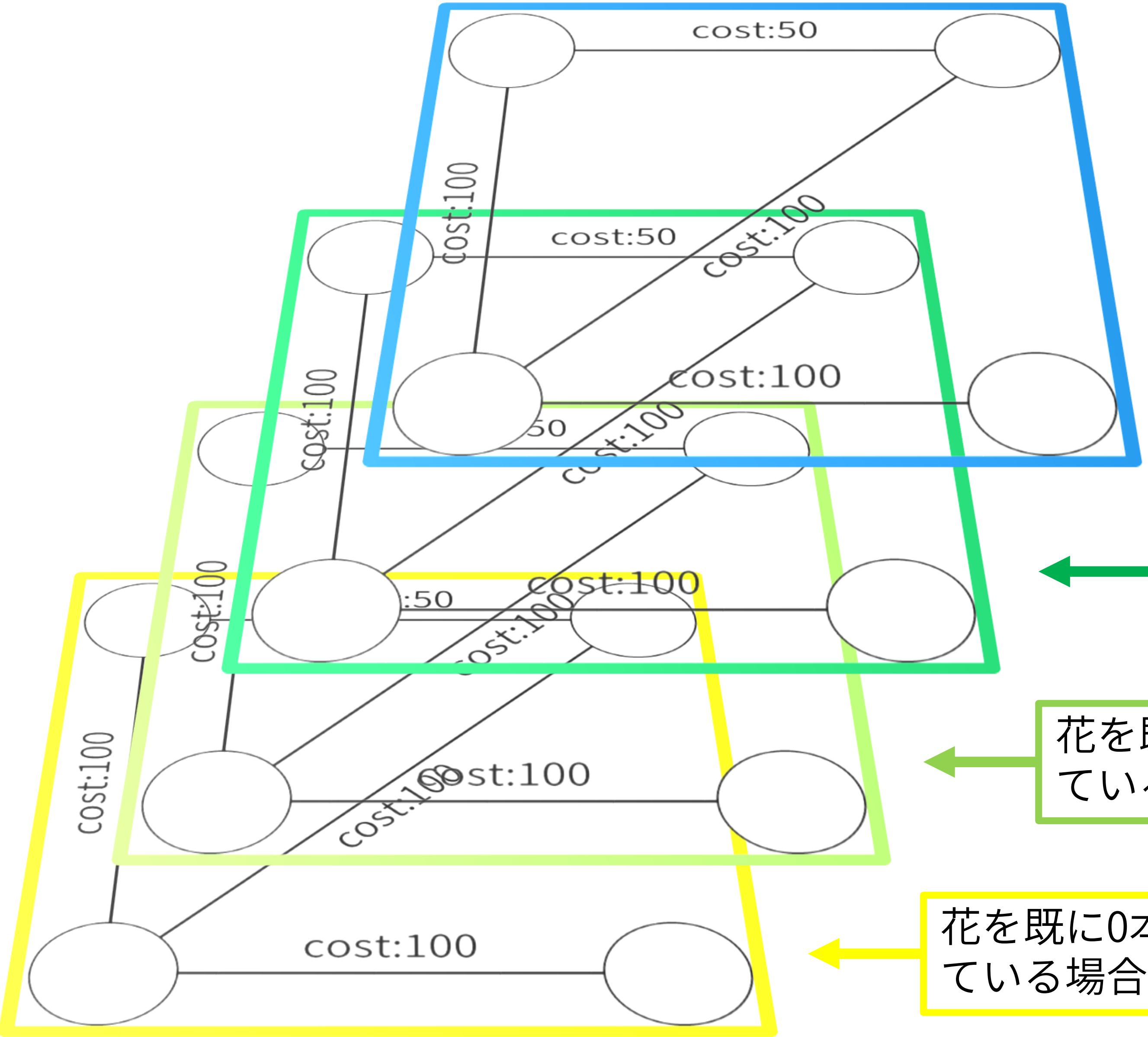
頂点の拡張：入出力例1



頂点の拡張：入出力例1


cost:50
cost:100
cost:50
cost:100
cost:100
cost:100
cost:100
cost:100
cost:100
cost:100
cost:100
cost:100
cost:100
cost:100

頂点の拡張：入出力例1



花を既に3本買っている場合の層

花を既に2本買っている場合の層

花を既に1本買っている場合の層

花を既に0本買っている場合の層

考察

- このように頂点を拡張すると
- 頂点数はK倍または2K倍になる
  →2NK個 $\leqq 2\times10^6$
- 辺数もK倍になり、上下を接続する辺が最大で2NK本増える
  →MK+2NK本 $\leqq 4\times10^6$

- ダイクストラ法の時間計算量はO(ElogV)だから、この拡張を行っても十分高速

余談

- 頂点を拡張してダイクストラ法を行うことを、拡張ダイクストラ法と呼ぶこともある
- ダイクストラ法を拡張しているような誤解を招くことから、この表現を嫌う人も多い
- 結局このテクを何と呼ぶかはTwitterでも未だに議論が盛んである

解法① 頂点を拡張したダイクストラ法

```
#include"bits/stdc++.h"
using namespace std;
const long long INF = 1LL<<62;
int N, M, K, X[2000];
long long Y[2000];
vector<pair<int,long long>> edge[2000];
long long cost[2000][4000];
int main(){
        cin >> N >> M >> K;
        for(int i=0; i<M; i++){
                int A, B;
                long long C;
                cin >> A >> B >> C;
                A--; B--;
                edge[A].push_back(make_pair(B, C));
                edge[B].push_back(make_pair(A, C));
        }
        for(int i=0; i<N; i++){
                cin >> X[i] >> Y[i];
                for(int j=0; j<2*K; j++) cost[i][j] = INF;
        }
        cout << dykstra() << endl;
}
```

```
long long dykstra(){
        priority_queue<pair<long long, pair<int, int>>> que;
        cost[0][0] = 0; que.push({ 0,{0,0} });
        while(!que.empty()){
                long long c = -que.top().first;
                long long sta = que.top().second.first;
                long long flo = que.top().second.second;
                que.pop();
                if(flo < K && cost[sta][flo+X[sta]] > c + Y[sta]){
                        cost[sta][flo+X[sta]] = c + Y[sta];
                        que.push({ -
cost[sta][flo+X[sta]] ,{sta,flo+X[sta]} });
                }
                for(int i=0; i<edge[sta].size(); i++){
                        if(cost[edge[sta][i].first][flo] > c +
edge[sta][i].second){
                                cost[edge[sta][i].first][flo] = c +
edge[sta][i].second;
                                que.push({ -
cost[edge[sta][i].first][flo] ,{edge[sta][i].first,flo} });
                        }
                }
        }
        long long answer = INF;
        for(int i=0; i<K; i++) answer = min(answer, cost[N-1][K+i]);
        if(answer == INF) answer = -1;
        return answer;
}
```

Thank you for reading!