My First Document

My Name

目录

1 Introduction

This is the introduction. ?!

2 Methods

2.1 Stage 1

The first part of the methods.

2.2 Stage 2

The second part of the methods.

3 Results

Here are my results. Referring to section 2.2 on page 2

中文测试

```cpp
#include <bits/stdc++.h>

template<typename T>
class ST {
private:
    std::vector<std::vector<T>> dp;
    T (*get) (T, T);
// [](int x, int y) { return std::gcd(x, y); }
// [](int x, int y) { return std::max(x, y); }
// [](int x, int y) { return std::min(x, y); }
public:
    ST(const std::vector<T>& inputs, auto getFunc) {
        get = getFunc;
        size_t len = inputs.size();
        int exp = log2(len);
        // dp[s][k] 代表从 s 出发走 2^k 步内的最值
        dp.resize(len, std::vector<T>(exp + 1, 0));
        for (size_t s = 0; s < len; s++) {
            dp[s][0] = inputs[s];
        }

        for (int k = 1; k <= exp; k++) {
            for (size_t s = 0; s + (1 << k) <= len; s++) {
                dp[s][k] = get(dp[s][k - 1], dp[s + (1 << (k - 1))][k - 1]);
            }
        }
    }

    T query(size_t start, size_t end) const {
```

```
        if (start > end) throw std::invalid_argument("start should be less than or
        ↪  equal to end");
        int exp = log2(end - start + 1);
        return get(dp[start][exp], dp[end - (1 << exp) + 1][exp]);
    }
};
```

```
#include <bits/stdc++.h>

struct node {
    std::vector<std::pair<int, int>> to;
    bool finished = false;
    int minDis = INT_MAX;
};

struct task {
    int id, dis;
    task(int id, int dis) : id(id), dis(dis) {}

    bool operator>(const task& another) const {
        return dis > another.dis;
    }
};

void solve()
{
    int N, M, S; // 点 边 出发点
    std::cin >> N >> M >> S;
    std::vector<node> graph(N + 1);
    for (int i = 0; i < M; i++) {
        int u, v, len;
        std::cin >> u >> v >> len;
        graph[u].to.emplace_back(v, len);
        //graph[v].to.emplace_back(u, len);
    }

    std::priority_queue<task, std::vector<task>, std::greater<task>> pending;
    graph[S].minDis = 0;
    pending.emplace(S, 0);
    while(!pending.empty()) {
        node& cur = graph[pending.top().id];
        pending.pop();
        if (cur.finished) continue;
        cur.finished = true;
        for (auto [id, cost] : cur.to) {
            if (graph[id].finished) continue;
            if (graph[id].minDis > cur.minDis + cost) {
```

```
                graph[id].minDis = cost + cur.minDis;
                pending.emplace(id, cost + cur.minDis);
            }
        }
    }

    for (int i = 1; i <= N; i++) {
        std::cout << graph[i].minDis << ' ';
    }
}

int main()
{
    std::cin.tie(nullptr)->sync_with_stdio(false);
    solve();
    return 0;
}
```