# 暨南大学本科实验报告专用纸(附页)

## 图上 bfs（最短路）

课程名称 数据结构 成绩评定
实验项目名称 图上 bfs（最短路） 指导老师 干晓聪
实验项目编号 07 实验项目类型 设计性 实验地点 数学系机房
学生姓名 郭彦培 学号 2022101149
学院 信息科学技术学院 系 数学系 专业 信息管理与信息系统
实验时间 2024 年 6 月 13 日上午 ~ 2024 年 7 月 13 日中午

## 1. 实验目的

利用优先队列优化的 bfs 实现 Dijstra 算法求最短路

## 2. 实验环境

计算机：PC X64

操作系统：Windows + Ubuntu20.0LTS

编程语言：C++：GCC std20

IDE：Visual Studio Code

## 3. 程序原理

已确定最短路的节点集合$\mathbb{S}$，未确定的节点集合$\mathbb{T}$

对$\mathbb{T}$中最小的节点$T$进行 BFS，松弛其所有子节点后，将$T$加入$\mathbb{S}$中，直到算法收敛。

对于松弛操作$S \to u \to v$ 有 $\mathrm{dis}(v) = \min(\mathrm{dis}(v), \mathrm{dis}(u) + w(u, v))$

# 4. 程序代码

## 4.1. bfs.cpp

```cpp
#include <iostream>
#include <vector>
#include <stdlib.h>
#include <map>
#include <set>
#include <algorithm>
#include <queue>

using namespace std;
using pii = pair<int, int>;

#define int long long
#define pb push_back
#define F first
#define S second
#define all(x) x.begin(), x.end()
#define loop(i, n) for (int i = 0; i < n; i++)

const int mod = 1e9 + 7;
const int INF = 1e18;

// 优先队列队列 BFS 求最短路
//
void solve()
{
    int n, m;
    cin >> n >> m;
    vector<vector<pii>> cnj(n + 1);
    vector<int> rcnj(n+1);
    loop(i, m)
    {
        int u, v, w;
        cin >> u >> v >> w;
        cnj[u].pb({v, w});
    }
     priority_queue<pii,vector<pii>,greater<pii>> dfsOrder;
    set<int> unReached;
    loop(i, n) unReached.insert(i + 1);
    vector<int> dis(n+1,INF);
    vector<int> locked(n+1,0);
    int ori, tar;
    cin >> ori >> tar;
    dis[ori] = 0;
    dfsOrder.push({0, ori});

    auto release = [&](int _n) -> void
    {
```

```
        for (auto x : cnj[_n])
        {
            // dis[x.first] = min(dis[x.first], dis[_n] + x.second);
            // dfsOrder.push({dis[x.first], x.first});
            if(locked[x.first]) continue;
            if(dis[x.first] > dis[_n] + x.second) {
                dis[x.first] = dis[_n] + x.second;
                rcnj[x.first] = _n;
            } dfsOrder.push({dis[x.first], x.first});
        }
    };

    while (unReached.size())
    {
        auto u = dfsOrder.top();
        dfsOrder.pop();
        release(u.second);
        unReached.erase(u.second);
        locked[u.second] = 1;
    }

    vector<int> path;

    int ptt = tar;
    while(ptt != ori) {
        path.pb(ptt);
        ptt = rcnj[ptt];
    } path.pb(ori);

    loop(i,path.size()) cout << path[path.size()-1-i] <<( i != path.size()-1?" -> ":"\n");

}

signed main()
{
    // std::ios::sync_with_stdio(false);
    // std::cin.tie(nullptr);
    // std::cout.tie(nullptr);

    int T = 1;
    cin >> T;
    while (T--)
        solve();
    system("pause");
    return 0;
}
```

## 5. 测试数据与运行结果

运行上述`_PRIV_TEST.cpp`测试代码中的正确性测试模块，得到以下内容：

```
1
8 13
1 2 2
1 3 14
1 4 15
2 3 2
3 4 3
2 5 15
3 5 15
4 6 1
6 5 3
5 6 3
5 7 2
5 8 14
8 7 3
1 7
1 -> 2 -> 3 -> 4 -> 6 -> 5 -> 7
```

可以看出，代码运行结果与预期相符，可以认为代码正确性无误。