问题求解（三）代码模板

（以下按照所学顺序排序）

- 单源最短路径dijkstra
- 多源最短路径floyd
- 二分图匹配Hungarian
- 最大流dinic

单源最短路径

题目描述

给定一个 $n+1$ 个点（编号从0到n），m 条有向边的带非负权图，请你计算从 s 出发，到每个点的距离。

若不能达到，输出-1。

输入格式

第一行为三个正整数 n, m, s。
第二行起 m 行，每行三个非负整数 u_i, v_i, w_i，表示从 u_i 到 v_i 有一条权值为 w_i 的有向边。

输出格式

输出一行 $n+1$ 个空格分隔的非负整数，表示 s 到每个点的距离。

样例 #1

样例输入 #1

```
4 6 1
1 2 2
2 3 2
2 4 1
1 3 5
3 4 3
1 4 4
```

样例输出 #1

```
-1 0 2 4 3
```

数据范围

$1 \le n \le 10^5$; $1 \le m \le 2 \times 10^5$; $0 \le u_i, v_i \le n$; $0 \le w_i \le 2 * 10^9$。

代码

```
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <iostream>
#include <queue>
```

```cpp
using namespace std;
typedef long long ll;
const int N = 1000000 + 5;
const int M = 1000000 + 5;
const ll inf = 0x3f3f3f3f3f3f3f3fll;
int h[N], cnt;
struct node{
    int next;
    int to;
    ll val;
}e[M];
void addedge(int x, int y, ll z) {
    cnt++; e[cnt].next = h[x]; e[cnt].to = y; e[cnt].val = z; h[x] = cnt;
}
ll dis[N];
bool vis[N];
priority_queue<pair<ll, int> >pq;
int n, m, s;
void dij() {
    memset(dis, 0x3f, sizeof(dis));
    dis[s] = 0;
    for (int i = 0; i <= n; i++) vis[i] = false;
    pq.push(make_pair(0, s));
    while(!pq.empty()) {
        pair<ll, int> now = pq.top(); pq.pop();
        int x = now.second;
        if (vis[x]) continue;
        vis[x] = true;
        for (int i = h[x]; i; i = e[i].next) {
            int y = e[i].to;
            if (dis[y] > dis[x] + e[i].val) {
                dis[y] = dis[x] + e[i].val;
                pq.push(make_pair(dis[y] * -1, y));
            }
        }
    }

}
int main() {
    scanf("%d%d%d", &n, &m, &s);
    for (int i = 1; i <= m; i++) {
        int x, y;
        ll z;
        scanf("%d%d%lld", &x, &y, &z);
        addedge(x, y, z);
    }
    dij();
    for (int i = 0; i <= n; i++) printf("%lld ", dis[i] >= inf ? -1 : dis[i]);
    return 0;
```

```
}
```

多源最短路径

题目描述

给定一个 $n+1$ 个点（编号从0到n），m 条有向边的带非负权图，请你计算任意两点的距离。

输入格式

第一行为三个正整数 n, m。
第二行起 m 行，每行三个非负整数 u_i, v_i, w_i，表示从 u_i 到 v_i 有一条权值为 w_i 的有向边。

输出格式

输出$(n+1) \times (n+1)$的矩阵，其中第i行第j列的数字表示$i-1$到$j-1$的最短距离。若其不存在，则为-1。

样例 #1

样例输入 #1

```
4 6
1 2 2
2 3 2
2 4 1
1 3 5
3 4 3
1 4 4
```

样例输出 #1

```
0 -1 -1 -1 -1
-1 0 2 4 3
-1 -1 0 2 1
-1 -1 -1 0 3
-1 -1 -1 -1 0
```

数据范围

$1 \le n \le 500$；$1 \le m \le 2 \times 10^5$；$0 \le u_i, v_i \le n$；$0 \le w_i \le 2 * 10^9$。

代码

```
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <iostream>
#include <queue>
using namespace std;
const int N = 500 + 5;
typedef long long ll;
const ll inf = 0x3f3f3f3f3f3f3f3fll;
ll g[N][N];
int n, m;
int main()
{
    scanf("%d%d", &n, &m);
    memset(g, 0x3f, sizeof(g));
    for (int i = 0; i <= n; i++)
        g[i][i] = 0;
    for (int i = 1; i <= m; i++) {
        int x, y;
        ll z;
        scanf("%d%d%lld", &x, &y, &z);
        g[x][y] = min(g[x][y], z);
    }
    for (int k = 0; k <= n; k++)
        for (int i = 0; i <= n; i++)
            for (int j = 0; j <= n; j++)
                g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= n; j++)
            printf("%lld ", g[i][j] >= inf ? -1 : g[i][j]);
        printf("\n");
    }
    return 0;
}
```

二分图最大匹配

题目描述

给定一个二分图，其左部点的个数为 n，右部点的个数为 m，边数为 e，求其最大匹配的边数。

左部点从 1 至 n 编号，右部点从 1 至 m 编号。

输入格式

输入的第一行是三个整数，分别代表 n，m 和 e。

接下来 e 行，每行两个整数 u, v，表示存在一条连接左部点 u 和右部点 v 的边。

输出格式

输出一行一个整数，代表二分图最大匹配的边数。

样例 #1

样例输入 #1

```
1 1 1
1 1
```

样例输出 #1

```
1
```

数据范围

$1 \leq n, m \leq 500$，$1 \leq e \leq 5 \times 10^4$，$1 \leq u \leq n$，$1 \leq v \leq m$。

代码

```
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <iostream>
using namespace std;
const int N = 1000 + 5;
const int M = 1000000 + 5;
int h[N], cnt;
struct node{
    int next;
    int to;
```

```
}e[M];
void addedge(int x, int y) {
    cnt++; e[cnt].next = h[x]; e[cnt].to = y; h[x] = cnt;
}
int n, m, ee;
int w[N] ,vis[N];
bool hunger(int x) {
    for (int i = h[x]; i; i = e[i].next) {
        int y = e[i].to;
        if (vis[y]) continue;
        vis[y] = 1;
        if (w[y] == -1 || hunger(w[y])) {
            w[y] = x;
            return true;
        }
    }
    return false;
}
int main() {
    scanf("%d%d%d" ,&n, &m, &ee);
    for (int i = 1; i <= ee; i++) {
        int x, y;
        scanf("%d%d", &x, &y);
        addedge(x, y + n);
    }
    memset(w, -1, sizeof(w));
    int ans = 0;
    for (int i = 1; i <= n; i++) {
        memset(vis, 0,sizeof(vis));
        ans += hunger(i);
    }
    printf("%d\n", ans);
    return 0;
}
```

网络流

题目描述

给出一个网络图，以及其源点和汇点，求出其网络最大流。

输入格式

第一行包含四个正整数 n, m, s, t，分别表示点的个数、有向边的个数、源点序号、汇点序号。

接下来 m 行每行包含三个正整数 u_i, v_i, w_i，表示第 i 条有向边从 u_i 出发，到达 v_i，边权为 w_i（即该边最大流量为 w_i）。

输出格式

一行，包含一个正整数，即为该网络的最大流。

样例 #1

样例输入 #1

```
4 5 4 3
4 2 30
4 3 20
2 3 20
2 1 30
1 3 40
```

样例输出 #1

```
50
```

数据范围

$1 \leq n \leq 200$，$1 \leq m \leq 5000$，$0 \leq w < 2^{31}$。

代码

```
#include <cstring>
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <queue>
using namespace std;
typedef long long ll;
const int N = 1000000 + 5;
```

```
const ll inf = 100000000000000ll;
int h[N], cnt;
struct node{
    int next;
    int to;
    int val;
}e[N];
void addedge(int x, int y, int z) {
    cnt++; e[cnt].next = h[x]; e[cnt].to = y; e[cnt].val = z; h[x] = cnt;
}
int n, m, s, t;
ll ans;
ll dep[N];
bool bfs() {
    queue<int >q;
    for (int i = 1; i <= n; i++) dep[i] = inf;
    dep[s] = 0;
    q.push(s);
    while(!q.empty()) {
        int x = q.front(); q.pop();
        for (int i = h[x]; i; i = e[i].next) {
            int y = e[i].to;
            if (dep[y] != inf || e[i].val == 0) continue;
            dep[y] = dep[x] + 1;
            q.push(y);
        }
    }
    return dep[t] != inf;
}
int cur[N];
ll dfs(int x, ll minn) {
    if (x == t) return minn;
    ll tmp, now = 0;
    for (int i = cur[x]; i; i = e[i].next) {
        int y = e[i].to; cur[x] = i;
        if (e[i].val == 0) continue;
        if (dep[y] == dep[x] + 1 && (tmp = dfs(y, min(minn, (ll)e[i].val)))) {
            e[i].val -=tmp; e[i ^ 1].val += tmp;
            minn -= tmp; now += tmp;
            if (minn == 0) break;
        }
    }
    return now;
}
void dinic() {
    while(bfs()) {
        for (int i = 1; i <= n; i++) cur[i] = h[i];
        ans += dfs(s, inf);
    }
```

```
    printf("%lld\n", ans);
}

int main() {
    cnt = 1;
    scanf("%d%d%d%d", &n, &m, &s, &t);
    for (int i = 1; i <= m; i++) {
        int x, y, z;
        scanf("%d%d%d" ,&x, &y, &z);
        addedge(x, y, z);
        addedge(y, x, 0);
    }
    dinic();
    return 0;
}
```