《第四次上机实验》解题报告

1.图的深度优先搜索 I

1.1 题目描述

分数　100
作者　谷方明
单位　吉林大学

无向图 G 有 n 个顶点和 m 条边。求图 G 的深度优先搜索树(森林)以及每个顶点的发现时间和完成时间。每个连通分量从编号最小的结点开始搜索，邻接顶点选择顺序遵循边的输入顺序。
在搜索过程中，第一次遇到一个结点，称该结点被发现；一个结点的所有邻接结点都搜索完，该结点的搜索被完成。深度优先搜索维护一个时钟，时钟从 0 开始计数，结点被搜索发现或完成时，时钟计数增 1，然后为当前结点盖上时间戳。一个结点被搜索发现和完成的时间戳分别称为该结点的发现时间和完成时间

输入格式:
第 1 行，2 个整数 n 和 m，用空格分隔，分别表示顶点数和边数， 1≤n≤50000，1≤m≤100000.
第 2 到 m+1 行，每行两个整数 u 和 v，用空格分隔，表示顶点 u 到顶点 v 有一条边，u 和 v 是顶点编号，1≤u,v≤n.

输出格式:
第 1 到 n 行，每行两个整数 di 和 fi，用空格分隔，表示第 i 个顶点的发现时间和完成时间 1≤i≤n 。
第 n+1 行，1 个整数 k ，表示图的深度优先搜索树(森林)的边数。
第 n+2 到 n+k+1 行，每行两个整数 u 和 v,表示深度优先搜索树(森林)的一条边<u,v>，边的输出顺序按 v 结点编号从小到大。

输入样例:
在这里给出一组输入。例如：
6 5
1 3
1 2
2 3
4 5
5 6
输出样例:
在这里给出相应的输出。例如：

1 6
3 4
2 5
7 12
8 11
9 10
4
3 2
1 3
4 5
5 6

1.2 思路

时间戳：对图进行深度优先搜索，在对每个结点 dfs 开始和结束时打上时间戳，分别使用两个数组保存发现时间、完成时间。

边的输出顺序：搜索完成后使用 sort 对 v 进行排序；

所遇问题及收获：
使用 i <= path.size()-1 作为循环判断条件时在第一个点会发生段错误;
原因为:vector 的 size()返回类型为 unsigned int ,因此当 vector 为空时-1，会转为一个很大的正数，循环不能按预期终止。

1.3 代码：

```
#include <iostream>
#include <vector>
#include <algorithm>
#define cit(x,y,z)    for(int (x)=(y);(x)<=(z);(x)++)

inline int read() {
    int x = 0; int w = 1; register char c = getchar();
    for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
    if (c == '-') w = -1, c = getchar();
    for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1) + c - '0';
    return x * w;
}
using namespace std;
const int maxn = 50010;
```

```
vector<int> G[maxn];
vector<pair<int, int>> path;
int n, e;
int st[maxn]={0}, en[maxn]={0};
int tim = 0;
void dfs(int u) {
	st[u] = ++tim;
	for (int i = 0; i < G[u].size(); i++) {
		int v = G[u][i];
		if (!st[v])path.push_back(make_pair(v, u)),dfs(v);
	}
	en[u] = ++tim;
}

int main() {
	n = read(); e = read();
	int	u = 0, v = 0;
	cit(i,1,e) {
		u = read(); v = read();
		G[u].push_back(v);
		G[v].push_back(u);
	}
	cit(i,1,n)if(!st[i])dfs(i);
	cit(i,1,n)printf("%d %d\n", st[i], en[i]);
	printf("%d\n", path.size());
	sort(path.begin(), path.end(), less<pair<int, int>>());

	for (int i = 0; i < path.size();i++)printf("%d %d\n", path[i].second, path[i].first);
	return 0;
}
```

2. 数字变换

2.1 题目描述

分数　100
作者　谷方明
单位　吉林大学

利用变换规则，一个数可以变换成另一个数。变换规则如下：（1）x 变为 x+1；（2）x 变为 2x；（3）x 变为 x-1。给定两个数 x 和 y，至少经过几步变换能

让 x 变换成 y.

输入格式:
1 行，2 个整数 x 和 y，用空格分隔， 1≤x,y≤100000.

输出格式:
第 1 行，1 个整数 s，表示变换的最小步数。
第 2 行，s 个数，用空格分隔，表示最少变换时每步变换的结果。规则使用优先级顺序: （1），（2），（3）。

输入样例:
在这里给出一组输入。例如：
2 14

输出样例:
在这里给出相应的输出。例如：
4
3 6 7 14

2.2 思路

通过 bfs 找到 x,y 之间的最短路。

通过找到合理约束条件减少不必要的扩展：
1. 对 x+1,当 x>y 时即可停止对此状态的扩展；
2. 对 x*2,若 x*2>y+1,考虑到路径最短，则在此之前应先进行 x-1 操作。
3. 对 x-1，因有 1≤x,y,所以当 x-1<0 时，可停止对此状态的扩展；

路径存储：使用 path 指针指向前一个结点

2.3 代码

```
#include<iostream>
#include<queue>

using namespace std;
const int maxn = 1e5 + 5;

typedef struct node {
    int data;
    struct node* path;
    node(int d, struct node* p):data(d),path(p){}
```

```
}Node;
int x, y;
bool vis[maxn] = { 0 };
queue<Node*> Q;

void bfs(int s, int end) {
	Node* f = new Node{s,NULL}, * tmp;
	Q.push(f);
	while (!Q.empty()) {
		tmp = Q.front();
		if (tmp->data == y)return;
		Q.pop();
		int dat = tmp->data;
		if (dat + 1 <= y && !vis[dat + 1]) {
			Node* p = new Node{dat+1,tmp};
			Q.push(p); vis[dat + 1] = 1;
		}
		if (dat << 1 <= y + 1 && !vis[dat << 1]) {
			Node* p = new Node{dat<<1,tmp};
			  Q.push(p); vis[dat << 1] = 1;
		}
		if (dat - 1 >= 0 && !vis[dat - 1]) {
			Node* p = new Node{dat-1,tmp};
			Q.push(p); vis[dat - 1] = 1;
		}
	}
}

int main() {
	scanf("%d%d", &x, &y);
	bfs(x, y);
	int road[maxn];
	int rp = 0;
	for (Node* t = Q.front(); t; t = t->path)
		road[rp++] = t->data;
	printf("%d\n", rp - 1);
	for (rp -= 2; rp >= 0; rp--)
		if (rp != 0)printf("%d ", road[rp]);
		else printf("%d\n", road[rp]);

	return 0;
}
```

3.修轻轨

3.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

长春市有 n 个交通枢纽，计划在 1 号枢纽到 n 号枢纽之间修建一条轻轨。轻轨由多段隧道组成，候选隧道有 m 段。每段候选隧道只能由一个公司施工，施工天数对各家公司一致。有 n 家施工公司，每家公司同时最多只能修建一条候选隧道。所有公司可以同时开始施工。请评估：修建这条轻轨最少要多少天。

输入格式:
第 1 行，两个整数 n 和 m，用空格分隔，分别表示交通枢纽的数量和候选隧道的数量，$1 \le n \le 100000$，$1 \le m \le 200000$。
第 2 行到第 m+1 行，每行三个整数 a、b、c，用空格分隔，表示枢纽 a 和枢纽 b 之间可以修建一条双向隧道，施工时间为 c 天，$1 \le a, b \le n$，$1 \le c \le 1000000$。

输出格式:
输出一行，包含一个整数，表示最少施工天数。

输入样例:
在这里给出一组输入。例如：
6 6
1 2 4
2 3 4
3 6 7
1 4 2
4 5 5
5 6 6

输出样例:
在这里给出相应的输出。例如：
6

3.2 思路

修建一条轻轨的最少天数取决于所选择的路径中施工时间最长的隧道，即找到两点之间最短路径上权值最大的边；

通过并查集模拟，每次从未选择的边中取权值最小的边合并，直到 1 和 n 连通；而因为边是从小到大选取，故最后并入的边即为 1 到 n 的路径中权值最大的边。

3.3 代码

```
#include<iostream>
#include<vector>
#include<algorithm>

inline int read() {
	int x = 0; int w = 1; register char c = getchar();
	for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
	if (c == '-') w = -1, c = getchar();
	for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1) + c - '0';
	return x * w;
}
using namespace std;
const int maxn = 1e5 + 5;
const int maxe = 2e5 + 5;

int n, e;
int fa[maxn];
typedef struct node {
	int u, w, v;
	node(int t1, int t2, int t3) :u(t1), w(t2), v(t3) {}
}edge;
vector<edge>Edges;

inline int Find(int x) {
	if (fa[x] <= 0)return x;
	return fa[x] = Find(fa[x]);
}
inline void Union(int x, int y) {
	int xf = Find(x), yf = Find(y);
	if (xf == yf)return;
	if (fa[xf] < fa[yf])fa[yf] = xf;
	else {
		if (fa[xf] == fa[yf])fa[yf]--;
		fa[xf] = yf;
	}
}

bool cmp(const edge& x, const edge& y) {
```

```
	return x.w < y.w;
}
void Kruskal() {
	sort(Edges.begin(), Edges.end(), cmp);
	for (int i = 0; i < e; i++) {
		Union(Edges[i].u, Edges[i].v);
		if (Find(1) == Find(n)) {
			printf("%d", Edges[i].w); break;
		}
	}
}

int main() {
	n = read(); e = read();

	for (int i = 1; i <= e; i++) {
		int x = read(), y = read(), w = read();
		Edges.push_back({ x,w,y });
	}
	if (n == 1) putchar('0');
	else	Kruskal();
	return 0;
}
```

4. 发红包

4.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

新年到了，公司要给员工发红包。员工们会比较获得的红包，有些员工会有钱数的要求，例如，c1 的红包钱数要比 c2 的多。每个员工的红包钱数至少要发 888 元，这是一个幸运数字。
公司想满足所有员工的要求，同时也要花钱最少，请你帮助计算。

输入格式:
第 1 行，两个整数 n 和 m(n<=10000,m<=20000)，用空格分隔，分别代表员工数和要求数。
接下来 m 行，每行两个整数 c1 和 c2，用空格分隔，表示员工 c1 的红包钱数要比 c2 多，员工的编号 1~n 。

输出格式:
一个整数，表示公司发的最少钱数。如果公司不能满足所有员工的需求，输出-1.

输入样例:
在这里给出一组输入。例如：
2 1
1 2

输出样例:
在这里给出相应的输出。例如：
1777

4.2 思路

通过拓扑排序分层，找到每个员工最小的红包金额；
建图：建立有向图，若要求 c1 比 c2 红包多,则 c2 为 c1 的一个约束，增加一条 c2 到 c1 的边；

使用 val 数组保存每个员工的最小红包金额，将没有要求的设为最小（888），每层之间相差 1；
使用队列存储每个入度为 0 的点，若入队数少于点数，则说明出现环，条件不能满足；

4.3 代码

```
#include<iostream>
#include<queue>
#define cit(x,y,z)    for(int x=y;x<=z;x++)

inline int read() {
	int x = 0; int w = 1; register char c = getchar();
	for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
	if (c == '-') w = -1, c = getchar();
	for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1) + c - '0';
	return x * w;
}
using namespace std;
const int maxn = 1e4 + 5;
const int maxe = 2e4 + 5;

int n, e, ecnt = 0;
int head[maxn], in[maxn];
int val[maxn];
```

```
long long allval = 0;
struct node {
	int to, next;
}edges[maxe];

inline void addedge(int x, int y) {
	edges[ecnt].to = y;
	edges[ecnt].next = head[x];
	head[x] = ecnt++;
}

void topo() {
	queue<int>Q;
	cit(i, 1, n) {
		if (in[i] == 0) { Q.push(i); val[i] = 888; }
	}
	cit(j, 1, n) {
		if (Q.empty()) { cout << -1; return; }
		int tmp = Q.front(); Q.pop();
		for (int i = head[tmp]; ~i; i = edges[i].next) {
			int near = edges[i].to; in[near]--;
			if (in[near] == 0) {
				val[near] = val[tmp] + 1; Q.push(near);
			}
		}
	}

	cit(i, 1, n)allval += (long long)val[i];
	printf("%lld\n", allval);
}

int main() {
	n = read(); e = read();
	cit(i, 0, n)head[i] = -1;
	cit(i, 1, e) {
		int x = read(), y = read();
		addedge(y, x); in[x]++;
	}
	topo();
}
```