1 Prim SMST 判断树唯一 POJ 1679 The Unique MST

1. **const** **int** inf = 0x3f3f3f3f;
2. **const** **int** maxn = 110;
4. **int** n, m, G[maxn][maxn];
5. **int** d[maxn];
6. **bool** vst[maxn];
7. **int** maxD[maxn][maxn];   //MST中从i->j的最大权值
8. **int** pre[maxn];          //某一点父节点
9. **bool** mst[maxn][maxn];   //该点是否已经在MST中
10. **typedef** pair<**int**, **int**> P;
12. **int** Prim(**int** s) {
13. **for**(**int** i = 0; i <= n; ++i){
14. d[i] = inf;
15. vst[i] = 0;
16. pre[i] = s;
17. **for**(**int** j = 0; j <= n; ++j){
18. mst[i][j] = 0;
19. maxD[i][j] = 0;
20. }
21. }
22. priority\_queue<P, vector<P>, greater<P> > q;
23. q.push(P(d[s]=0, s));
24. **int** sum = 0;
25. **while**(!q.empty()) {
26. P cur = q.top();
27. q.pop();
28. **int** k = cur.second;
29. **if**(vst[k]) **continue**;
30. vst[k] = 1;
31. sum += d[k];
32. mst[k][pre[k]] = mst[pre[k]][k] = **true**; //加入到MST中
33. **for**(**int** j = 1; j <= n; ++j) {
34. **if**(vst[k] && j != k)
35. maxD[k][j] = maxD[j][k] = max(maxD[pre[k]][k], d[k]);
36. **if**(G[k][j] < d[j]) {
37. d[j] = G[k][j];
38. pre[j] = k;                     //更新父节点
39. q.push(P(d[j], j));
40. }
41. }
42. }
43. **return** sum;
44. }
46. **int** main() {
47. **int** T, x, y, z;
48. scanf("%d", &T);
49. **while**(T--) {
50. memset(G, inf, **sizeof**(G));
51. scanf("%d%d", &n, &m);
52. **while**(m--) {
53. scanf("%d%d%d", &x, &y, &z);
54. G[x][y] = G[y][x] = z;
55. }
56. **int** ans = Prim(1);
57. **bool** flag = **false**;  // 次小生成树是否等于最小生成树
58. **for**(**int** u = 1; u <= n && !flag; ++u) {
59. **for**(**int** v = 1; v <= n; ++v) {
60. **if**(mst[u][v] || G[u][v]==inf)// 枚举所有未使用的边, 判断该边<i, j> 权值是否等于maxd[i][j]
61. **continue**;
62. **if**(G[u][v] == maxD[u][v]) {// 边<i, j> 权值等于maxd[i][j]说明不唯一
63. flag = **true**;
64. **break**;
65. }
66. }
67. }
68. **if**(flag)
69. printf("Not Unique!\n");
70. **else**
71. printf("%d\n", ans);
72. }
73. **return** 0;
74. }

2 Prim SMST 大秦 HDU 4081 Qin Shi Huang's National Road System

1. **typedef**  **long** **long** LL;
2. **const** **int** inf = 1 << 30;
3. **const** LL maxn = 1010;
5. **int** n;
6. **double** G[maxn][maxn];
7. **struct** node {
8. **int** x, y;
9. **int** p;
10. node(**int** xx, **int** yy, **int** pp) {x = xx, y = yy, p = pp;}
11. node() {}
12. } point[maxn];
14. **double** d[maxn];
15. **bool** vst[maxn];
16. **double** maxD[maxn][maxn];   //MST中从i->j的最大权值
17. **int** pre[maxn];          //某一点父节点
18. **bool** mst[maxn][maxn];   //该点是否已经在MST中
19. **typedef** pair<**int**, **int**> P;
21. **double** Prim(**int** s) {
22. **for**(**int** i = 0; i <= n; ++i){
23. d[i] = inf;
24. pre[i] = s;
25. vst[i] = 0;
26. **for**(**int** j = 0; j <= n; ++j){
27. maxD[i][j] = 0;
28. mst[i][j] = 0;
29. }
30. }
31. priority\_queue<P, vector<P>, greater<P> > q;
32. q.push(P(d[s] = 0, s));
33. **double** sum = 0;
34. **while**(!q.empty()) {
35. P cur = q.top();
36. q.pop();
37. **int** k = cur.second;
38. **if**(vst[k])
39. **continue**;
40. vst[k] = 1;
41. sum += d[k];
42. mst[k][pre[k]] = mst[pre[k]][k] = 1; //加入到MST中
43. **for**(**int** j = 1; j <= n; ++j) {
44. **if**(vst[j] && j != k)
45. maxD[k][j] = maxD[j][k] = max(maxD[pre[k]][j], d[k]);
46. **if**(G[k][j] < d[j]) {
47. d[j] = G[k][j];
48. pre[j] = k;                     //更新父节点
49. q.push(P(d[j], j));
50. }
51. }
52. }
53. **return** sum;
54. }
56. **double** getDis(**int** x1, **int** y1, **int** x2, **int** y2) {
57. **return** sqrt((**double**)(x1 - x2)\*(x1 - x2) + (y1 - y2)\*(y1 - y2));
58. }
60. **int** main() {
61. **int** T, x, y, z;
62. scanf("%d",&T);
63. **while**(T--) {
64. scanf("%d", &n);
65. **for**(**int** i = 0; i <= n; ++i){
66. point[i] = node(0, 0, 0);
67. **for**(**int** j = 0; j <= n; ++j)
68. G[i][j] = inf;
69. }
70. **for**(**int** i = 1; i <= n; ++i) {
71. scanf("%d%d%d", &x, &y, &z);
72. point[i] = node(x, y, z);
73. }
74. **for**(**int** i = 1; i < n; ++i)
75. **for**(**int** j = i+1; j <= n; ++j)
76. G[i][j] = G[j][i] = getDis(point[i].x, point[i].y, point[j].x, point[j].y);
78. //枚举删边, 找出最大值
79. **double** B = Prim(1), A, ans = -1;
80. **for**(**int** i = 1; i < n; ++i)
81. **for**(**int** j = i+1; j <= n; ++j){
82. A = point[i].p + point[j].p;
83. //这条边未在MST中使用, 尝试加边并删去生成环中的最长边, 已使用则直接变0
84. **if**(mst[i][j]){
85. ans = max(ans, A/(B - G[i][j]));
86. }**else**{
87. ans = max(ans, A/(B - maxD[i][j]));
88. }
89. }
90. printf("%.2f\n", ans);
91. }
93. **return** 0;
94. }

3 Prim SMST 裸题 UVA 10600

1. **typedef**  **long** **long** LL;
3. **const** **int** inf = 0x3f3f3f3f;
4. **const** LL maxn = 110;
6. **int** n, m;
7. **int** G[maxn][maxn];
9. **int** d[maxn];
10. **bool** vst[maxn];
11. **int** maxD[maxn][maxn];   //MST中从i->j的最大权值
12. **int** pre[maxn];          //某一点父节点
13. **bool** mst[maxn][maxn];   //该点是否已经在MST中
14. **typedef** pair<**int**, **int**> P;
16. **int** Prim(**int** s) {
17. **for**(**int** i = 0; i <= n; ++i){
18. d[i] = inf;
19. pre[i] = s;
20. vst[i] = 0;
21. **for**(**int** j = 0; j <= n; ++j){
22. maxD[i][j] = 0;
23. mst[i][j] = 0;
24. }
25. }
26. priority\_queue<P, vector<P>, greater<P> > q;
27. q.push(P(d[s] = 0, s));
28. **int** sum = 0;
29. **while**(!q.empty()) {
30. P cur = q.top();
31. q.pop();
32. **int** k = cur.second;
33. **if**(vst[k])
34. **continue**;
35. vst[k] = 1;
36. sum += d[k];
37. mst[k][pre[k]] = mst[pre[k]][k] = 1; //加入到MST中
38. **for**(**int** j = 1; j <= n; ++j) {
39. **if**(vst[j] && j != k)        // 更新全部结点的maxD
40. maxD[k][j] = maxD[j][k] = max(maxD[pre[k]][j], d[k]);
41. **if**(!vst[j] && G[k][j] < d[j]) {
42. d[j] = G[k][j];
43. pre[j] = k;                     // 更新父节点
44. q.push(P(d[j], j));
45. }
46. }
47. }
48. **return** sum;
49. }
51. **int** SMST(){
52. **int** res = inf;
53. **for**(**int** i = 1; i <= n; ++i){
54. **for**(**int** j = i+1; j <= n; ++j){
55. **if**(!mst[i][j] && G[i][j] != inf){
56. res = min(res, G[i][j] - maxD[i][j]);
57. }
58. }
59. }
60. **return** res;
61. }
63. **int** main() {
64. **int** T, x, y, z;
65. scanf("%d", &T);
66. **while**(T--) {
67. scanf("%d%d", &n, &m);
68. **for**(**int** i = 0; i <= n; ++i)
69. **for**(**int** j = 0; j <= n; ++j){
70. G[i][j] = inf;
71. }
73. **for**(**int** i = 1; i <= m; ++i) {
74. scanf("%d%d%d", &x, &y, &z);
75. G[x][y] = G[y][x] = z;
76. }
78. **int** ans = Prim(1);
79. **int** res = SMST();
81. printf("%d %d\n", ans, ans+res);
82. }
84. **return** 0;
85. }

4 Kruskal SMST 裸题 重边 UVA 10462 Is There A Second Way Left

1. #define ms(x, n) memset(x,n,sizeof(x));
2. **typedef**  **long** **long** LL;
3. **const** **int** inf = 1 << 30;
4. **const** LL maxn = 1100;
6. **int** n, m;
7. **struct** Edge {
8. **int** u, v, w;
9. Edge(**int** uu, **int** vv, **int** ww) {
10. u = uu, v = vv, w = ww;
11. }
12. Edge() {}
13. } es[maxn \* maxn];
15. **int** par[maxn], rak[maxn];
16. **void** init(**int** n) {
17. **for**(**int** i = 0; i <= n; i++)
18. par[i] = i, rak[i] = 0;
19. }
20. **int** Find(**int** x) {
21. **if**(x == par[x])
22. **return** x;
23. **else**
24. **return** par[x] = Find(par[x]);
25. }
27. **bool** cmp(**const** Edge &a, **const** Edge &b) {
28. **return** a.w < b.w;
29. }
31. **int** maxD[maxn][maxn];  //MST中从i->j的最大权值
32. **bool** vst[maxn \* maxn];
33. vector<**int**> mst[maxn]; //该点是否已经在MST中
34. **int** sum, csum;
36. **int** Kruskal() {
37. sort(es + 1, es + 1 + m, cmp);
38. init(n);
39. ms(maxD, 0);
40. ms(mst, 0);
41. ms(vst, 0);
42. **for**(**int** i = 0; i <= n; ++i)
43. mst[i].push\_back(i);
45. sum = 0;
46. **int** k = 0;
47. **for**(**int** i = 1; i <= m; ++i) {
48. **int** u = Find(es[i].u);
49. **int** v = Find(es[i].v);
50. **int** w = es[i].w;
51. **if**(Find(u) != Find(v)) {
52. sum += w;
53. vst[i] = **true**;
54. **for**(**int** j = 0; j < mst[u].size(); ++j)
55. **for**(**int** k = 0; k < mst[v].size(); ++k)
56. maxD[ mst[u][j] ][ mst[v][k] ] = maxD[ mst[v][k] ][ mst[u][j] ] = w;
57. par[u] = v;
58. **for**(**int** j = 0; j < mst[u].size(); ++j)
59. mst[v].push\_back(mst[u][j]);    //单向存储即可
60. **if**(++k == n-1) **break**;
61. }
63. }
64. //    cout << sum << endl;
65. **if**(k != n-1) **return** -1;
66. csum = inf;
67. **for**(**int** i = 1; i <= m; ++i)
68. **if**(!vst[i])
69. csum = min(csum, sum + es[i].w - maxD[es[i].u][es[i].v]);
70. **return** csum;
71. }

74. **int** main() {
75. **int** T;
76. cin >> T;
77. **for**(**int** t = 1; t <= T; ++t){
78. cin >> n >> m;
79. **for**(**int** i = 1; i <= m; ++i)
80. cin >> es[i].u >> es[i].v >> es[i].w;
82. **int** ret = Kruskal();
83. cout << "Case #" << t << " : ";
84. **if**(ret == -1) cout << "No way\n";
85. **else** **if**(csum==inf) cout << "No second way\n";
86. **else** cout << csum << endl;
87. }
88. **return** 0;
89. }

5 zhuliu 最小树形图 矩阵 POJ 3164 Command Network

1. **typedef**  **long** **long** LL;
2. **const** **int** inf = 0x3f3f3f3f;
3. **const** LL maxn = 110;
5. **int** n, m;
6. **double** G[maxn][maxn];
7. **struct** node {
8. **double** x, y;
9. } point[maxn];
10. **double** getDis(**int** x1, **int** y1, **int** x2, **int** y2) {
11. **return** sqrt( (x1-x2)\*(x1-x2) + (y1-y2)\*(y1-y2) );
12. }
14. **bool** vst[maxn], flag[maxn];
15. **int** pre[maxn];
17. **double** zhuliu(**int** s) {
18. **for**(**int** i = 0; i <= n; ++i) {
19. vst[i] = 0;
20. flag[i] = 0;
21. }
22. **double** ans = 0;
23. **while**(**true**) {
24. // 1找最小入边
25. **for**(**int** i = 1; i <= n; ++i)
26. **if**(i != s && !flag[i]) {
27. G[i][i] = inf, pre[i] = i;
28. **for**(**int** j = 1; j <= n; ++j)
29. **if**(!flag[j] && G[j][i] < G[pre[i]][i])
30. pre[i] = j;
31. **if**(pre[i] == i) **return** -1;// 图不连通，不存在树形图
32. }
33. // 2找环
34. **int** i;
35. **for**(i = 1; i <= n; ++i) {
36. **if**(i != s && !flag[i]) {
37. **int** j = i, cnt = 0;
38. **while**(j != s && pre[j]!=i && cnt <= n)
39. j = pre[j], ++cnt;
40. **if**(j == s || cnt > n) **continue**;
41. **break**;
42. }
43. }
44. **if**(i > n) {
45. **for**(**int** i = 1; i <= n; ++i)
46. **if**(i != s && !flag[i])
47. ans += G[pre[i]][i];
48. **return** ans;
49. }
51. **int** j = i;
53. **for**(**int** i = 0; i <= n; ++i)
54. vst[i] = 0;
56. **do** {
57. ans += G[pre[j]][j], j = pre[j], vst[j] = flag[j] = **true**;
58. } **while**(j != i);
59. flag[i] = **false**;
61. **for**(**int** k = 1; k <= n; ++k)
62. **if**(vst[k]) {
63. **for**(**int** j = 1; j <= n; ++j)
64. **if**(!vst[j]) {
65. **if**(G[i][j] > G[k][j])
66. G[i][j] = G[k][j];
67. **if**(G[j][k] < inf && G[j][k]-G[pre[k]][k] < G[j][i])
68. G[j][i] = G[j][k] - G[pre[k]][k];
69. }
70. }
71. }
72. **return** ans;
73. }
75. **int** main() {
76. **int** x, y;
77. **while**(scanf("%d%d", &n, &m)!=EOF) {
78. **for**(**int** i = 0; i <= n; ++i){
79. **for**(**int** j = 0; j <= n; ++j){
80. G[i][j] = inf;
81. }
82. }
83. **for**(**int** i = 1; i <= n; ++i)
84. scanf("%lf%lf", &point[i].x, &point[i].y);
85. **for**(**int** i = 1; i <= m; ++i) {
86. scanf("%d%d", &x, &y);
87. G[x][y] = getDis(point[x].x, point[x].y, point[y].x, point[y].y);
88. }
89. **double** ans = zhuliu(1);
90. **if**(ans==-1)
91. printf("poor snoopy\n");
92. **else**
93. printf("%.2f\n",ans);
94. }
95. **return** 0;
96. }

6 zhuliu 最小树形图 edge UVA 11183 Teen Girl Squad

1. **const** **int** inf = 0x3f3f3f3f;
2. **const** **int** maxn = 1010;
3. **int** mp[maxn][maxn];
4. **int** pre[maxn], belong[maxn], vst[maxn], in[maxn];
6. **struct** Edge {
7. **int** from, to, val;
8. }edge[40010];
10. **int** zhuliu(**int** root, **int** n, **int** m) {
11. **int** ret = 0, u, v;
12. **while**(1) {
13. // 1找最小入边
14. **for**(**int** i=0; i<n; i++) in[i] = inf;
15. **for**(**int** i=0; i<m; i++)
16. **if**(edge[i].from != edge[i].to && edge[i].val < in[edge[i].to]) {
17. pre[edge[i].to] = edge[i].from;
18. in[edge[i].to] = edge[i].val;
19. }
20. **for**(**int** i = 0; i < n; ++i){
21. **if**(i == root)
22. **continue**;
23. **if**(in[i] == inf)
24. **return** inf;// 图不连通，不存在树形图
25. }
26. // 2找环
27. **int** cnt = 0;// 缩点后节点个数
28. memset(belong, -1, **sizeof** belong);
29. memset(vst, -1, **sizeof** vst);
30. in[root] = 0;
31. **for**(**int** i=0; i < n; ++i) {// 先给环上的点编号
32. ret += in[i];//注意：ret只在zhuliu开始时清空一次，在整个过程中都是累加的
33. v = i;
34. **while**(vst[v] != i && belong[v] == -1 && v != root) {// 如果没有环，会因为v == root而结束，有环时v会指向环的起点
35. vst[v] = i;
36. v = pre[v];
37. }
38. // 环之前的点筛除是靠前推到根节点，环之后的点筛除是靠前推到上一个环
39. **if**(v != root && belong[v] == -1) {
40. **for**(**int** u = pre[v]; u != v; u = pre[u])
41. belong[u] = cnt;
42. belong[v] = cnt ++;
43. }
44. }
45. **if**(cnt == 0)// 没有环了， 找到最小树形图！
46. **break**;
48. // 3缩点，重新编号
49. **for**(**int** i = 0; i < n; ++i)
50. **if**(belong[i] == -1)
51. belong[i] = cnt++;
52. **for**(**int** i = 0; i < m; ) {
53. v = edge[i].to;
54. edge[i].from = belong[edge[i].from];
55. edge[i].to = belong[edge[i].to];
56. // 对于单点到单点，环到单点的权值，因为已经在找环的时候累加过，所以这么减会把权值减成0，恰好符合要求——不然就重复相加了
57. // 对于单点到环，或者环到环，因为该点的一条入边已经确定，且只能有一条，所以权值=原来的权值减去在环上的入边的权值
58. // 这儿有一个隐含条件：环上的入边的权值一定小于环外入边的权值，如若不然，在求解in数组进而求环的时候，根本不会把这个环认为是环
59. **if**(edge[i].from != edge[i].to) edge[i++].val -= in[v];
60. **else** swap(edge[i], edge[--m]);
61. }
62. n = cnt;
63. root = belong[root];
64. }
65. **return** ret;
66. }
68. **int** main() {
69. **int** x, y, z, T, n, m;
70. scanf("%d", &T);
71. **for**(**int** Case = 1; Case <= T; ++Case) {
72. memset(mp, inf, **sizeof**(mp));
73. scanf("%d%d", &n, &m);
74. **for**(**int** i = 0; i < m; ++i) {
75. scanf("%d%d%d", &x, &y, &z);
76. mp[x][y] = min(mp[x][y], z);// 去重
77. }
78. **int** top = 0;
79. **for**(**int** i = 0; i < n; ++i)
80. **for**(**int** j = 0; j < n; ++j)
81. **if**(mp[i][j] != inf) {
82. edge[top].from = i;
83. edge[top].to = j;
84. edge[top++].val = mp[i][j];
85. }
86. **int** ans = zhuliu(0, n, m);
87. printf("Case #%d: ", Case);
88. **if**(ans == inf) printf("Possums!\n");
89. **else** printf("%d\n", ans);
90. }
91. **return** 0;
92. }

7 zhuliu 虚拟根 找根 HDU 2121 Ice\_cream’s world II

1. **const** **int** maxn = 1005;
2. **const** **int** maxm = 10005;
3. **const** **int** inf = 0x3f3f3f3f;
5. **struct** Edge {
6. **int** u,v,cost;
7. };
8. Edge edge[maxm];
9. **int** pre[maxn], id[maxn], visit[maxn], in[maxn], pos;
11. **int** zhuliu(**int** root,**int** n,**int** m) {
12. **int** res = 0,u,v;
13. **while**(**true**) {
14. **for**(**int** i = 0; i < n; i ++) {
15. in[i] = inf;
16. }
17. **for**(**int** i = 0; i < m; i ++) {
18. **if**(edge[i].u != edge[i].v && edge[i].cost < in[edge[i].v]) {
19. pre[edge[i].v] = edge[i].u;
20. in[edge[i].v] = edge[i].cost;
21. **if**(edge[i].u == root) pos = i; // 找根
22. }
23. }
24. **for**(**int** i = 0; i < n; i ++) {
25. **if**(i != root && in[i] == inf)
26. **return** -1;
27. }
28. **int** tn = 0;
29. memset(id,-1,**sizeof** (id));
30. memset(visit,-1,**sizeof**(visit));
31. in[root] = 0;
32. **for**(**int** i = 0; i < n; i ++) {
33. res += in[i];
34. v = i;
35. **while**(visit[v] != i && id[v] == -1 && v != root) {
36. visit[v] = i;
37. v = pre[v];
38. }
39. **if**(v != root && id[v] == -1) {
40. **for**(u = pre[v]; u != v; u = pre[u])
41. id[u] = tn;
42. id[v] = tn ++;
43. }
44. }
45. **if**(tn == 0) **break**;
46. **for**(**int** i = 0; i < n; i ++) {
47. **if**(id[i] == -1)
48. id[i] = tn ++;
49. }
50. **for**(**int** i = 0; i < m; i ++) {
51. v = edge[i].v;
52. edge[i].u = id[edge[i].u];
53. edge[i].v = id[edge[i].v];
54. **if**(edge[i].u != edge[i].v)
55. edge[i].cost -= in[v];
56. //else swap(edge[i],edge[-- m]);
57. }
58. n = tn;
59. root = id[root];
60. }
61. **return** res;
62. }
64. **int** main() {
65. **int** n,m;
66. **while**(~scanf("%d%d",&n,&m)) {
67. **int** sum = 0;
68. **for**(**int** i = 0; i < m; i ++) {
69. scanf("%d%d%d",&edge[i].u,&edge[i].v,&edge[i].cost);
70. sum += edge[i].cost;
71. edge[i].u ++;
72. edge[i].v ++;
73. }
74. sum ++;
75. **for**(**int** i = 0; i < n; i ++) {
76. edge[i + m].u = 0;
77. edge[i + m].v = i + 1;
78. edge[i + m].cost = sum;
79. }
80. **int** ans = zhuliu(0,n + 1,n + m);
81. **if**(ans == -1 || ans >= 2 \* sum ) printf("impossible\n");
82. **else** printf("%d %d\n",ans - sum,pos - m);
83. printf("\n");
84. }
85. **return** 0;
86. }

7 zhuliu 虚拟根 找根2 HDU 2121 Ice\_cream’s world II

1. **const** **int** MAXN = 1e3 + 10;
2. **typedef** **long** **long** LL;
3. **int** id[MAXN], vis[MAXN], pre[MAXN], pos;
4. LL INF = 1e17, d[MAXN];
5. **struct** node {
6. **int** u, v, cost;
7. } edge[MAXN \* MAXN];
9. LL zhuliu(**int** root, **int** V, **int** E) {
10. LL res = 0;
11. **while**(**true**) {
12. **for**(**int** i = 0 ; i < V ; i++) {
13. d[i] = INF;
14. }
15. **for**(**int** i = 0 ; i < E ; i++) {
16. **int** u = edge[i].u, v = edge[i].v;
17. **if**(u != v && d[v] > edge[i].cost) {
18. d[v] = edge[i].cost;
19. pre[v] = u;
20. **if**(u == root) {
21. pos = i;    //记录位置    除了这里不一样 其他地方都是朱刘算法的模板
22. }
23. }
24. }
25. **for**(**int** i = 0 ; i < V ; i++) {
26. **if**(d[i] == INF && i != root) {
27. **return** -1;
28. }
29. }
30. **int** cont = 0;
31. memset(id, -1, **sizeof**(id));
32. memset(vis, -1, **sizeof**(vis));
33. d[root] = 0;
34. **for**(**int** i = 0 ; i < V ; i++) {
35. **int** v = i;
36. res += d[i];
37. **while**(id[v] == -1 && vis[v] != i && v != root) {
38. vis[v] = i;
39. v = pre[v];
40. }
41. **if**(id[v] == -1 && v != root) {
42. **for**(**int** u = pre[v] ; u != v ; u = pre[u]) {
43. id[u] = cont;
44. }
45. id[v] = cont++;
46. }
47. }
48. **if**(!cont) {
49. **break**;
50. }
51. **for**(**int** i = 0 ; i < V ; i++) {
52. **if**(id[i] == -1) {
53. id[i] = cont++;
54. }
55. }
56. **for**(**int** i = 0 ; i < E ; i++) {
57. **int** u = edge[i].u, v = edge[i].v;
58. edge[i].u = id[u];
59. edge[i].v = id[v];
60. **if**(id[u] != id[v]) {
61. edge[i].cost -= d[v];
62. }
63. }
64. V = cont;
65. root = id[root];
66. }
67. **return** res;
68. }
70. **int** main() {
71. **int** n, m;
72. **while**(~scanf("%d %d", &n, &m)) {
73. LL sum = 0;
74. **for**(**int** i = 0 ; i < m ; i++) {
75. scanf("%d %d %d", &edge[i].u, &edge[i].v, &edge[i].cost);
76. edge[i].u++, edge[i].v++;
77. sum += edge[i].cost;
78. }
79. sum++;   //边权大于总权值
80. **for**(**int** i = m ; i < n + m ; i++) {
81. edge[i].u = 0;    //0为虚拟节点
82. edge[i].v = i - m + 1;
83. edge[i].cost = sum;
84. }
85. LL res = zhuliu(0, n + 1, n + m);     //n + 1 个点  n + m 条边
86. **if**(res == -1 || res - sum >= sum) {    //要是res - sum < sum 的话就说明 0的出度为1  说明原图是连通图
87. printf("impossible\n\n");
88. } **else** {
89. printf("%lld %d\n\n", res - sum, pos - m);
90. }
91. }
92. }

8 zhuliu 虚拟根 找根2 HDU 4009 Transfer water

1. **const** **int** maxn = 1e3 + 10;
2. **int** n, m;
3. **int** id[maxn], vis[maxn], pre[maxn];
4. **int** INF = 0x3f3f3f3f, d[maxn];
5. **struct** node {
6. **int** u, v, cost;
7. } edge[maxn \* maxn];
9. **int** zhuliu(**int** root, **int** V, **int** E) {
10. **int** res = 0;
11. **while**(**true**) {
12. **for**(**int** i = 0 ; i < V ; i++) {
13. d[i] = INF;
14. }
15. **for**(**int** i = 0 ; i < E ; i++) {
16. **int** u = edge[i].u, v = edge[i].v;
17. **if**(u != v && d[v] > edge[i].cost) {
18. d[v] = edge[i].cost;
19. pre[v] = u;
20. }
21. }
22. **for**(**int** i = 0 ; i < V ; i++) {
23. **if**(d[i] == INF && i != root) {
24. **return** -1;
25. }
26. }
27. **int** cont = 0;
28. memset(id, -1, **sizeof**(id));
29. memset(vis, -1, **sizeof**(vis));
30. d[root] = 0;
31. **for**(**int** i = 0 ; i < V ; i++) {
32. **int** v = i;
33. res += d[i];
34. **while**(id[v] == -1 && vis[v] != i && v != root) {
35. vis[v] = i;
36. v = pre[v];
37. }
38. **if**(id[v] == -1 && v != root) {
39. **for**(**int** u = pre[v] ; u != v ; u = pre[u]) {
40. id[u] = cont;
41. }
42. id[v] = cont++;
43. }
44. }
45. **if**(!cont) {
46. **break**;
47. }
48. **for**(**int** i = 0 ; i < V ; i++) {
49. **if**(id[i] == -1) {
50. id[i] = cont++;
51. }
52. }
53. **for**(**int** i = 0 ; i < E ; i++) {
54. **int** u = edge[i].u, v = edge[i].v;
55. edge[i].u = id[u];
56. edge[i].v = id[v];
57. **if**(id[u] != id[v]) {
58. edge[i].cost -= d[v];
59. }
60. }
61. V = cont;
62. root = id[root];
63. }
64. **return** res;
65. }
67. **int** x[maxn], y[maxn], z[maxn];
68. **int** X, Y, Z;
70. **int** main(){
71. **while**(~scanf("%d%d%d%d",&n,&X,&Y,&Z)){
72. **if**(!n && !X && !Y && !Z)
73. **break**;
74. m = 0;
75. **for**(**int** i = 1; i <= n; ++i)
76. scanf("%d%d%d", &x[i], &y[i], &z[i]);
77. **for**(**int** i = 1; i <= n; ++i){
78. **int** k, v;
79. scanf("%d", &k);
80. **for**(**int** j = 1; j <= k; ++j){
81. scanf("%d", &v);
82. edge[m].u = i;
83. edge[m].v = v;
84. edge[m].cost = ( abs(x[i] - x[v]) + abs(y[i] - y[v]) + abs(z[i] - z[v])) \* Y;
85. **if**(z[i] < z[v])
86. edge[m].cost += Z;
87. m++;
88. }
89. }
90. **for**(**int** i = 1; i <= n; ++i){
91. edge[m].u = 0;
92. edge[m].v = i;
93. edge[m].cost = z[i] \* X;
94. m++;
95. }
96. **int** res = zhuliu(0, n + 1, m);     //n + 1 个点  m 条边
97. printf("%d\n", res);
98. }
99. }