Laboratory work # 8

Student: *Zhanghao Chen*Student ID: 21321205  
Timus Name: *BennyChan*

Mail: [1824362950@qq.com](mailto:1824362950@qq.com)

Problem # 1160 *Network*

Screenshot from Timus:



Explanation of algorithm:

I implement Kruskal's algorithm for finding the minimum spanning tree of a graph. The algorithm works by first sorting the edges by weight, and then selecting edges from smallest to largest. If selecting an edge would create a cycle, it is not selected. The algorithm continues until (*n*-1) edges have been selected. It is worth noting that the algorithm uses a union-find data structure to detect cycles.

Computational complexity of algorithm:

where *E* is the number of edges and *N* is the number of nodes

Source code:

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
4. **int** fa[1010];
6. **struct** node3{
7. **int** x,y,len;
8. }p[15010];
10. **bool** cmp(node3 a,node3 b){
11. **return** a.len<b.len;
12. }
14. **int** find(**int** x){
15. **if**(fa[x]!=x){
16. fa[x]=find(fa[x]);
17. }
18. **return** fa[x];
19. }
21. **int** main(){
22. **int** n,m,i,j,s=0,maxz=0;
23. cin>>n>>m;
25. **for**(i=1;i<=n;i++){
26. fa[i]=i;
27. }
29. **for**(i=1;i<=m;i++){
30. cin>>p[i].x>>p[i].y>>p[i].len;
31. }
33. sort(p+1,p+m+1,cmp);
35. **for**(i=1;i<=m&&s<n-1;i++){
36. **int** tx=find(p[i].x);
37. **int** ty=find(p[i].y);
38. **if**(tx!=ty){
39. fa[tx]=ty;
40. s++;
41. maxz=max(maxz,p[i].len);
42. }
43. }
45. cout<<maxz<<endl;
46. cout<<i-1<<endl;
48. **for**(j=1;j<i;j++){
49. cout<<p[j].x<<" "<<p[j].y<<endl;
50. }
52. **return** 0;
53. }

Problem # 1162 *Currency Exchange*

Screenshot from Timus:



Explanation of algorithm:

I utilize the shortest path faster algorithm (SPFA) to detect whether foreign exchange arbitrage is possible. Given a list of currency exchange rates and transaction fees, the algorithm determines if there exists a positive cycle in the graph of currency exchange rates.

Computational complexity of algorithm:

where *M* is the number of edges and *N* is the number of nodes

Source code:

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
4. #define maxn 1003
5. #define INF 0x3f3f3f3f
6. #define eps 1e-7
8. **struct** Edge
9. {
10. **int** fromVertex, toVertex;
11. **double** exchangeRate, transactionFee;
12. Edge(**int** from, **int** to, **double** rate, **double** fee)
13. {
14. fromVertex = from;
15. toVertex = to;
16. exchangeRate = rate;
17. transactionFee = fee;
18. }
19. };
21. vector<Edge> ways[maxn];
22. **double** dis[maxn];
23. **int** cnt[maxn];
24. **bool** ever[maxn];
25. **int** n, m, s;
26. **double** v;
28. **void** initialize()
29. {
30. memset(ways, 0, **sizeof**(ways));
31. memset(cnt, 0, **sizeof**(cnt));
32. memset(ever, 0, **sizeof**(ever));
33. **for**(**int** i = 1; i <= n; ++i)
34. {
35. dis[i] = 0;
36. }
37. **for**(**int** i = 0; i < m; ++i)
38. {
39. **int** fromVertex, toVertex;
40. **double** toRate1, toFee1, toRate2, toFee2;
41. cin >> fromVertex >> toVertex;
42. cin >> toRate1 >> toFee1 >> toRate2 >> toFee2;
43. ways[fromVertex].push\_back(Edge(fromVertex, toVertex, toRate1, toFee1));
44. ways[toVertex].push\_back(Edge(toVertex, fromVertex, toRate2, toFee2));
45. }
46. }
48. **bool** SPFA()
49. {
50. ever[s] = 1;
51. cnt[s]++;
52. queue<**int**> q;
53. q.push(s);
54. dis[s] = v;
55. **while**(!q.empty())
56. {
57. **int** cur = q.front();
58. q.pop();
59. **int** len = ways[cur].size();
60. ever[cur] = 0;
61. **for**(**int** i = 0; i < len; ++i)
62. {
63. **int** y = ways[cur][i].toVertex;
64. **if**((dis[cur] - ways[cur][i].transactionFee)\*ways[cur][i].exchangeRate - dis[y] > eps)
65. {
66. dis[y] = (dis[cur] - ways[cur][i].transactionFee)\*ways[cur][i].exchangeRate;
67. **if**(!ever[y])
68. {
69. q.push(y);
70. ever[y] = 1;
71. cnt[y]++;
72. **if**(cnt[y] >= n)
73. **return** **true**;
74. }
75. }
76. }
77. }
78. **return** **false**;
79. }
81. **int** main()
82. {
83. cin >> n >> m >> s >> v;
84. initialize();
85. **bool** isPositiveCycle = SPFA();
86. **if**(isPositiveCycle)
87. printf("YES");
88. **else**
89. printf("NO");
90. **return** 0;
91. }