Given a connected and undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

O(ElogE) or O(ElogV). Sorting of edges takes O(ELogE) time. After sorting, we iterate through all edges and apply find-union algorithm. The find and union operations can take atmost O(LogV) time. So overall complexity is O(ELogE + ELogV) time. The value of E can be atmost O(V2), so O(LogV) are O(LogE) same. Therefore, overall time complexity is O(ElogE) or O(ElogV)

---------------------------------------------------------

1. #include <iostream>
2. #include <vector>
3. #include <cstdlib>
4. #include <search.h>
5. #include <algorithm>
7. **using** **namespace** std;
9. struct subset{
10. int rank;
11. int parent;
12. };
13. struct edge{
14. int u;
15. int v;
16. int w;
17. };
18. vector<edge >e;
19. vector<subset> sub;
20. int mycomp(edge a,edge b){
21. return a.w>b.w;
22. }
23. int mycomp2(edge a,edge b){
24. return a.w<b.w;
25. }
26. int myfind(int u){
27. if(sub[u].parent !=u){
28. return sub[u].parent=myfind(sub[u].parent);
29. }
30. return sub[u].parent;
31. }
32. void myunion(int u,int v){
33. int x=myfind(u);
34. int y=myfind(v);
35. if(sub[x].rank>sub[y].rank){
36. sub[y].parent=x;
37. }else{
38. if(sub[x].rank<sub[y].rank){
39. sub[x].parent=y;
40. }else{
41. sub[x].parent=y;
42. sub[x].rank++;
43. }
44. }
45. }
46. int mst1(int n){
47. for(int i=0;i<=n;i++){
48. subset tmp;
49. tmp.rank=0;
50. tmp.parent=i;
51. sub.push\_back(tmp);
52. }
53. sort(e.begin(),e.end(),mycomp2);//do minimum
54. *//cout<<e[0].w<<" test "<<endl;*
55. int i=0;
56. int sum=0;
57. int counter=0;
58. while(counter!=n){
59. int x=myfind(e[i].u);
60. int y=myfind(e[i].v);
61. if(x!=y){
62. myunion(x,y);
63. sum+=e[i].w;
64. counter++;
65. }
66. i++;
67. }
69. sort(e.begin(),e.end(),mycomp);//do muximum
70. i=0;
71. int sum2=0;
72. counter=0;
73. *// cout<<sum<<" test2"<<endl;*
74. sub.clear();
75. for(int i1=0;i1<=n;i1++){
76. subset tmp;
77. tmp.rank=0;
78. tmp.parent=i1;
79. sub.push\_back(tmp);
80. }
81. while(counter!=n){
82. int x=myfind(e[i].u);
83. int y=myfind(e[i].v);
84. if(x!=y){
85. myunion(x,y);
86. sum2+=e[i].w;
87. counter++;
88. }
89. i++;
90. }
91. return sum+sum2;
92. }
94. int main() {
95. int T;cin>>T;
96. for(int i=0;i<T;i++){
97. int n;cin>>n;
98. e.clear();
99. sub.clear();
100. while(**true**){
101. int u;cin>>u;
102. int v;cin>>v;
103. int w;cin>>w;
104. if(!u && !v && !w){
105. break;
106. }
107. edge tmp;
108. tmp.u=u;
109. tmp.v=v;
110. tmp.w=w;
111. e.push\_back(tmp);
112. }
113. long long int res=mst1(n);
114. cout<<"Case "<<i+1<<": ";
115. if(res%2==0){
116. cout<<res/2<<endl;
117. }else{
118. cout<<res<<"/2"<<endl;
119. }
121. }

124. return 0;
125. }