# Kruskal’s Algorithm

```
1   Algorithm Kruskal(E, cost, n, t)
2   // E is the set of edges in G. G has n vertices. cost[u, v] is the
3   // cost of edge (u, v). t is the set of edges in the minimum-cost
4   // spanning tree. The final cost is returned.
5   {
6       Construct a heap out of the edge costs using Heapify;
7       for i := 1 to n do parent[i] := -1;
8       // Each vertex is in a different set.
9       i := 0; mincost := 0.0;
10      while ((i < n - 1) and (heap not empty)) do
11      {
12          Delete a minimum cost edge (u, v) from the heap
13          and reheapify using Adjust;
14          j := Find(u); k := Find(v);
15          if (j ≠ k) then
16          {
17              i := i + 1;
18              t[i, 1] := u; t[i, 2] := v;
19              mincost := mincost + cost[u, v];
20              Union(j, k);
21          }
22      }
23      if (i ≠ n - 1) then write ("No spanning tree");
24      else return mincost;
25  }
```