```
Algorithm Prim(E, cost, n, t)
// E is the set of edges in G.
// cost[l : n,1 : n] is the cost adjacency matrix of an n vertex graph such that
// cost[i,j] is either a positive real number or ∞ if no edge (i, j) exists.
//A minimum spanning tree is computed and stored as a set of edges in the
// array t[1:n-1,1: 2].
// (t[i,1],t[i,2]) is an ith edge in the minimum-cost spanning tree.
// The final cost is returned.
{
    Let (k, l) be an edge of minimum cost in E;
    mincost := cost[k, l];
    t[l, l] := k; t[1, 2] := l;
    for i := 1 to n do     // Initialize near[ ] array.
        if (cost[i,l] < cost[i,k]) then  // If l is nearer to i than k
            near[i] := l;
        else
            near[i] := k;
    near[k] := near[l] := 0;
    for i:= 2 to n -1 do
        {
        // Find n-2 additional edges for t.
            Let j be an index such that near[j] ≠ 0 and cost[j, near [j]] is
            minimum;
            t [i, 1] := j;
            t[i, 2] := near[j];
            mincost := mincost + cost[j, near [j]];
            near[j] :=0;
            for k :=1 to n do //update near[ ] array
                if (near [k] ≠ 0) and ((cost [k, near [k]) > (cost [k, j])) then
                    near [k] :=j;
        }
    return mincost;
}
```