Algoritmos y Estructuras de Datos
UT8-PD3

Santiago Blanco

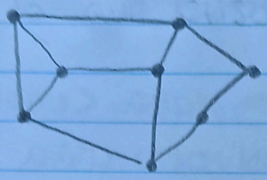
27-06-2025

Ejercicio #1

UT8 - PD3

Es. ①

1. El problema puede ser solucionado encontrando el árbol abarcador de costo mínimo entre todos los vértices (casas).
   ↳ Los algoritmos que permiten realizar esto son: Kruskal y Prim



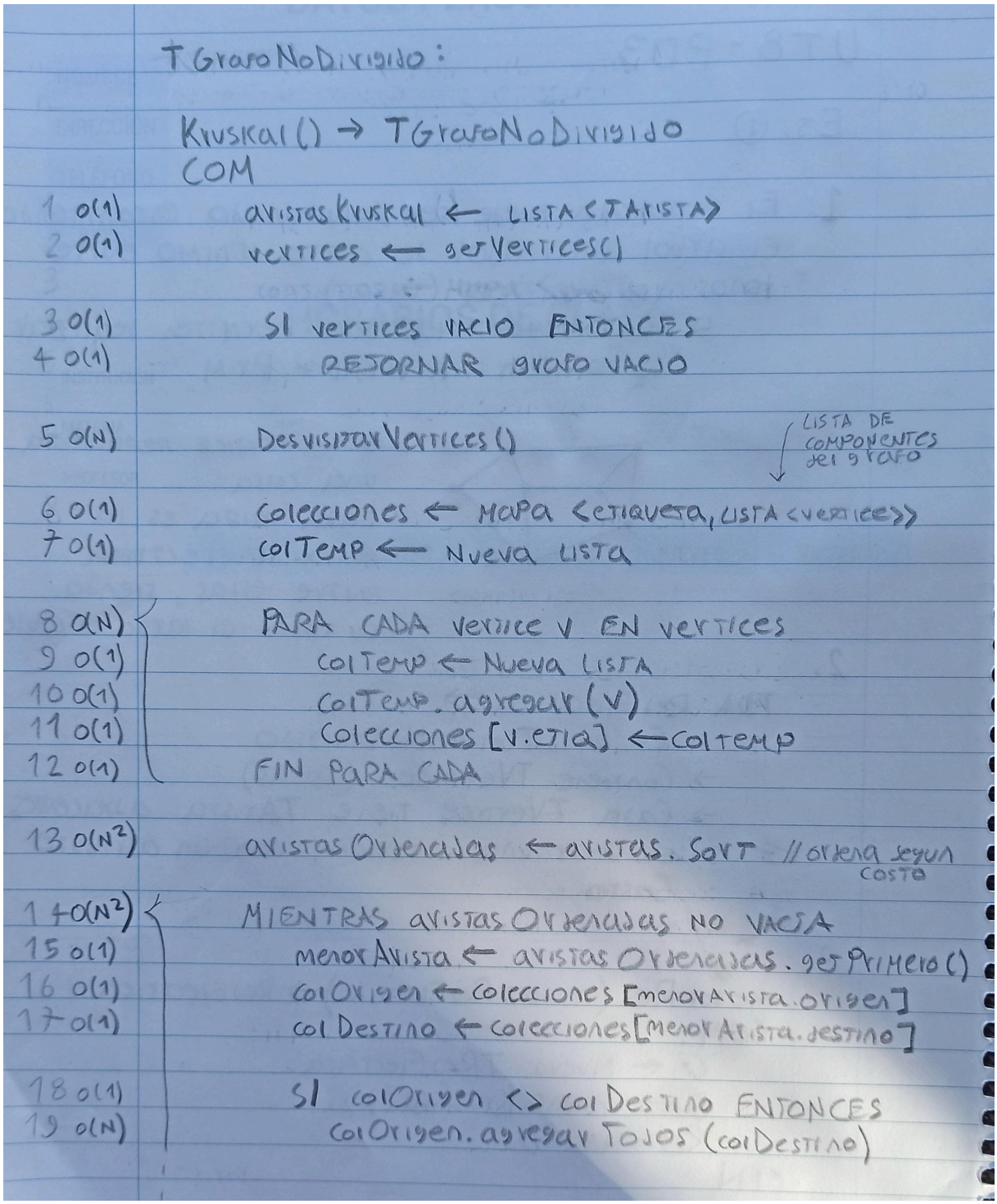
- Cada vertice representa una casa
- Casa arista es la conexión eléctrica entre ellas, siendo su costo el largo del cable.

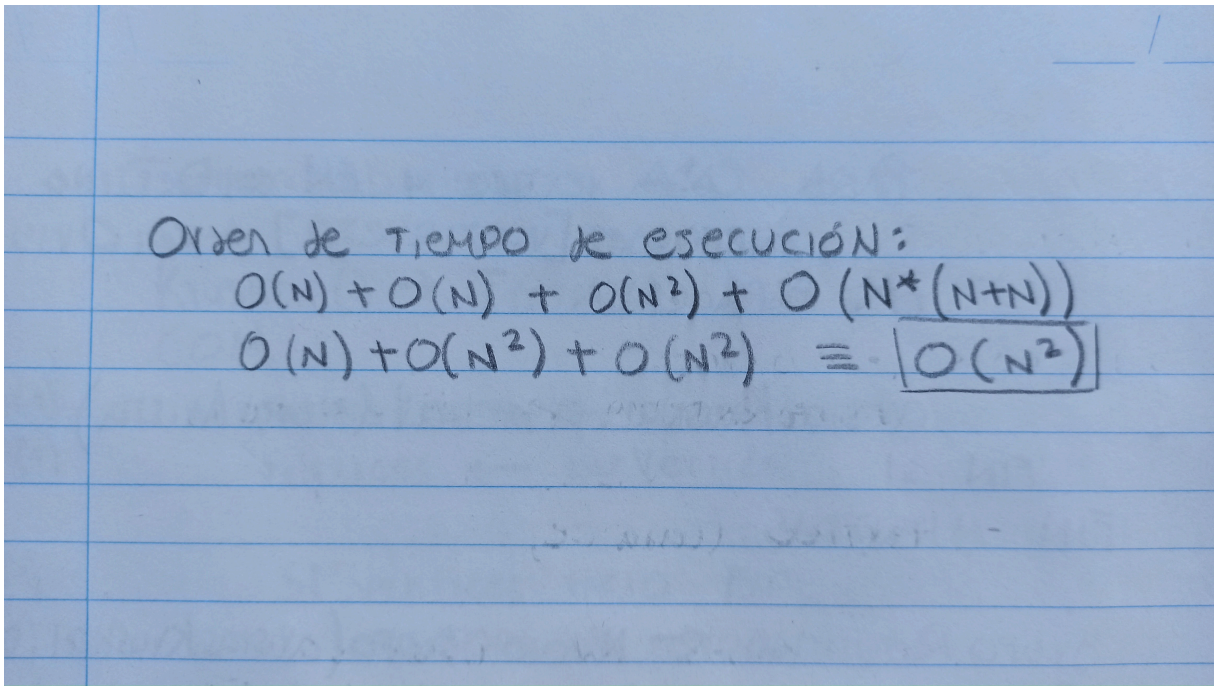
2.

TDA RedElectrica
↳ Es un TGrafoNoDirigido
→ Contiene TVertices (casas)
→ Casa TVertice tiene TAristas adyacentes, es decir una conexión, con un atributo Costo.

3.

```
mejorRed(plano: RedElectrica) -> RedElectrica
COM
    G ← Nuevo TRedElectrica
    G ← plano.Kruskal()
    RETORNAR G
FIN
```


TGrafoNoDirigido:
Kruskal() → TGrafoNoDirigido
COM
1 O(1) aristasKruskal ← LISTA<TARISTA>
2 O(1) vertices ← getVertices()
3 O(1) SI vertices VACIO ENTONCES
4 O(1) REJORNAR grafo VACIO
5 O(N) DesvisitarVertices()
LISTA DE COMPONENTES del GRAFO
6 O(1) colecciones ← MAPA <ETIQUETA, LISTA<VERTICE>>
7 O(1) colTemp ← Nueva LISTA
8 O(N) PARA CADA vertice v EN vertices
9 O(1) colTemp ← Nueva LISTA
10 O(1) colTemp.agregar(v)
11 O(1) colecciones[v.etiq] ← colTemp
12 O(1) FIN PARA CADA
13 O(N²) aristasOrdenadas ← aristas.sort // ordena segun costo
14 O(N²) MIENTRAS aristasOrdenadas NO VACIA
15 O(1) menorArista ← aristasOrdenadas.getPrimero()
16 O(1) colOrigen ← colecciones[menorArista.origen]
17 O(1) colDestino ← colecciones[menorArista.destino]
18 O(1) SI colOrigen <> colDestino ENTONCES
19 O(N) colOrigen.agregarTodos(colDestino)


Orden de tiempo de ejecución:
O(N) + O(N) + O(N²) + O(N*(N+N))
O(N) + O(N²) + O(N²) ≡ O(N²)