Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-1

11 de abril de 2019
Actividad Formativa

Actividad 03

Estructuras nodales: Parte 1

Introducción

Luego de haber derrotado al malvado **Doctor Valdivieso** en la guerra de civilizaciones, su gran aliado y amigo del mal **Thanenzo** ha decidido robar todos los cables del sistema eléctrico de Chile para construir el gran *Olvida-inador*, un artefacto capaz de borrar cualquier lenguaje de programación de la mente de las personas. Es por esto, que **Pynel**, la conocida empresa encargada del sistema de cableado en Chile, ha elegido a los grandes estudiantes del IIC2233 para que modelen la reconstrucción del sistema. Así, se podrá restablecer el sistema eléctrico y hackear la malévola máquina para iniciar su autodestrucción. Para lograrlo, los alumnos implementarán el **DCCableling**.

¡Ayúdalos a reconstruir el sistema!

Figura 1: Logo Pynel Chile

DCCableling

DCCableling es el más moderno sistema de modelación de redes eléctricas creado en el IIC2233 este 2019. Consiste en un programa que permite modelar y analizar el comportamiento de sistemas eléctricos usando **árboles** como estructura de datos interna. Cada sistema eléctrico cuenta con una **Central Generadora** (**nodo raíz**) que provee electricidad a una red jerárquica de instalaciones (**nodos internos**) que consumen energía eléctrica.

Sistemas eléctricos

Cada sistema eléctrico esta formado por distintas **instalaciones** que distribuyen la energía por todo el país. Esta distribución se genera en forma de **árbol** encabezado por **una** Central Generadora, donde cada instalación inferior recibe energía de una única instalación superior[1] y la distribuye solo a instalaciones de grado inferior según lo detallado a continuación:

- **Central Generadora**: Primer nodo del árbol, es la instalación encargada de producir la energía y se conecta a Distribuidoras Regionales.
- **Distribuidora Regional**: Recibe la energía de la Central Generadora y se la distribuye a las Distribuidoras Comunales.
- **Distribuidora Comunal**: Esta instalación distribuye la energía recibida a todas las casas de una comuna.
- **Casa**: Recibe la energía de su Distribuidora Comunal y la consume.

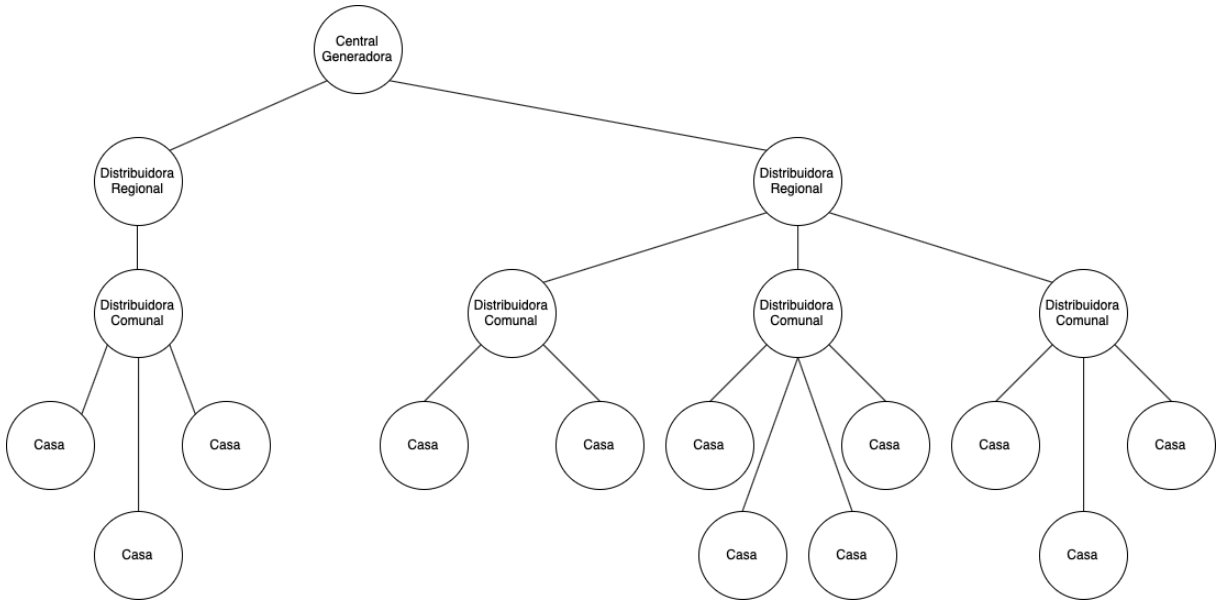

Figura 2: Árbol de ejemplo.

Atributos de las instalaciones

Cada instalación que compone la jerarquía tiene al menos los siguientes atributos:

- `self.tipo`: es un `str` que indica el tipo de instalación que es. Este solo puede ser uno de los indicados anteriormente: `"Generadora"`, `"Distribuidora Regional"`, `"Distribuidora Comunal"` o `"Casa"`.
- `self.region`: es un `str` de la región en la cual se encuentra la instalación.
- `self.comuna`: es un `str` de la comuna en la cual se encuentra ubicada la instalación.
- `self.hijos`: es una `list` que contiene a las instalaciones que dependen **directamente** de la instalación, es decir, aquellas que reciben electricidad de la instalación.

[1] La Central se encarga de producir la energía, por lo tanto no tiene una instalación superior

- `self.consumo`: es un `float` positivo que entrega cuanta energía ocupa la instalación al distribuir la energía o en el caso de las casa, la energía que ellas consumen.
- `self.energia`: es un `float` que corresponde a la energía que recibe la instalación desde su instalación padre al hacer la distribución de energía o en el caso de la Central Generadora[2], la energía que esta genera. Este valor nunca puede ser negativo.

Armar el sistema eléctrico

Para un sistema eléctrico, la primera instalación creada es la Central Generadora y es quien recibe inicialmente cada nueva instalación que se quiera agrega al sistema eléctrico.

Cada instalación en el sistema se comporta de la misma forma cuando recibe una instalación para agregar (o conectar): si recibe una instalación del nivel jerárquico **directamente siguiente** del suyo, entonces lo conecta como un hijo a si misma. Por ejemplo: las Distribuidoras Regionales al recibir una Distribuidora Comunal, se la conecta directamente como hija, en cambio una Casa, no.

En caso contrario, la instalación debe conectase a un nivel inferior. Entonces, la instalación debe escoger un hijo que coincida en un atributo **geográfico** a la nueva instalación y pedirle que agregue la nueva instalación. Todas las Distribuidoras Comunales se conectan a la Distribuidora Regional que coincida con su `self.region`, mientras que las Casas se conectan a la Distribuidora Comunal que coincida con su `self.comuna`[3]. Puede observarse un ejemplo de esto en la siguiente figura:

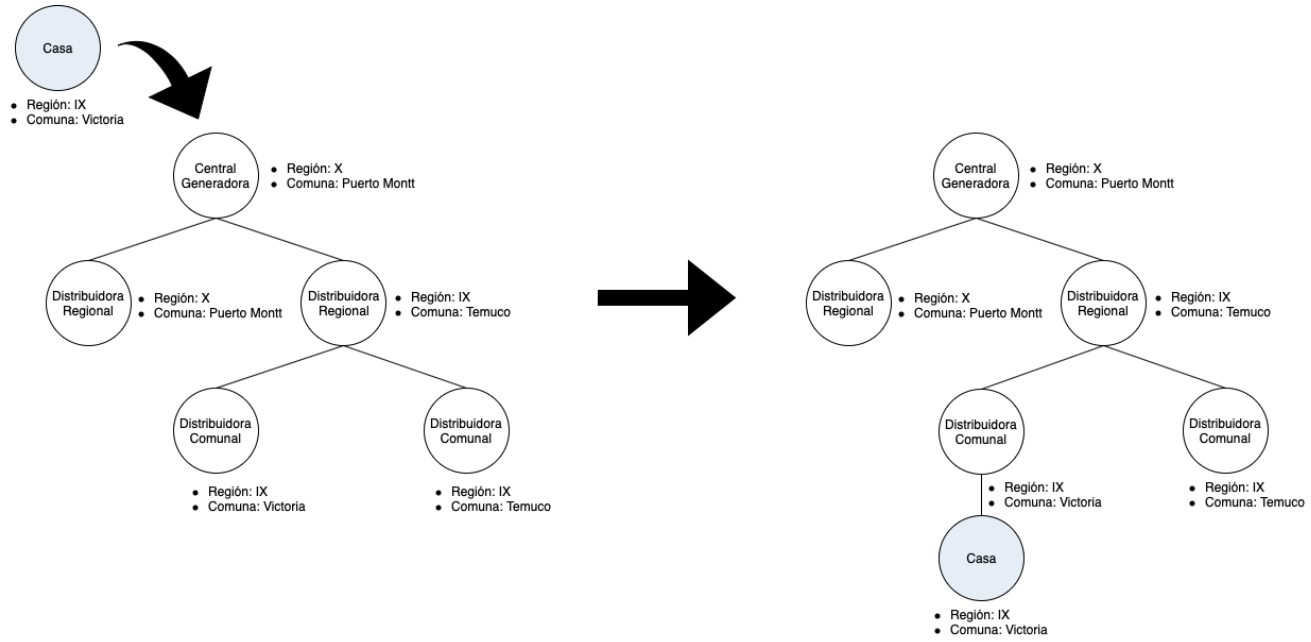

Figura 3: Ejemplo de instalación agregada

Como se muestra en la **Figura 3** se agregó una instalación del tipo Casa entregándosela a la Central Generadora, quien a su vez se la entregó a la Distribuidora Regional correspondiente a su región (`"IX"`), quien posteriormente se la entregó a la Distribuidora Comunal correspondiente a su comuna (`"Victoria"`).

[2]La Central Generadora es la única instalación que comienza con una energía mayor que 0.
[3]Puede asumirse que siempre se encontrará una instalación que coincida para la nueva instalación.

Una vez armado el sistema completo se debe **distribuir la energía** desde la Central Generadora, que es la única que comienza con energía hasta las Casas. Cada instalación debe distribuir a sus hijos mediante la siguiente fórmula:

$$E_{hijo} = \frac{E_{recibida} - Consumo}{N_{hijos}}$$

donde E_{hijo} es la energía a repartir a cada hijo, N_{hijos} la cantidad de hijos y $Consumo$ es el consumo de la instalación actual.

Este método comenzará en la Central Generadora con su propia energía y luego debe avanzar hijo a hijo de forma recursiva hasta las Casas. En caso de que el cálculo de energía a distribuir E_{hijo} resulte un valor negativo, se debe terminar la distribución o en su defecto repartir $E_{hijo} = 0$, ya que la energía que recibe una instalación nunca debe ser negativa.

Para lograr armar el sistema eléctrico de forma correcta, debes completar una clase `Instalacion` que tendrá los siguiente métodos:

- `def agregar_instalacion(self, instalacion)`: recibe una instalación y la agrega a sus conexiones directas o escoge entre una de sus conexiones actuales a la más conveniente, para que esta a su vez repita el proceso.
- `def distribuir_energia(self, energia_recibida)`: es un método que distribuye la energía recibida hacia **cada uno de sus hijos** siguiendo la lógica descrita.

Consultas

Una vez terminado tu árbol, quieres saber dónde están los problemas de energía. Para esto quieres diseñar 3 consultas sobre el sistema:

- `def resumen_sistema(sistema)`: imprime un resumen completo del sistema, con al menos:
  - Ubicación de la Central Generadora.
  - Consumo total del sistema.
  - Cantidad total de instalaciones.
- `def comuna_mayor_gasto(sistema)`: imprime la región y la comuna de la Distribuidora Comunal en la cual se produce un mayor consumo total de energía.
- `def casas_insuficiencia(sistema)`: imprime la cantidad de Casas del sistema completo que reciben menos energía que su consumo.

Entienda el **consumo total** de una instalación como la suma de su `self.consumo` y la suma de los consumos de todos sus descendientes (es decir, no solo sus hijos directos).

Estas consultas deben estar implementadas como funciones normales dentro del `main`. Cada una de ellas recibe como único argumento la Central Generadora (nodo raíz). Finalmente dentro de cada función deberás imprimir el resultado de estas consultas para cada sistema.

Archivos adjuntos a la actividad

Con el enunciado de esta actividad, se adjuntan los archivos `main.py`, `leer_archivos.py` y la carpeta `sistemas`.

Hemos implementado por ti lo necesario para poblar cuatro sistemas eléctricos distintos. `sistemas` contiene archivos CSV que se cargan en `leer_archivos.py`. Este último provee una función para cargar dichos archivos. **No es necesario que revisen en detalle estos archivos.**

Por el otro lado, se te entregó también el archivo `main.py` que es donde deberás completar los dos métodos y las tres funciones de consulta que te han solicitado. Por lo tanto, **es el único archivo que necesitas editar**.

`main.py` contiene ya la clase principal `Instalacion` con su constructor creado y listo para usarse. En el podrás notar como se asignan los atributos de cada instalación, como fueron descritos en la sección de **Atributos de las instalaciones**. También, `main.py` ya contiene el código que inicializa por separado cuatro sistemas eléctricos y llama a funciones con las consultas pedidas. Mediante la función `obtener_sistema` (importada de `leer_archivos.py`) recibe los atributos necesarios para crear cada instalación de un sistema eléctrico completo (a través de una lista de diccionarios). Luego, llama a la función `instanciar_sistema` que recibe estos argumentos y crea el sistema eléctrico mediante los métodos que debes implementar.

Recuerda que tienes completa libertad en además crear las funciones y clases auxiliares **que creas necesarias**.

Caso de prueba

Con el fin de guiarte en tus resultados, **Pynel** con gran esfuerzo encontró en sus arcaicos archivos la correcta modelación del Sistema N^{o} 4 (el último sistema cargado en `main.py`), por lo tanto, los resultados a las consultas referidas a este sistemas son:

```
Resumen del sistema:
     Generadora en Sierra Gorda, Antofagasta
     Consumo total: 1526.0
     Número de instalaciones: 286

Comuna con mayor gasto:
     Comuna: San Joaquín
     Región: Santiago

Casas con energía insuficiente:
     Número de casas: 10
```

Notas

- No olvide que esta actividad es de estructuras de datos con nodos, en particular, árboles. No debería ser difícil modelar el problema de esta manera.
- Todas la instalaciones comenzarán con energía igual a 0, a excepción de la Central Generadora que es quien la produce.
- Puedes asumir que en una misma región nunca habrá más de una Distribuidora Regional, y que en una misma comuna no habrá más de una Distribuidora Comunal.
- No debes cubrir los casos borde donde no se encuentren instalaciones que coincidan geográficamente. El orden en que se agreguen las instalaciones aseguran que siempre se puede encontrar una instalación para agregar una nueva.

Requerimientos

- (3.60 pts) Armar los Sistemas:
  - (2.0 pts) El método `agregar_instalacion` está correctamente implementada.
  - (1.60 pts) El método `distribuir_energia` esta correctamente implementada.
- (2.40 pts) Consultas
  - (0.80 pts) La función `resumen_sistema` está correctamente implementada.
  - (0.80 pts) La función `comuna_mayor_gasto` está correctamente implementada.
  - (0.80 pts) La función `casa_insuficiencia` está correctamente implementada.

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC03/`
- **Hora del último *push*:** 16:30

Auto-evaluación

Como esta corresponde a una actividad formativa, te extendemos la instancia de responder, después de terminada la actividad, una auto-evaluación de tu desempeño. Esta se habilitará a las **16:50 del 11 de abril de 2019** y tendrás plazo para responderla hasta las **23:59 del 12 de abril de 2019**. Puedes acceder al formulario mediante el siguiente enlace:

`https://forms.gle/Fa9nWWdZkqeHqAQ37`

El asistir, realizar la actividad y responder la auto-evaluación otorgará como bonificación al alumno 2 décimas para sumar en su peor actividad sumativa del semestre.