Universidad Nacional Autónoma de México

Facultad de Ciencias

Redes de Computadoras

Tarea

Bonilla Reyes Dafne - 319089660
García Ponce José Camilo - 319210536
González Peñaloza Arturo - 319091193
Main Cerezo Ashael Said - 319260658
Raudry Rico Emilio Arsenio - 318289276

En equipos contesten las siguientes preguntas:

1) **¿Cuál es la diferencia entre una red Doméstica y una LAN? ¿Existe realmente una diferencia?**

   Hablando estrictamente de las definiciones de una red doméstica y una LAN, podemos decir que las redes domésticas son un tipo específico de LAN. Recordemos que una LAN comprende la administración de recursos de dispositivos interconectados en un área geográfica limitada (por ejemplo, un edificio, una casa, una oficina, etc.), mientras que una red doméstica es una LAN residencial, es decir, se establece en una casa. Por esto mismo, podemos decir que se diferencian en términos de propósito y cobertura de área geográfica. (*What Is a LAN? Local Area Network*, n.d.).

2) **Menciona todas las topologías de redes, pon un ejemplo de estas, así como ventajas y desventajas.**

   La topología de red se refiere a la forma en que los dispositivos están conectados entre sí dentro de una red informática, es decir, es el diseño o la estructura de una red.

   - **Point to point**

     Tipo de topología que trabaja sobre la comunicación entre dos nodos, en la que uno es el emisor y el otro es el receptor, sin la intervención de ningún otro dispositivo o nodo intermedio.

     - **Ejemplo:** Conexión directa por cable USB entre una computadora y una impresora.
     - **Ventajas:** Algunas ventajas de este tipo de topología son: baja latencia, ya que la comunicación es directa, requiere de una configuración simple, y no presenta congestión de tráfico, ya que solo se comunican dos dispositivos.
     - **Desventajas:** Algunas desventajas de este tipo de topología son: falta de escalabilidad, ya que si queremos hacer redes grandes, necesitaríamos enlaces

separados entre cada par de dispositivos, además de que esto podría elevar los costos de su mantenimiento.

- **Mesh**

  En una topología de malla, cada dispositivo está conectado a varios dispositivos a través de un canal en particular, lo que permite múltiples rutas para los datos.

  - **Ejemplo:** Una red de trenes en un sistema de transporte. Cada tren está conectado a otros trenes mediante una red de líneas ferroviarias. Esta configuración forma una malla de conexiones. Si un tramo de vía se interrumpe, los trenes aún pueden encontrar rutas alternativas para llegar a su destino.
  - **Ventajas:** Algunas ventajas de este tipo de topología son: robusticidad, fácil diagnóstico de fallas, ya que cada dispositivo está conectado a través de canales dedicados, comunicación rápida.
  - **Desventajas:** Algunas desventajas de este tipo de topología son: instalación y configuración difíciles, además de que el costo es elevado por la gran cantidad de cableado que se necesita.

- **Bus**

  La topología de bus es un tipo de red en el que todos los equipos y dispositivos de red están conectados a un solo canal de comunicación o cable llamado bus. Es bidireccional, cuenta con conexión multipunto y es no robusta, dado que si la red troncal falla, la topología se bloquea.

  - **Ejemplo:** En una pequeña oficina con varias computadoras, todas las computadoras están conectadas a un cable común que actúa como el bus. Cada computadora puede enviar y recibir datos a través de ese cable compartido. Si una computadora desea comunicarse con otra, simplemente envía sus datos al bus, y todas las demás computadoras en la red pueden escuchar y procesar esa información.
  - **Ventajas:** Algunas ventajas de este tipo de topología son: costos bajos de instalación y mantenimiento en comparación a otras topologías, es una tecnología familiar, por lo que su instalación y resolución de problemas son conocidos.
  - **Desventajas:** Algunas desventajas de este tipo de topología son: si el bus falla, todo el sistema se bloquea, agregar nuevos dispositivos a la red la ralentizará, seguridad es baja.

- **Star**

  En esta topología todos los dispositivos están conectados a un solo hub a través de un cable. Este concentrador es el nodo central y todos los demás nodos están conectados al nodo central, que gestiona la comunicación entre ellos.

  - **Ejemplo:** En una oficina con varias computadoras, cada computadora está conectada al nodo central mediante cables Ethernet o conexiones inalámbricas.

    - **Ventajas:** Algunas ventajas de este tipo de topología son: robusticidad, fácil identificación de fallos, bajo costo, ya que utiliza un cable coaxial económico.
    - **Desventajas:** Algunas desventajas de este tipo de topología son: Si falla el hub en el que se basa toda la topología, todo el sistema se caerá, el costo de instalación es alto.
- **Tree**

  Esta topología es una combinación de varias topologías de estrella. Los nodos se organizan jerárquicamente y están conectados a un nodo central de nivel superior, formando una estructura similar a un árbol.

    - **Ejemplo:** En una empresa con varias sucursales, la sede central actúa como nodo raíz con el servidor principal y la infraestructura central. Cada sucursal (A, B, C) se conecta directamente a la sede central, y dentro de cada una hay departamentos (como ventas, recursos humanos, contabilidad) conectados a través de la infraestructura local. La topología en árbol permite que todas las sucursales y departamentos se comuniquen eficientemente con la sede central.
    - **Ventajas:** Algunas ventajas de este tipo de topología son: disminuye la distinción que recorre la señal para llegar a los dispositivos, es fácil agregar nuevos dispositivos a la red, detección y corrección de errores fácilmente.
    - **Desventajas:** Algunas desventajas de este tipo de topología son: Si falla el hub central, el resto del sistema falla, alto costo debido al cableado.
- **Ring**

  En una topología de anillo, forma un anillo que conecta dispositivos exactamente con dos dispositivos vecinos. Se utilizan varios repetidores para la topología de anillo con un gran número de nodos, porque si alguien quiere enviar algunos datos al último nodo de la topología de anillo con 100 nodos, entonces los datos tendrán que pasar por 99 nodos para llegar al nodo número 100. Por lo tanto, para evitar la pérdida de datos, se utilizan repetidores en la red.

    - **Ejemplo:** Una planta de fabricación, las estaciones de trabajo se conectan en secuencia, de modo que los datos viajan en un bucle. Esta configuración distribuye la carga uniformemente, pero si un nodo falla, toda la red se ve afectada, dificultando el diagnóstico de errores.
    - **Ventajas:** Algunas ventajas de este tipo de topología son: transmisión de datos de alta velocidad, minimización de colisión, bajo costo de instalación y ampliación.
    - **Desventajas:** Algunas desventajas de este tipo de topología son: si un nodo falla, puede hacer que el resto falle, poco seguro.
- **Hybrid**

  Es una combinación de dos o más topologías de red, que se utiliza para aprovechar las ventajas de múltiples configuraciones. La topología híbrida se utiliza cuando los nodos pueden adoptar cualquier forma.

- **Ejemplo:** Una empresa puede combinar una topología estrella en la sede central, donde todos los dispositivos están conectados a un switch, con una topología de bus en las sucursales más pequeñas, donde los dispositivos comparten un único cable principal.
- **Ventajas:** Algunas ventajas de este tipo de topología son: flexibilidad, facilidad para ampliar la red.
- **Desventajas:** Algunas desventajas de este tipo de topología son: dificultad para diseñar la red, altos costos de infraestructura. (Jain, 2024)

3) Investiga en que consiste el enrutamiento estático.

El enrutamiento estático también se conoce como enrutamiento no adaptativo, que no cambia la tabla de enrutamiento a menos que el administrador de red la cambie o modifique manualmente. El enrutamiento estático no utiliza algoritmos de enrutamiento complejos y proporciona mayor o mayor seguridad que el enrutamiento dinámico.

El enrutamiento estático tiene como ventajas que no genera sobrecarga en la CPU del router, lo que permite usar dispositivos más económicos, mejora la seguridad al ser controlado solo por un administrador y no consume ancho de banda entre routers. Sin embargo, sus desventajas incluyen la complejidad en redes grandes, donde los administradores deben configurar manualmente cada ruta, y la necesidad de un conocimiento detallado de la topología, lo que puede dificultar el trabajo de nuevos administradores.

4) Investiga en qué consiste el enrutamiento dinámico.

El enrutamiento dinámico es el método por el cual los routers intercambian información automáticamente utilizando **Protocolos de Enrutamiento Dinámico**. Entre los protocolos de enrutamiento están RIP (routing information protocol), IGRP (Interior Gateway Protocol), EIGRP (Enhanced Interior Gateway Protocol), OSPF (Open Shortest Path First) y IS-IS (Intermediate System to Intermediate System).

Entre las cosas que puede hacer un router configurado con un protocolo anterior están:

- Procesar las actualizaciones que recibe de otros routers vecinos que usan el mismo tipo de protocolo.
- Aprender sobre redes remotas por medio de actualizaciones que mandan routers vecinos.
- Determinar la mejor ruta a una red remota por medio de un algoritmo (si es que existen múltiples rutas).
- Actualizar sus rutas cuando ocurren cambios en la topología de la red.

Los routers configurados con un enrutamiento dinámico tienen la ventaja de ser más adaptables dado que pueden reaccionar de forma automática a cambios en la red. Además, tienen un tiempo de convergencia más rápido y una mejor escalabilidad en redes grandes.

5) Investiga y menciona de qué maneras se puede asignar de manera automática la IP a una zona.

Entre las principales maneras en las que se puede asignar de manera automática una IP a una zona se encuentran:

- **Protocolo de configuración dinámica de host (DHCP):** este es el método más común, automáticamente asigna direcciones IP a los dispositivos de una red. A estos dispositivos se les asigna una dirección IP dinámica en la máscara de subred correcta.. Entre los beneficios de este protocolo se encuentran una configuración confiable de direcciones IP y una administración de redes reducida
- **SLAAC (Stateless Address Autoconfiguration)**: es un mecanismo que permite a cada host de la red auto-configurar una dirección IPv6 única sin que ningún dispositivo lleve la cuenta de qué dirección se asigna a cada nodo. Los routers envían mensajes que permiten a los dispositivos generar su propia dirección basada en la red en la que se encuentran.
- **Zero Configuration Networking:** conjunto de tecnologías que permiten que dispositivos en una red se descubran entre ellos sin necesidad de configuración manual o servidores. Permite asignar automáticamente direcciones IP, convertir nombres de host a direcciones IP y localizar automáticamente servicios de red.

6) **De su región 1 Gráfica será el área total, a partir de esta genera 4 sub gráficas *CONEXAS de por lo menos 3 nodos*, posteriormente pon pesos a tus aristas (las rutas velas como aristas) de tal manera que quede como la gráfica de España, obtén los árboles de peso mínimo, obtén por lo menos 2 árboles generadores, el árbol generador de peso mínimo (pueden existir más de 1 en ese caso pon otro más), así mismo menciona que problema ves con ese árbol así mismo una propuesta para solucionarlo. (Esto se hace con las 5 gráficas).**

Para obtener el árbol generador de peso mínimo usamos el algoritmo de Kruskal (primero ordenar las aristas con base en su peso, luego tomar la arista de menor peso, revisar si agregando esa arista al árbol se genera un ciclo, si no lo genera agregarla al árbol, de otro modo no agregarla y repetir esto hasta tener *n-1* aristas en el árbol, con *n* el número de vértices del árbol) y para los árboles generadores solo quitamos y agregamos aristas al árbol generador de peso mínimo. Para calcular el árbol generador de peso mínimo usamos el siguiente código:

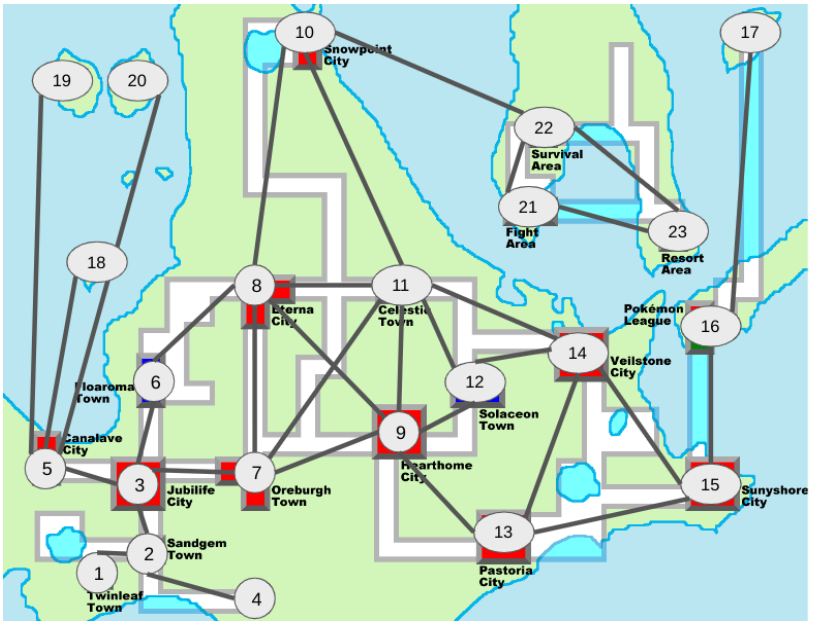
```
# Algoritmo de Kruskal para obtener el árbol generador mínimo de una gráfica
# Métodos para obtener el árbol generador mínimo de una gráfica.

# Método para obtener el árbol generador mínimo de una gráfica
# Usando el algoritmo de Kruskal
def arbol_generado_peso_minimo(aristas, n):
    # Paso 1: Ordenar las aristas de la gráfica
    aristas_ordenas = sorted(aristas, key=lambda x: x[2])
    # Árbol, conjunto de aristas
    arbol = []
    cantidad = 0
    # Union-Find
    uf = UnionFind(n)
    # Repetir los siguientes pasos hasta que el árbol tenga n-1 aristas
    while cantidad < n-1:
```

```
        # Paso 2: Tomar la arista de menor peso y agregarla al árbol
         arista = aristas_ordenas.pop(0)
         vertice1 = arista[0]-1
         vertice2 = arista[1]-1
        # Paso 3: Revisar si la arista forma un ciclo con el árbol, usando
        Union-Find
         if not uf.union(vertice1, vertice2):
             arbol.append(arista)
             print("Se agregó la arista", arista)
             cantidad += 1
         else:
             # Si forma un ciclo, se elimina la arista
             print("Se rechazó la arista", arista)
    return arbol

# Clase para implementar Union-Find
class UnionFind:
    # Constructor
    def __init__(self, n):
        # Al inicio cada nodo es su propio padre
        self.padre = [i for i in range(n)]
        # Inicializar rango
        self.rango = [0 for i in range(n)]

    # Método para encontrar el representante del conjunto
    def find(self, x):
        if self.padre[x] != x:
            self.padre[x] = self.find(self.padre[x])
        return self.padre[x]

    # Método para unir dos conjuntos
    def union(self, x, y):
        raiz_x = self.find(x)
        raiz_y = self.find(y)
        # Si estan en el mismo conjunto, se formaria un ciclo
        if raiz_x == raiz_y:
            return True
        # Union por rango
        if self.rango[raiz_x] > self.rango[raiz_y]:
            self.padre[raiz_y] = raiz_x
        else:
            self.padre[raiz_x] = raiz_y
            if self.rango[raiz_x] == self.rango[raiz_y]:
                self.rango[raiz_y] += 1
        return False


aristas_region = [(1,2,201), (2,3,203), (2,4,660), (3,5,218), (3,6,204),
(3,7,203),
                  (5,18,200), (5,19,400), (5,20,400), (6,8,205), (7,8,206),
(7,9,415),
                  (7,11,418), (8,9,419), (8,10,644), (8,11,211), (9,11,419),
(9,12,209),
                  (9,13,212), (10,11,644), (10,22,200), (11,12,210),
(11,14,425),
                  (12,14,215), (13,14,427), (13,15,435), (14,15,436),
(15,16,223),
                  (16,17,400), (21,22,225), (21,23,459), (22,23,683)
                  ]
```

```
"""
arbol_region = arbol_generado_peso_minimo(aristas_region, 23)
print("Aristas seleccionadas")
for arista in arbol_region:
    print(arista)
print("Aristas no seleccionadas")
for arista in aristas_region:
    if arista not in arbol_region:
        print(arista)
"""

aristas_subgrafica_1 = [(1,2,201), (2,3,660)]

"""
arbol_subgrafica_1 = arbol_generado_peso_minimo(aristas_subgrafica_1, 3)
print("Aristas seleccionadas")
for arista in arbol_subgrafica_1:
    print(arista)
print("Aristas no seleccionadas")
for arista in aristas_subgrafica_1:
    if arista not in arbol_subgrafica_1:
        print(arista)
"""

aristas_subgrafica_2 = [(1,2,206), (1,3,415), (1,4,418), (2,3,419),
(2,4,211), (3,4,419)]

"""
arbol_subgrafica_2 = arbol_generado_peso_minimo(aristas_subgrafica_2, 4)
print("Aristas seleccionadas")
for arista in arbol_subgrafica_2:
    print(arista)
print("Aristas no seleccionadas")
for arista in aristas_subgrafica_2:
    if arista not in arbol_subgrafica_2:
        print(arista)
"""

aristas_subgrafica_3 = [(1,3,200), (2,3,225), (2,4,459), (3,4,683)]

"""
arbol_subgrafica_3 = arbol_generado_peso_minimo(aristas_subgrafica_3, 4)
print("Aristas seleccionadas")
for arista in arbol_subgrafica_3:
    print(arista)
print("Aristas no seleccionadas")
for arista in aristas_subgrafica_3:
    if arista not in arbol_subgrafica_3:
        print(arista)
"""

aristas_subgrafica_4 = [(1,2,427), (1,3,435), (2,3,436), (3,4,223)]

"""
arbol_subgrafica_4 = arbol_generado_peso_minimo(aristas_subgrafica_4, 4)
print("Aristas seleccionadas")
for arista in arbol_subgrafica_4:
    print(arista)
```

```
print("Aristas no seleccionadas")
for arista in aristas_subgrafica_4:
    if arista not in arbol_subgrafica_4:
        print(arista)
"""

aristas_ejercicio_7 = [(1,2,201), (2,4,660), (1,3,30), (1,5,29), (1,6,45),
(1,7,48),
                       (4,8,60), (2,9,25), (2,10,10), (2,11,40), (2,12,42),
(2,13,46),
                       (9,10,10), (11,12,12)]

"""
arbol_ejercicio_7 = arbol_generado_peso_minimo(aristas_ejercicio_7, 13)
print("Aristas seleccionadas")
for arista in arbol_ejercicio_7:
    print(arista)
print("Aristas no seleccionadas")
for arista in aristas_ejercicio_7:
    if arista not in arbol_ejercicio_7:
        print(arista)
"""
```

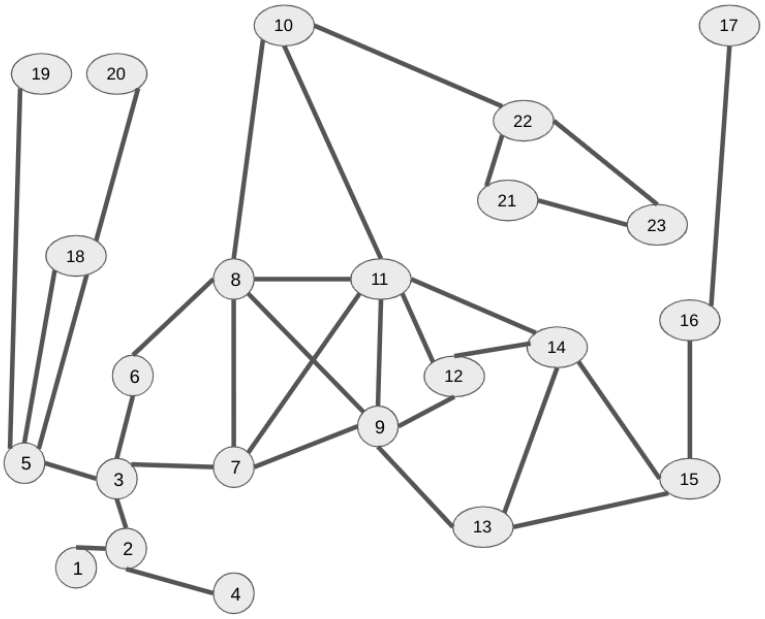
Entonces, esta es la gráfica de nuestra región:



Los pesos de las aristas serán la suma de las rutas por las que pasa y en otros casos serán inventados (como en las islas)

Observación: 17, 19 y 20 no tiene casitas pero aun así las contamos, ya que tiene puertitos

Y con pesos se ve así:

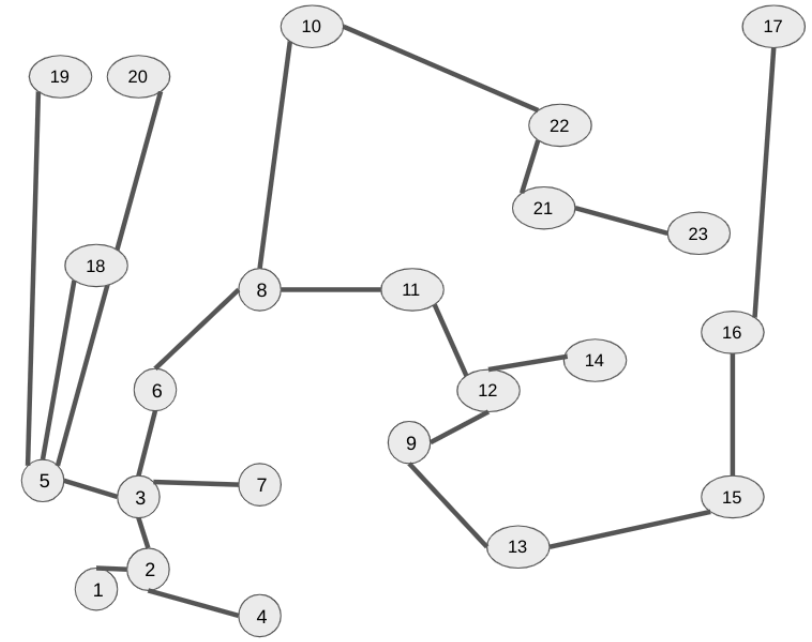


Pesos:

1-2: 201	11-12: 210
2-3: 203	11-14: 425
2-4: 660	12-14: 215
3-5: 218	13-14: 427
3-6: 204	13-15: 435
3-7: 203	14-15: 436
5-18: 200	15-16: 223
5-19: 400	16-17: 400
5-20: 400	21-22: 225
6-8: 205	21-23: 459
7-8: 206	22-23: 683
7-9: 415	----------
7-11: 418	----------
8-9: 419	----------
8-10: 644	----------
8-11: 211	----------
9-11: 419	----------
9-12: 209	----------
9-13: 212	----------
10-11: 644	----------
10-22: 200	----------

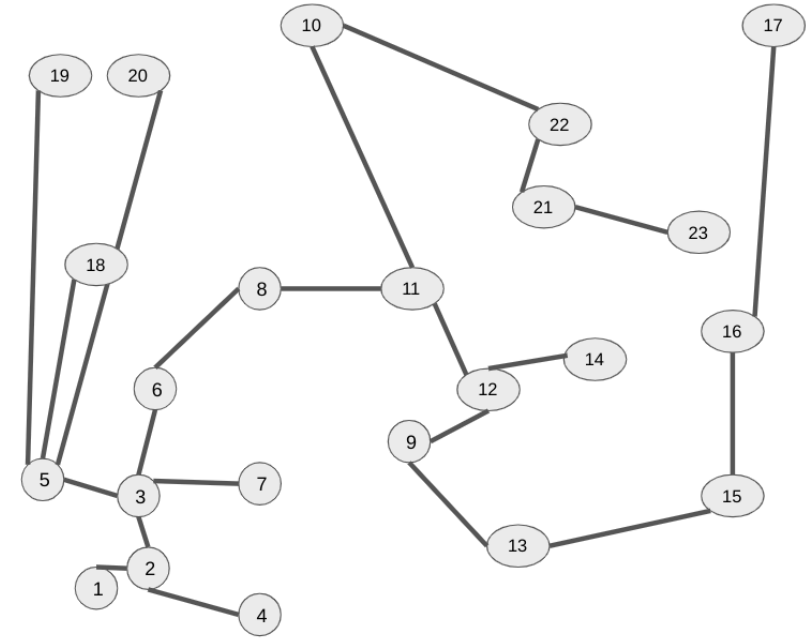
Usando el código para obtener el árbol generador de peso mínimo obtuvimos esto:

```
Se agrego la arista (5, 18, 200)
Se agrego la arista (10, 22, 200)
Se agrego la arista (1, 2, 201)
Se agrego la arista (2, 3, 203)
Se agrego la arista (3, 7, 203)
Se agrego la arista (3, 6, 204)
Se agrego la arista (6, 8, 205)
Se rechazo la arista (7, 8, 206)
Se agrego la arista (9, 12, 209)
Se agrego la arista (11, 12, 210)
Se agrego la arista (8, 11, 211)
Se agrego la arista (9, 13, 212)
Se agrego la arista (12, 14, 215)
Se agrego la arista (3, 5, 218)
Se agrego la arista (15, 16, 223)
Se agrego la arista (21, 22, 225)
Se agrego la arista (5, 19, 400)
Se agrego la arista (5, 20, 400)
Se agrego la arista (16, 17, 400)
Se rechazo la arista (7, 9, 415)
Se rechazo la arista (7, 11, 418)
Se rechazo la arista (8, 9, 419)
Se rechazo la arista (9, 11, 419)
Se rechazo la arista (11, 14, 425)
Se rechazo la arista (13, 14, 427)
Se agrego la arista (13, 15, 435)
Se rechazo la arista (14, 15, 436)
Se agrego la arista (21, 23, 459)
Se agrego la arista (8, 10, 644)
Se rechazo la arista (10, 11, 644)
Se agrego la arista (2, 4, 660)
```



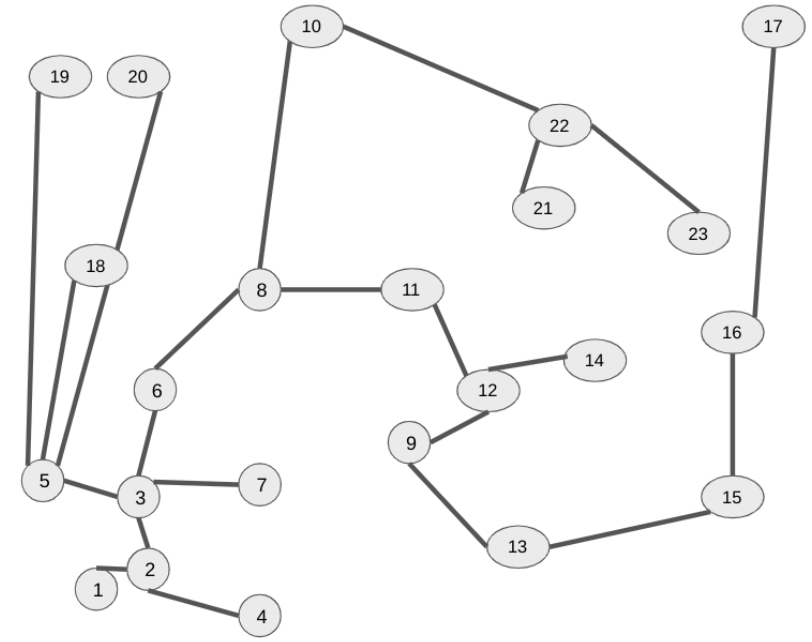
Árbol generador de peso mínimo (pero no es único, ya que se rechaza una arista con peso igual a una de las seleccionadas)

Entonces otro árbol generador de peso mínimo se ve así:

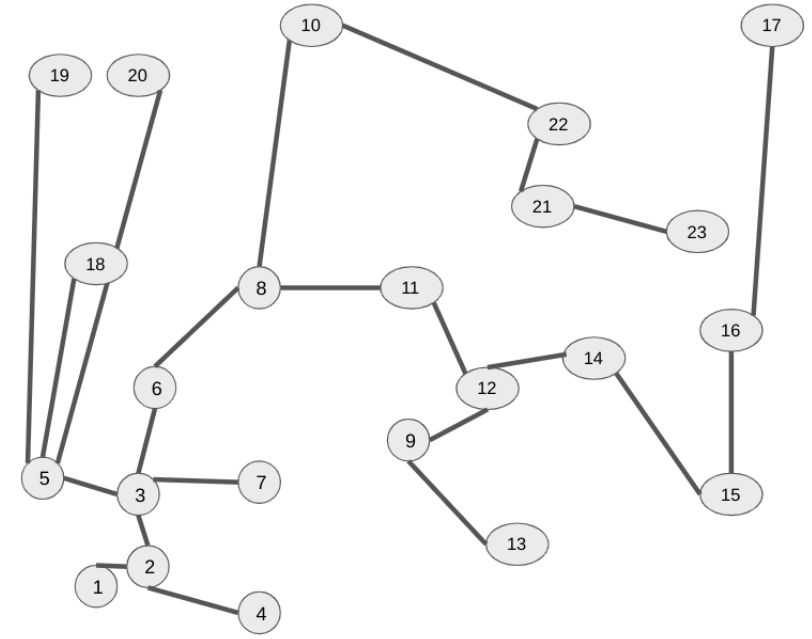


Otro árbol generador de peso mínimo (ya que el peso de las aristas 8-10 y 10-11 es el mismo)

Y los otros dos árboles generador (no son de peso mínimo) son estos:



Usando los árboles generadores de peso mínimo creamos otro árbol generado quitando la arista 21-23 y agregando la arista 22-23



Usando los árboles generadores de peso mínimo creamos otro árbol generado quitando la arista 13-15 y agregando la arista 14-15

Los problemas con la gráfica de la región son los siguientes, suponiendo que los nodos de mayor grado son las ciudades más importantes o con más población entonces tenemos que en la gráfica de la región las ciudades más importantes tienen varias conexiones (aristas) hacia otras aristas, por cual si alguna arista o conexión conecta a estas ciudades falla, van a seguir existiendo caminos de esa ciudad hacia otras por lo tanto no quedaría sin ninguna conexión, pero si alguna ciudad pequeña o no tan importante (con grado 2 o 1) pierde una arista (alguna conexión falla) entonces puede quedarse sin conexión con el resto de las demás ciudades o hasta en casos como en el del nodo 10 donde si la aristas 11-22 falla entonces va a pasar que los nodos 22 a 24 no van a tener contacto con los demás nodos. Una posible solución sería agregar nuevas aristas desde cada nodo a otro nodo más cercano (para que el peso/precio no sea tan alto) e intentar lograr que entre cada par de nodos existe al menos 2 caminos, pero este proceso de agregar muchas aristas puede ser algo costoso, por lo tanto solo se tendría que

realizar para las ciudades que sean lo suficientemente importantes o que el precio lo valga, y por último el peso total de la gráfica de la región en 11029.

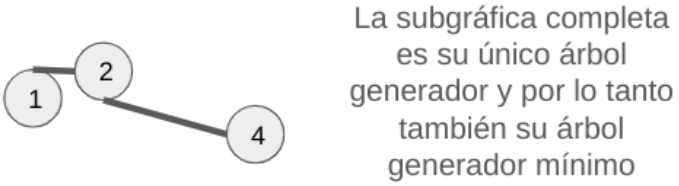
Y para el árbol generador de peso mínimo tenemos que los posibles problemas son que ahora las ciudades más importantes solo tiene grado 2, por lo tanto si alguna arista que conecta a la ciudad falla, vamos a tener que la ciudad puede quedarse sin comunicación con una parte del árbol, lo cual no es bueno para nosotros, de manera similar, los nodos con solo grado 1 tiene mayor riesgo a perder conexión con el resto de los nodos si la arista explota, una posible solución para estos problemas sería similar a la de la gráfica general sería agregar algunas aristas extras en las ciudades más importantes para de esta manera poder evitar que pierdan conexión, solo tomando en cuenta el precio/peso de las nuevas aristas. Y el peso total del árbol es 6537.

La primera sub gráfica es esta:

Usando el código para obtener el árbol generador de peso mínimo obtuvimos esto:

```
Se agrego la arista (1, 2, 201)
Se agrego la arista (2, 3, 660)
```

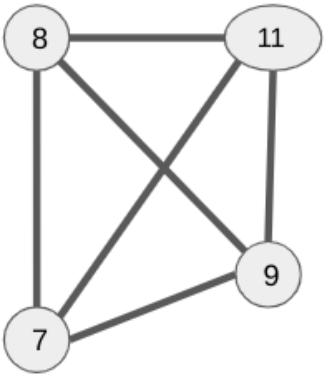
(para las sub gráfica renombramos a los vértices para que el código funciona)



Y, por lo tanto, existe un solo árbol generador.

Los problemas con esta subgráfica y por lo tanto con su árbol generador de peso mínimo es que si la arista 1-2 falla entonces el nodo 1 va a quedar desconectado de los demás, de igual manera si la arista 2-4 falla entonces el nodo 4 va a quedar desconectado, por lo tanto una posible solución sería agregar una arista entre el nodo 1 y el 4, para que de esta manera si una arista falla entonces ningún nodo va a quedar desconectado. Y el peso total de la subgráfica es 861.

La segunda sub gráfica es esta:

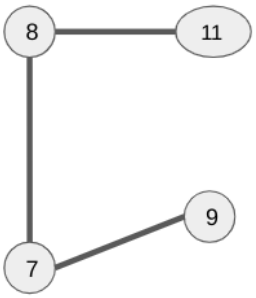


Pesos:
7-8: 206
7-9: 415
7-11: 418
8-9: 419
8-11: 211
9-11: 419

Usando el código para obtener el árbol generador de peso mínimo obtuvimos esto

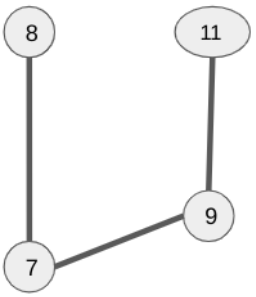
```
Se agrego la arista (1, 2, 206)
Se agrego la arista (2, 4, 211)
Se agrego la arista (1, 3, 415)
```

(para las sub gráfica renombramos a los vértices para que el código funciona)

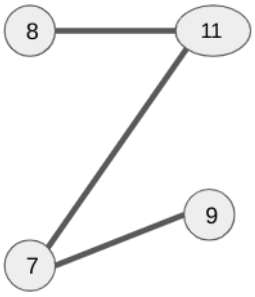


Árbol generador de peso mínimo (es único, ya que no se rechazó alguna con peso igual a las seleccionadas)

Y los otros dos árboles generador (no son de peso mínimo) son estos



Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 8-11 y agregando la arista 9-11

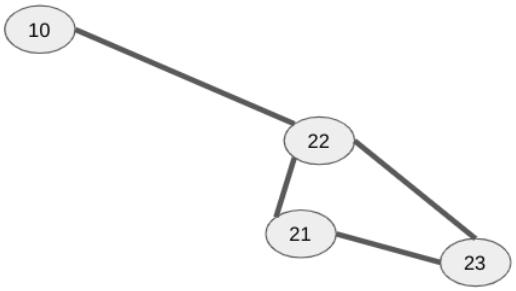


Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 7-8 y agregando la arista 7-11

En esta subgráfica no encontramos muchos problemas, debido a que es una gráfica donde cada nodo se conecta con cada otro nodo, por lo tanto si una arista falla no vamos a tener el problema de que un nodo quede desconectado. Y el peso total de la subgráfica es 2078.

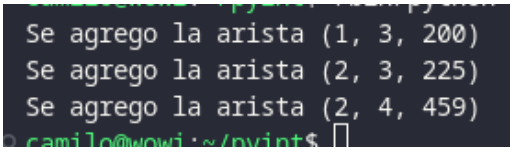
Y para el árbol generador de peso mínimo los problemas son que si alguna arista falla entonces vamos a dejar a un nodo desconectado de los demás (en el caso de la aristas 7-8, vamos a tener que 8 y 11 están conectados entre sí y 7 y 9 también pero no entre estos pares), por lo tanto una posible solución sería agregar la arista 9-11, para así formar un ciclo y evitar que los nodos se desconecten si una arista falla . Y el peso total del árbol es 832.

La tercera sub gráfica es esta:

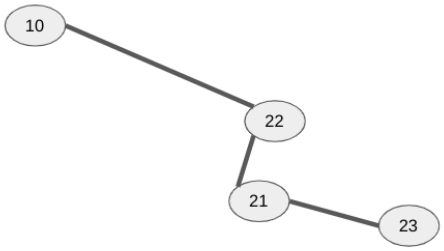


Pesos:
10-22: 200
21-22: 225
21-23: 459
22-23: 683

Usando el código para obtener el árbol generador de peso mínimo obtuvimos esto:

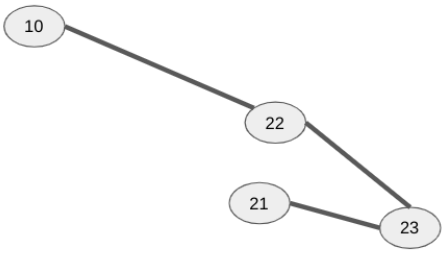


(para las subgráfica renombramos a las vértices para que el código funciona)

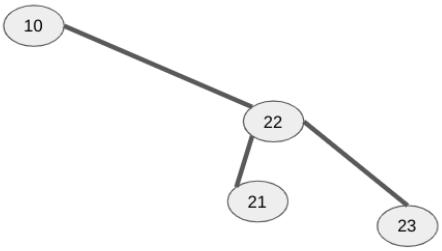


Árbol generador de peso mínimo (es único, ya que no se rechazó alguna con peso igual a las seleccionadas)

Y los otros dos árboles generador (no son de peso mínimo) son estos:



Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 21-22 y agregando la arista 22-23

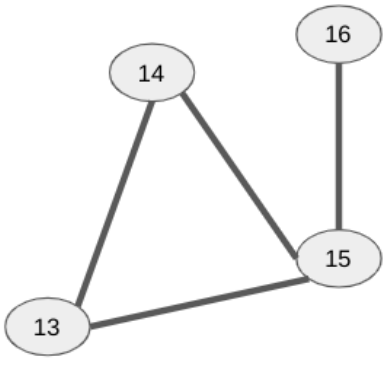


Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 21-23 y agregando la arista 22-23

Los problemas con esta subgráfica son que si la arista 10-22 falla el nodo 10 va a quedar desconectado de otros 3 nodos, en cambio si alguna otra arista falla no se quedarán desconectados, entonces la solucion podria ser agregar una nueva arista del nodo 10 al nodo 21 o 23, dependiendo de cual genera menos peso. Y el peso total de la subgráfica es 1567.

Y para el árbol generador de peso mínimo los problemas son que si alguna arista falla entonces vamos a dejar a un nodo desconectado de los demás (en el caso de la aristas 21-22, vamos a tener que 21 y 23 están conectados entre sí y 10 y 22 también pero no entre estos pares), por lo tanto una posible solución sería agregar la arista 10-23, para así formar un ciclo y evitar que los nodos se desconecten si una arista falla . Y el peso total del árbol es 884.

La cuarta subgráfica es esta

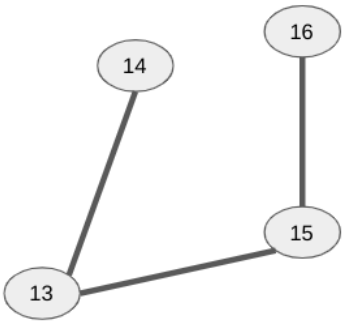


Pesos:
13-14: 427
13-15: 435
14-15: 436
15-16: 223

Usando el código para obtener el árbol generador de peso mínimo obtuvimos esto:

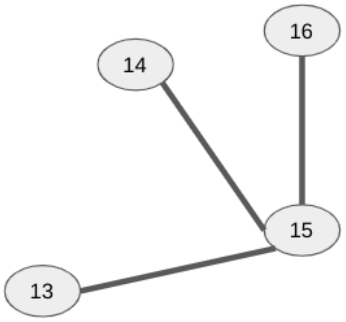
```
Se agrego la arista (3, 4, 223)
Se agrego la arista (1, 2, 427)
Se agrego la arista (1, 3, 435)
```

(para las subgráfica renombramos a las vértices para que el código funciona)

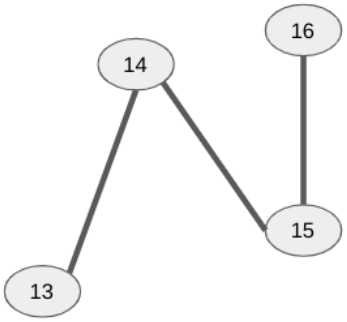


Árbol generador de peso mínimo (es único, ya que no se rechazó alguna con peso igual a las seleccionadas)

Y los otros dos árboles generador (no son de peso mínimo) son estos:



Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 13-14 y agregando la arista 14-15



Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 13-15 y agregando la arista 14-15

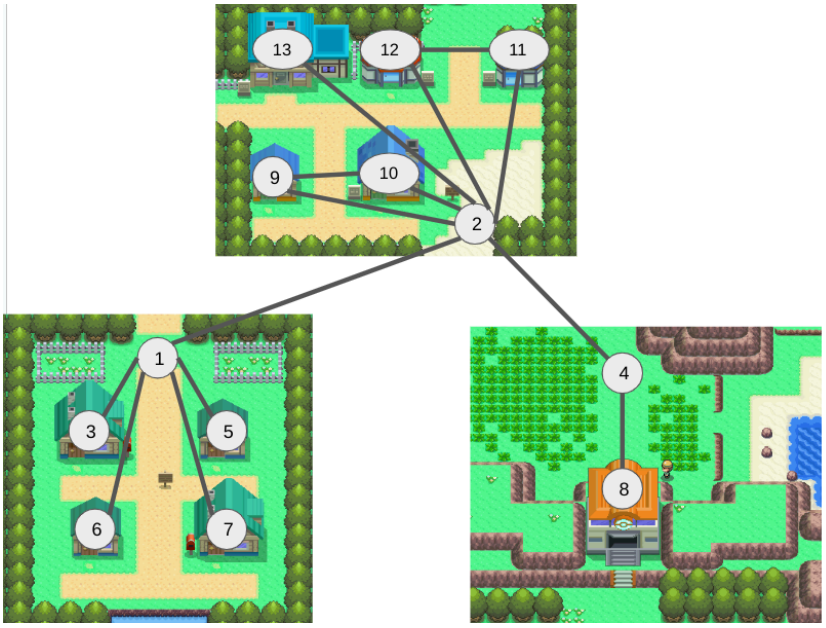
Los problemas con esta subgráfica son que si la arista 15-16 falla el nodo 16 va a quedar desconectado de otros 3 nodos, en cambio si alguna otra arista falla no se quedarán desconectados, entonces la solucion podria ser agregar una nueva arista del nodo 16 al nodo 14, ya que es el más cercano, aunque si la arista al del nodo 16 al nodo 13 tiene peso menor sería mejor agregarla. Y el peso total de la subgráfica es 1521.

Y para el árbol generador de peso mínimo los problemas son que si alguna arista falla entonces vamos a dejar a un nodo desconectado de los demás (en el caso de la aristas 13-15,

vamos a tener que 15 y 16 están conectados entre sí y 13 y 14 también pero no entre estos pares), por lo tanto una posible solución sería agregar la arista 14-16, para así formar un ciclo y evitar que los nodos se desconecten si una arista falla . Y el peso total del árbol es 1085.

7) **De lo anterior selecciona 1 sub gráfica y mapea toda la zona como otra gráfica, es decir, si en un nodo hay 4 casas, supón que existe un nodo que conecta las 4 casas a este (para este si se ve complicado puedes usar la imaginación y suponer que existe una caja que alimenta de Internet las casas), aplica los algoritmos anteriores ya con esta nueva gráfica.**

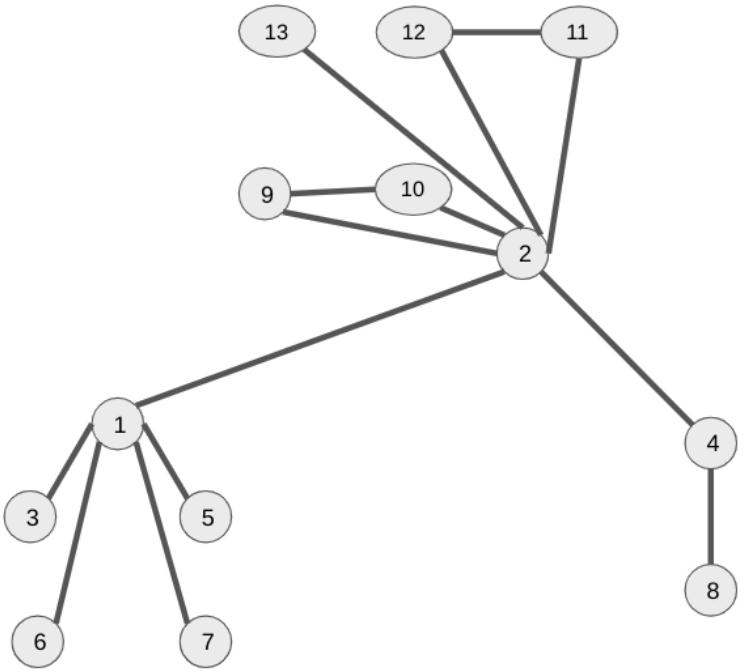
Usamos la subgráfica 1, que contenía a los nodos 1, 2 y 4, los cuales representan al Pueblo Hojaverde, Pueblo Arena y Parque Compi respectivamente, entonces usamos sus mapas para agregar nuevos nodos a la sub gráfica 1, quedando así:



Los pesos de las nuevas aristas serán serán valores inventados y los de la gráfica de la región se mantendrán

Se agregaron aristas entre los vértices 11 y 12, y entre 9 y 10, para que fuera una gráfica más interesante

Y con pesos de esta manera:

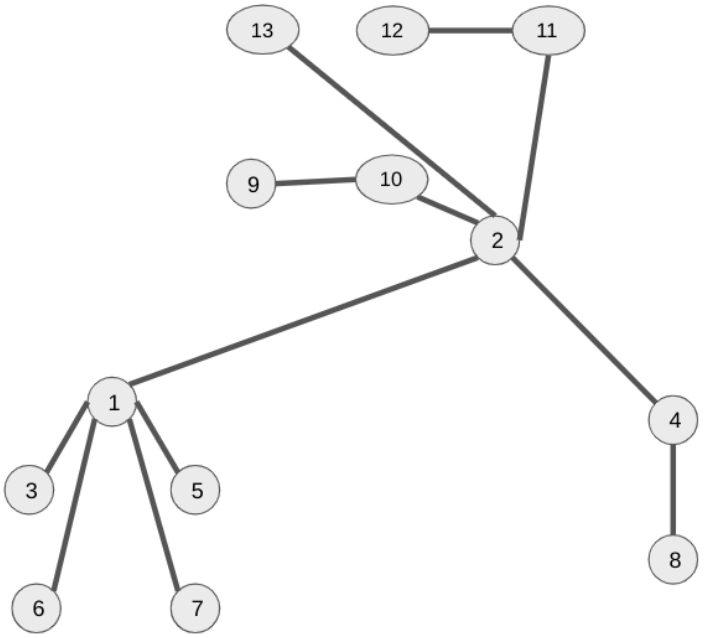


Pesos:
1-2: 201
2-4: 660
1-3: 30
1-5: 29
1-6: 45
1-7: 48
4-8: 60
2-9: 25
2-10: 10
2-11: 40
2-12: 42
2-13: 46
9-10: 10
11-12: 12

Usando el código para obtener el árbol generador de peso mínimo obtuvimos esto:

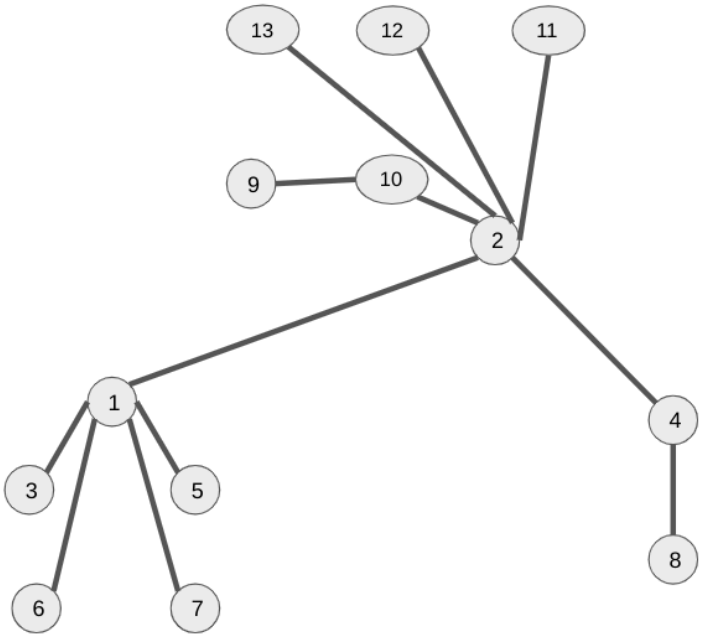
```
Se agrego la arista (3, 4, 223)
Se agrego la arista (1, 2, 427)
Se agrego la arista (1, 3, 435)
Se agrego la arista (2, 10, 10)
Se agrego la arista (9, 10, 10)
Se agrego la arista (11, 12, 12)
Se rechazo la arista (2, 9, 25)
Se agrego la arista (1, 5, 29)
Se agrego la arista (1, 3, 30)
Se agrego la arista (2, 11, 40)
Se rechazo la arista (2, 12, 42)
Se agrego la arista (1, 6, 45)
Se agrego la arista (2, 13, 46)
Se agrego la arista (1, 7, 48)
Se agrego la arista (4, 8, 60)
Se agrego la arista (1, 2, 201)
Se agrego la arista (2, 4, 660)
```

(para las subgráfica renombramos a los vértices para que el código funciona)

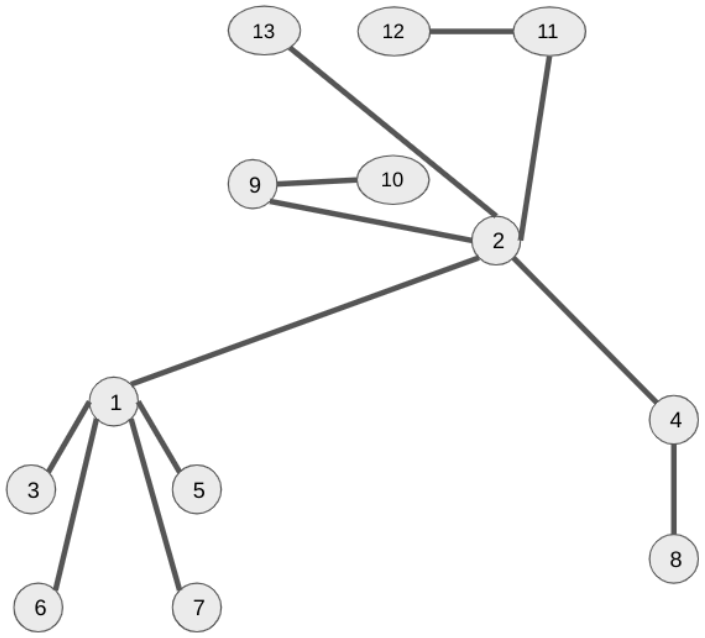


Árbol generador de peso mínimo (es único, ya que no se rechazó alguna con peso igual a las seleccionadas)

Y los otros dos árboles generador (no son de peso mínimo) son estos:



Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 11-12 y agregando la arista 2-12



Usando el árbol generador de peso mínimo creamos otro árbol generado quitando la arista 2-10 y agregando la arista 2-9

Los problemas con esta nueva gráfica son que si alguna de las aristas 1-2, o 2-4 fallas entonces vamos a tener que las casitas de los pueblos van a quedarse desconectados de los demás pueblos, pero entre las casitas del pueblo van a seguir conectados. Una posible solución sería la misma para la subgráfica 1 (la que se usó para generar esta gráfica) sería agregar una nueva arista entre el nodo 1 y 4 para que así sea más complicado que un pueblito se quede sin contacto con los demás. Otra posible solución sería agregar algunas nuevas aristas entre las casitas del nodo 1, para que así si falla una arista la casita de esa arista no quede del todo desconectada. Y el peso total de la subgráfica es 1258.

Y para el árbol generador de peso mínimo los problemas son muy similares a los de la gráfica, ya que el árbol solo tiene dos aristas menos que la gráfica, por lo tanto los mismos problemas de la gráfica están presentes, que sería posibles desconexiones de pueblitos si las aristas 1-2 o 2-4 fallan o hasta dejar casitas de ciertos pueblos sin coneccion, entonces las mismas soluciones se mantienen, que son agregar una aristas entre los nodos 1 y 4, para evitar desconectar pueblitos y tal vez agregar algunas aristas entre las casitas, pero esta última depende mucho de que tan caro/pesadas sean las aristas y si hay mucha demanda. Y el peso total del árbol es 1191.

8) Supón que eres un paquete de datos viajando a través de la red, ¿de qué manera o qué algoritmo aplicarías para pasar por cada nodo? ¿Tu solución es óptima?

El algoritmo que emplearía para pasar por cada nodo es Breadth-First Search. Lo realizaría de la siguiente manera:

1. Empiezo en un nodo arbitrario y lo marco como visitado.
2. Agrego al nodo inicial a una cola.
3. Mientras la cola no esté vacía:
    a. Saco el primer elemento de la cola.

b. Agrego a los vecinos a la cola y los marco como visitados.

La solución es óptima, ya que visita todos los nodos y únicamente los visita una vez.

9) Del anterior, supón que el paquete de datos es un virus que con acceder a cualquier nodo lo infecta, salvo los que tengan un antivirus, propón una solución en la cual se pueda infectar el mayor número de computadoras antes de perecer, el punto de salida es arbitrario.

Similarmente, ocuparemos BFS. El virus comenzará en un nodo que no tenga antivirus. Posteriormente, infectará el nodo y lo añadirá a una cola.

Después, por cada vecino del nodo actual:

1. Si el vecino no tiene antivirus y aún no ha sido infectado, se infecta y se añade a la cola.
2. En otro caso, el virus no se propaga al vecino.

El proceso continúa hasta que todos los nodos se han infectado o el antivirus no sea capaz de infectar más nodos.

10) Ahora supongo que las aristas (cables de red) tienen cierto peso, tu virus informático tiene x cantidad de peso que puede recorrer, ¿De qué manera podrías infectar el mayor número de equipos? Propón una solución.

Podemos proponer un algoritmo con un enfoque de programación dinámica. Donde generemos un diccionario de estados, donde cada uno de ellos guarda la mejor secuencia y la mayor cantidad de nodos visitados dado un nodo y una capacidad restante.

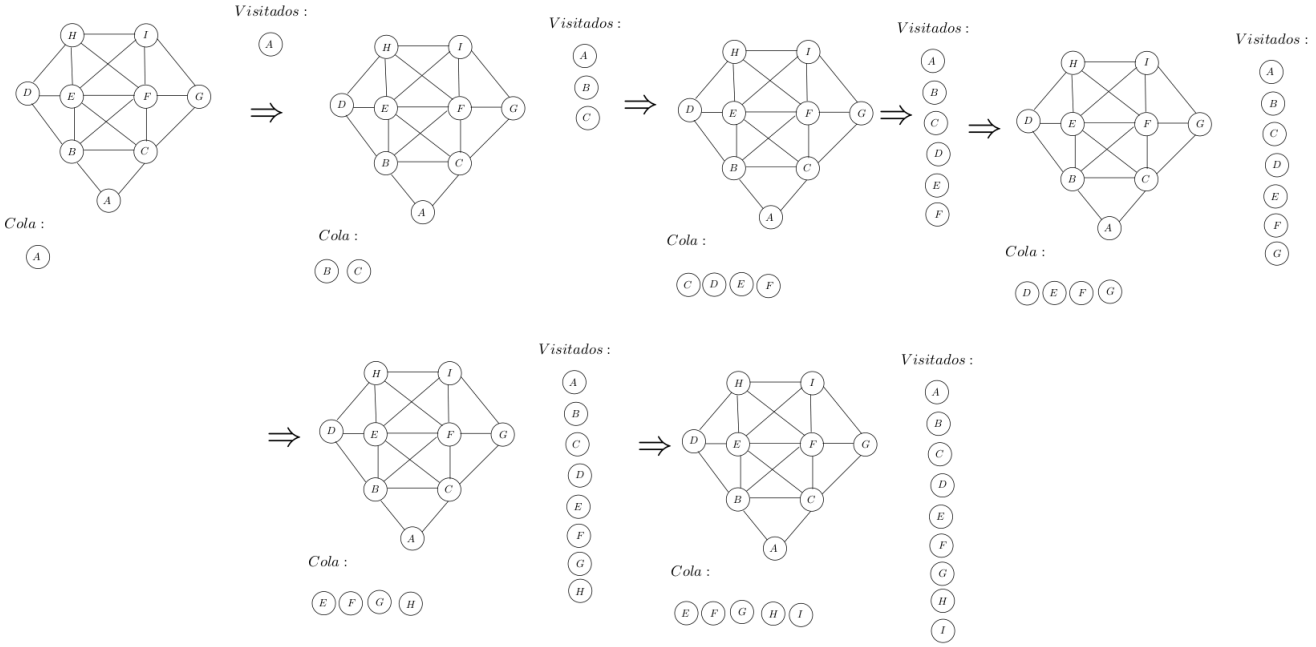
Empezamos con un nodo cualquiera y con la capacidad de recorrido actual:

- Si el nodo y dicha cantidad ya está dentro de nuestro diccionario, devolvemos su respectivo valor.
- Agregamos el nodo actual a un conjunto de nodos visitados e inicializamos un contador máximo y una lista con la mejor secuencia actuales.
- Por cada vecino llamamos a nuestro algoritmo con el vecino a calcular la capacidad actual (restandole el peso de la arista con la cual se une al vecino), guardando el contador de nodos y secuencia máxima que este estado tenga.
- Posteriormente, si los resultados de dicha llamada son mayores al contador máximo y al tamaño de nuestra solución actual, los reemplazamos (incluyendo nuestro nodo en la secuencia obtenida).
- Restamos el nodo actual de nuestro conjunto de nodos visitados, guardamos el estado del nodo actual en el diccionario y devolvemos nuestra cantidad de nodos recorridos y una lista con los nodos recorridos.

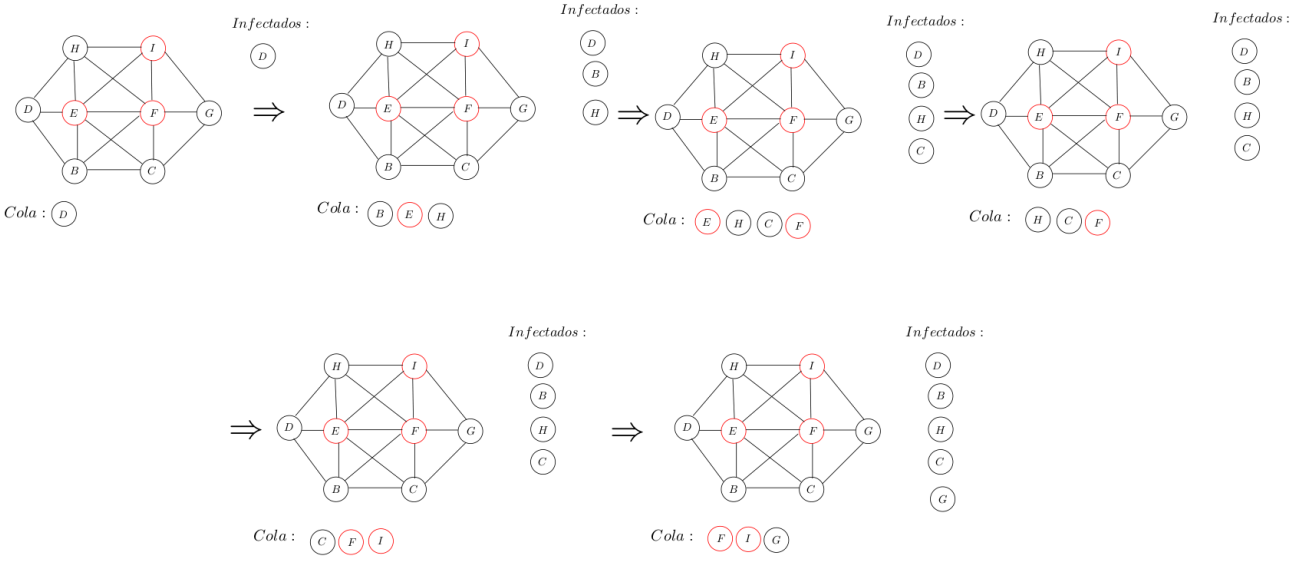
Como se puede observar, este algoritmo tiene un caso recursivo al calcular la mejor secuencia y mayor cantidad de nodos de los vecinos, el caso base se llegará cuando ya no haya vecinos o el peso de las aristas sean mayores a la capacidad restante. De esta manera veremos cuál de todas las posibilidades es aquella que recorre la mayor cantidad de nodos.

11) Para los 3 ejercicios anteriores da unos 3 ejemplos con redes de distintos tamaños. (Más de 6 nodos).

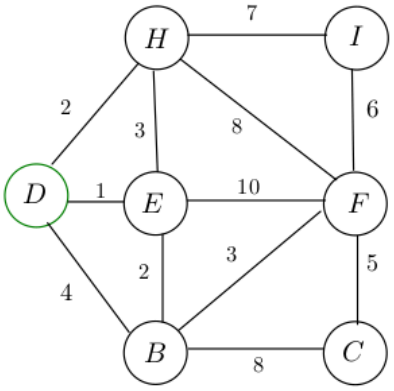
1) Ejercicio 8



2) Ejercicio 9



3) Ejercicio 10



$x = 12$

$(H, 10) = (4, \{D, H, E, B, F\})$

$(E, 11) = (5, \{D, E, B, F, C\})$

$(B, 8) = (4, \{D, B, F, C\})$

Tras llamar a (D,12), verificamos cuales de sus vecinos nos ofrecen una solución mejor, para reducir la gran cantidad de pasos, podemos ver que E nos devuelve un total de 5 nodos antes de terminar con la capacidad de "x".

12) Finalmente, relaciona los ejercicios 2, 6, 7, 8, 9 y 10 y pon las conclusiones que encuentres.

Dependiendo de la topología usada para la construcción de una red, podemos facilitar la comunicación y la seguridad entre varios dispositivos de manera más eficiente. Hay tipologías que permiten sacar mayor provecho a los algoritmos que queramos emplear para la navegación de un paquete, no obstante esto depende también de los usos que se tienen pensados para dichas redes. De igual manera, el uso de los datos que queramos enviar es importante, pues en caso de haber un virus es conveniente evitar la mayor cantidad de dispositivos afectados.

13) *EXTRA*:) Investiga si hay algún mapa de RED de alguna ciudad (Si es de CDMX mejor), de aquí observa cómo se está conectado y trata de identificar alguna topología, en caso de que sea muy grande, todavía, selecciona una sección arbitraria (También sirven los de cobertura).

Referencias:

- **Cisco Community.** *Enrutamiento: Conceptos Fundamentales*. Consultado de: https://community.cisco.com/legacyfs/online/attachments/document/enrutamiento-conceptos_basicos.pdf
- **Microsoft.** *Dynamic Host Configuration Protocol (DHCP)*. Consultado de: https://learn.microsoft.com/en-us/windows-server/networking/technologies/dhcp/dhcp-top
- **Network Academy.** *IPv6 Stateless Address Auto-configuration (SLAAC)*. Consultado de:https://www.networkacademy.io/ccna/ipv6/stateless-address-autoconfiguration-slaac#:~:text=What%20is%20SLAAC%3F,is%20assigned%20to%20which%20node.

- **IBM.** *Direccionamiento estático y dinámico*. Consultado de: https://www.ibm.com/docs/es/aix/7.2?topic=routing-static-dynamic