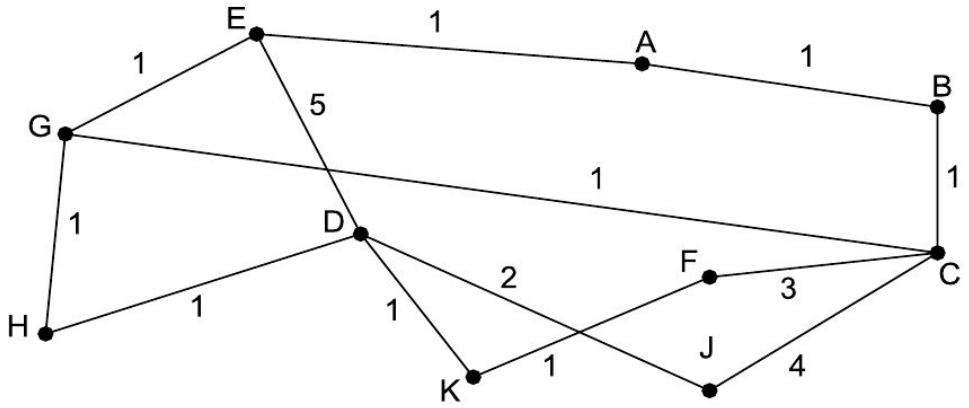
C5-P1. Ruteo

- **Aplique algún algoritmo de optimización (Dijkstra o Bellman-Ford) para encontrar la ruta más corta entre el nodo A y K en el siguiente grafo.**
  - Explique paso a paso su desarrollo, con el mayor detalle posible. Use tablas, diagramas o pseudo-código para que su explicación sea muy clara y completa.
  - No necesita programar nada, sólo explicar su algoritmo paso a paso.



DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Nombre: Alberto Rodríguez Z.

RUT: 19.950.454-1

C5-P1

Se usará el algoritmo de Bellman-Ford para encontrar la solución, explicaremos cada paso con el siguiente pseudo-código:

Algoritmo:

```
bool BellmanFord(Grafo G, nodo_origen A):
  //inicializamos el grafo,  ponemos distancias  a Infinito  menos el nodo origen que tiene distancia 0
  Para vertice ∈ V[G]:
      distancia[v] = INFINITO
  distancia[A]=0  //Al ser el nodo A el nodo de origen se inicializa en 0
  //Se relaja cada arista del grafo tantas veces como número de nodos -1 haya en el grafo
  Para h=1 en  | V[G] | -1: //h indica la cantidad de enlaces que deben haber
      Para (u, v) ∈ G:
          Si distancia[v] > distancia[u] + peso(u, v):
              distancia[v] = distancia[u] + peso(u, v)
              predecesor[v] = u
   //Se debe comprobar si hay ciclos en negativos
   Para(u, v) e G:
       Si distancia[v] > distancia[u] + peso(u, v):
           retornar FALSO
   retornar VERDADERO
```

Una de las condiciones de este algoritmo es que en cada paso que se hará, es que haya un número máximo de enlaces la cual denotaremos como h.

C5-P1

TABLA: En la siguiente diaposisitva

Nota: En la siguiente diapositiva se explica la tabla

Nodos	Paso 1 (h=0)	Paso 2 (h=1)	Paso 3 (h=2)	Paso 4 (h=3)	Paso 5 (h=4)	Paso 6 (h=5)	Paso 7 (h=6)
A	(A,0)	(A,0)	(A,0)	(A,0)	(A,0)	(A,0)	(A,0)
B	(A,∞)	(A,1)	(A,1)	(A,1)	(A,1)	(A,1)	(A,1)
C	(A,∞)	(A,∞)	(A-B,2)	(A-B,2)	(A-B,2)	(A-B,2)	(A-B,2)
D	(A,∞)	(A,∞)	(A-E,6)	(A-E,6)	(A-E-G-H,4)	(A-E-G-H,4)	(A-E-G-H,4)
E	(A,∞)	(A,1)	(A,1)	(A,1)	(A,1)	(A,1)	(A,1)
F	(A,∞)	(A,∞)	(A,∞)	(A-B-C,5)	(A-B-C,5)	(A-B-C,5)	(A-B-C,5)
G	(A,∞)	(A,∞)	(A-E,2)	(A-E,2)	(A-E,2)	(A-E,2)	(A-E,2)
H	(A,∞)	(A,∞)	(A,∞)	(A-E-G,3)	(A-E-G,3)	(A-E-G,3)	(A-E-G,3)
J	(A,∞)	(A,∞)	(A,∞)	(A-B-C,6)	(A-B-C,6)	(A-B-C,6)	(A-B-C,6)
K	(A,∞)	(A,∞)	(A,∞)	(A-E-D,7)	(A-B-C-F,6)	(A-E-G-H-D,5)	(A-E-G-H-D,5)

C5-P1 Explicación y resultados de la tabla

En la tabla se muestran la rutas que va calculando el algoritmo. Las filas indican los nodos del grafo y las columnas los pasos (donde en cada una va aumentando el h, que es la cantidad permitida en enlaces). En cada recuadro se pueden ver paréntesis, lo cuales indican el costo de llegar al nodo (de la fila) y cuál es la ruta para llegar a dicho nodo. Todo esto considerando que todos parten del nodo A, un ejemplo, si estamos en la fila del nodo D y en la columna del Paso 3, la ruta es (A-E, 6), es decir el costo para llegar al nodo D desde A es de 6 y la ruta para llegar a D es A-E.

Ya explicado los contenidos de la tabla, se explica cómo se obtuvo la ruta más corta entre A y K usando este algoritmo. El Paso 1 (h=0), es la etapa de inicialización, es decir se ponen todos los costos a los nodos desde a como infinito y al nodo A con costo 0. Ya en el Paso 2 cómo puede existir solo un enlace, solo se puede visitar los nodos adyacentes a A, es decir, los nodos B y E y se actualiza en la tabla, todos los demás siguen teniendo distancia infinita. En el Paso 3 cómo se permiten dos enlaces, otro nodos son actualizados en la tabla. En cada iteración se calcula la distancia mínima a cada nodo cumpliendo la restricción de enlaces.

El algoritmo tiene un límite, que es, que continuará iterando hasta la cantidad de nodos -1, osea, que para este caso como son 10 nodos la iteración sería hasta h=9, pero esta puede terminar antes. Como se aprecia en la tabla sólo llegó hasta h=6, esto ocurre cuando el algoritmo ve que no hay ningún cambio en los datos de la columna con su columna anterior, cuando esto ocurre el algoritmo se detiene.

Para obtener la ruta más corta entre A y K, basta con ver la última columna de la tabla en la fila K, la cual indica que la ruta de menor costo es A-E-G-H-D-K.

C5-P2. Capa de transporte

- **Elija SOLO UNA de las siguientes opciones**
  a. Utilizando la librería sockets de Python3 implemente un cliente y servidor UDP que permita transferir un archivo desde el servidor al cliente.
     i. Transfiera un archivo de al menos 200MB y calcule la tasa de datos alcanzada
     ii. Explique en qué parte del código se observa que UDP es un protocolo no orientado a conexión
     iii. Explique en qué parte de su código se observa que UDP es un protocolo orientado a datagramas (mensajes delimitados de largo fijo)
  b. Utilizando la librería sockets de Python3 implemente un cliente y servidor TCP que permita transferir un archivo desde el servidor al cliente.
     i. Transfiera un archivo de al menos 200MB y calcule la tasa de datos alcanzada
     ii. Explique en qué parte de su código se observa que TCP es un protocolo orientado a conexión
     iii. Explique en qué parte de su código se observa que TCP es un protocolo orientado a flujo de bytes (sin delimitación)

NOTA: Sólo entregue un documento PDF, explique su código y los resultados en el mismo documento.
NOTA: No necesita entregar el código por separado.

Nombre:	RUT:

C5-P2 Cliente-UDP

Se va a usar el protocolo UDP para desarrollar esta pregunta.

```
1  import socket
2  import sys
3
4  #Creando el socket del cliente
5  client_sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
6
7  #Asignamos direcion y puerto
8  server_address = ('localhost', 10000)
9
10 #Lectura del archivo en modo binario
11 f=open("200MB.zip","rb")
12 #De estos, primero se enviarán 65.000 bytes
13 data=f.read(65500)
14
15 while(data): #Este iterará hasta que se envie todo el archivo
16     #El socket envia los primero dados al servidor
17     if(client_sock.sendto(data, server_address)):
18         #Se vuelve a leer 65.000 bytes del archivo
19         data= f.read(65500)
20
21 f.close()
22 print("Enviado el archivo!")
23
24 #Se cierra la conexión
25 client_sock.close()
```

El primer código que se analiza es el del cliente. En la línea 5 se crea el socket con los argumentos de AF_INTET, que indica que pertenece a la familia del protocolo IPv4 y el segundo argumento SOCK_DGRAM indica que se va usar el protocolo UDP.

Después se indica cual es el ip y el puerto del servidor a cual se le enviará los datos.

Se lee de un archivo que tiene una tamaño de 200MB. Como UDP es un protocolo orientado a datagrama se debe indicar el tamaño fijo, este se muestra en la línea 13 del código, donde se indica un tamaño fijo, que es un pedazo del tamaño total, en este caso 65.500 bytes. Después dentro del while se van enviando varios pedazos de 65.500 bytes hasta que se envíe todo el archivo.

Ya cuando finalice por consola se indicará que se a enviado todo el archivo. Haciendo esto en la línea 25 se indica que se cierra la conexión del socket.

C5-P2 Servidor-UDP

```python
import socket
import sys

#Creando el socket del servidor
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

#Se enlaza el servidor a una ip con un puerto
server_address = ('localhost', 10000)
sock.bind(server_address)

#Se crea un ciclo el cual estará esperando que le lleguen mensajes
while True:
    print('\nEsperando recibir un mensaje')
    #Cuando llega, se guarda su contenido y la dirección del emisor
    data, address = sock.recvfrom(65500)
    print('recibidos {} bytes desde {}'.format(len(data), address))

```

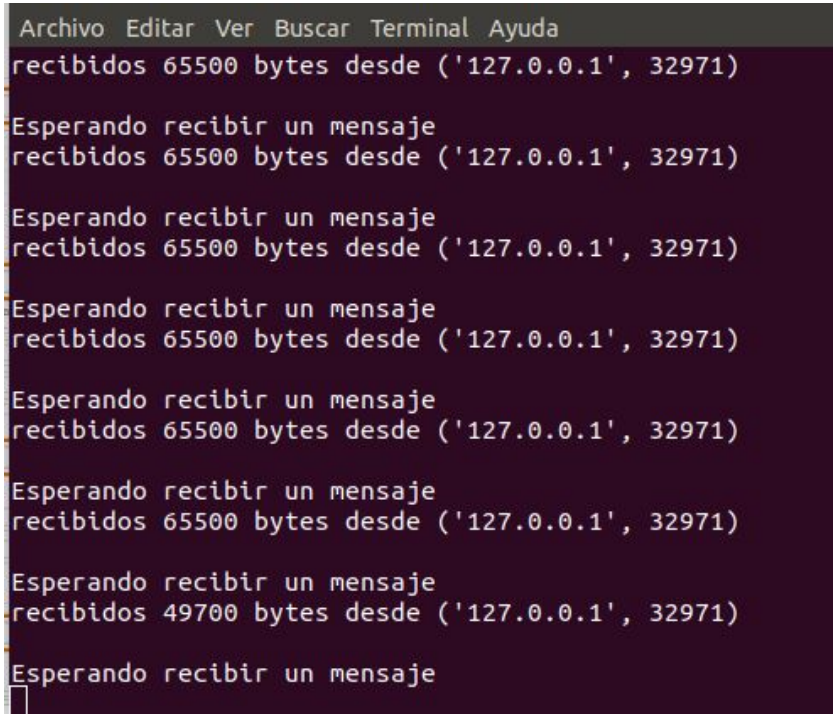
El segundo código que se analiza es el del servidor. En la línea 5 se crea el socket de la misma forma que el cliente.

Después se enlaza el socket a una dirección ip junto con un puerto (esto mediante la función bind( ))

En la última parte del código se crea un ciclo el cual espera recibir datagramas de cualquier cliente que se quiera comunicar con el. Al final solo se muestra por consola cuantos bytes le llegaron y desde que origen.

Como se usa un protocolo no orientado a conexión se puede observar en el código que el servidor no necesita aceptar la conexión con ningún cliente, solo espera que los datagramas lleguen. Este datagrama contiene la dirección del emisor con los datos. Si este quisiera responderle, ya sabe la dirección del emisor.

C5-P2 Ejecutando el programa



La terminal muestra la última parte de la ejecución del código del servidor, el cual va indicando que le van llegando 65.500 bytes desde la dirección del cliente. En el último le llegan 49.700 bytes que serian últimos bytes que faltan enviar del archivo.

Se probó con distintos tamaños para enviar el archivo. La cantidad máxima de bytes que se pueden enviar por pedazos son 65.500. Cuando se ponía un número mayor a este, por la terminal del cliente mostraba un error que no se pueden enviar esa cantidad de bytes por la función sendto( ).

Por lo tanto la tasa de datos alcanzada son 65.500 bytes/ en una unidad muy pequeña de segundos. Al ver que se puede transmitir muchos bytes se puede decir que es bastante rápido. Esto puede depender de la velocidad del internet, el SO o los dispositivos que se comunican (computador, celular, etc).