Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos 1'2024

Interrogación 3

9 de julio de 2024

Condiciones de entrega: Debe entregar solo 3 de las siguientes 4 preguntas.

Tiempo: 2 horas 30 minutos

Entrega: Al final de la prueba tienen 10 minutos para subir la imagen de la prueba a **https://iic2133.org**, cada pregunta por separado y en vertical

Evaluación: Cada pregunta tiene 6 puntos (+1 punto base). La nota es el promedio de las 3 preguntas entregadas.

1. Rutas en Grafos

Un agujero de gusano es un túnel que atraviesa el espacio y el tiempo y conecta dos sistemas estelares. Los agujeros de gusano tienen algunas propiedades peculiares:

- Cada agujero de gusano es de un único sentido y tiene dos puntos extremos, uno llamado extremo *origen* (que es donde comienza el agujero de gusano) y el otro llamado extremo *destino* (que es donde termina el agujero de gusano). Ambos extremos están situados en sistemas estelares distintos.
- El tiempo que se tarda en atravesar un agujero de gusano es insignificante.
- Un sistema estelar puede ser el punto extremo (origen o destino) de más de un agujero de gusano.
- Por alguna razón desconocida, a partir de nuestro sistema solar, siempre es posible terminar en cualquier sistema estelar siguiendo una secuencia de agujeros de gusano (quizás la Tierra sea el centro del universo).
- Entre cualquier par de sistemas estelares, hay como máximo un agujero de gusano en cualquier dirección.

Todos los agujeros de gusano tienen una diferencia de tiempo constante entre sus puntos extremos, medido en años usando un valor entero. Por ejemplo, un agujero de gusano específico puede hacer que la persona que lo atraviese viaje X años en el futuro. Otro agujero de gusano puede hacer que la persona viaje Y años en el pasado. Dado que se asume que el universo se originó con el Big Bang, diremos que ése es el tiempo 0. Ningún agujero de gusano puede hacernos retroceder a un instante anterior al Big Bang: si lo intentamos, quedaremos en el tiempo 0. Alice, una brillante doctora en física, quiere utilizar agujeros de gusano para viajar en el tiempo y estudiar diversos fenómenos.

(a) (1 punto) Modele esta realidad usando grafos, indicando todas las características que considere necesarias, de manera de poder resolver el problema planteado en el item (b).
Solución: Deben modelar la situación usando un grafo dirigido. Los nodos son los sistemas estelares, las aristas dirigidas son los agujeros de gusano. Las aristas deben tener peso representado por un número entero, indicando la diferencia en años que se produce al seguir un agujero de gusano. El valor del entero será positivo si el agujero de gusano permite viajar en el futuro, o negativo si se viaja al pasado.

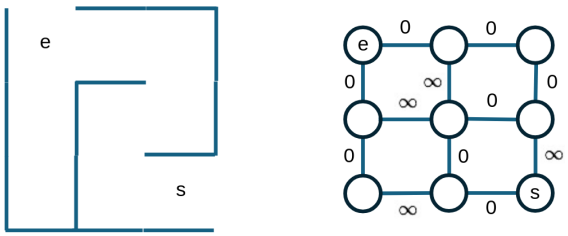
(b) (3 puntos) Escriba un algoritmo eficiente que, a partir de la representación de un conjunto de sistemas estelares y sus correspondientes agujeros de gusano, le permita a Alice determinar si es posible (o no) viajar a ver el Big Bang. Además, en caso afirmativo, su algoritmo debe indicar al menos una secuencia de agujeros de gusano tal que si se los atravieza, permitan ese viaje.
Solución: La idea es determinar si el grafo tiene un ciclo negativo o no, usando el algoritmo de Bellman-Ford, modificado como a continuación para poder obtener el ciclo negativo:

```
    BigBang(s):
 1    for u ∈ V :
 2        d[u] ← ∞;   π[u] ← ∅
 3    d[s] ← 0
 4    for k = 1 . . . |V| − 1 :
 5        for (u, v) ∈ E :
 6            if d[v] > d[u] + c(u, v) then
 7                d[v] ← d[u] + c(u, v)
 8                π[v] ← u
 9    if d[v] > d[u] + c(u, v) then
10        v′ ← u
11        L ← ∅
          while v′ ≠ v :
12            L.append(v′)
13            v′ ← π[v′]
14        return (true, L)
15    return (false, ∅)
```

(c) (2 puntos) ¿Cómo modificaría lo hecho en los items anteriores para obtener un algoritmo que permita determinar si es posible o no avanzar de forma indefinida en el futuro? Indique los detalles (aunque no es necesario escribir pseudocódigo).
Solución: Bellman-Ford no sirve en este caso, porque lo que hay que hacer es detectar un ciclo positivo. Sin embargo, se pueden modificar los pesos de las aristas, multiplicando los valores originales por -1, y así aplicar el algoritmo del item (b).

2. MST y Prim

Un laberinto se puede representar como un grafo conexo en que los nodos representan las celdas y las aristas conectan a las celdas con sus vecinas, dos celdas conectadas en el laberinto son unidas por una arista de costo 0 y un muro entre dos celdas es representado por una arista de costo ∞. Por ejemplo:



(a) (2 puntos) En este escenario, ¿es posible utilizar el algoritmo de Prim para determinar el camino más corto desde un nodo de partida e hasta el nodo de salida s del laberinto? Argumente su respuesta.
Solución: Si, es posible. Dado que Prim permite determinar el MST de un grafo conexo, en particular este MST conecta a costo mínimo a e y s, utilizando las aristas de costo cero.

(b) (4 puntos) Algunos juegos requieren crear laberintos de manera dinámica. Modifique el algoritmo de Prim (entregue el pseudocódigo) para crear un laberinto sin ciclos a partir de un grafo G con un nodo de partida e y un nodo de salida s. **Hint:** Puede suponer que inicialmente todas las celdas del grafo G están rodeadas por muros.
Solución: La solución asume que el grafo tiene inicialmente todas las aristas de costo ∞.

```
    Prim(s):
1     Q ← cola de prioridades con s
2     T ← lista vacía
3     x.key ← 0;  x.parent ← ∅
4     while Q no está vacía :
5         u ←Extract(Q);  u.color ← negro
6         if u.parent ≠ ∅ then
7             T ← T ∪ {(u.parent, u)}
8         for v ∈ α[u] ∧ v.color ≠ negro :
9             if v ∉ Q then
10                Insert(Q, v)
11                v.key ← 0
12                v.parent ← u
13                DecreaseKey(Q, v, v.key)
14    return T
```

Los nodos en Q son aquellos que ya fueron conectados (linea 5). Luego en la línea 9 los que no estaban en Q son nodos que no generan ciclos, entonces los agregamos Q y botamos el muro que lo separa de u en la linea 11 y lo conectamos, ajustando su prioridad en el heap en la linea 13.

3. Dijkstra

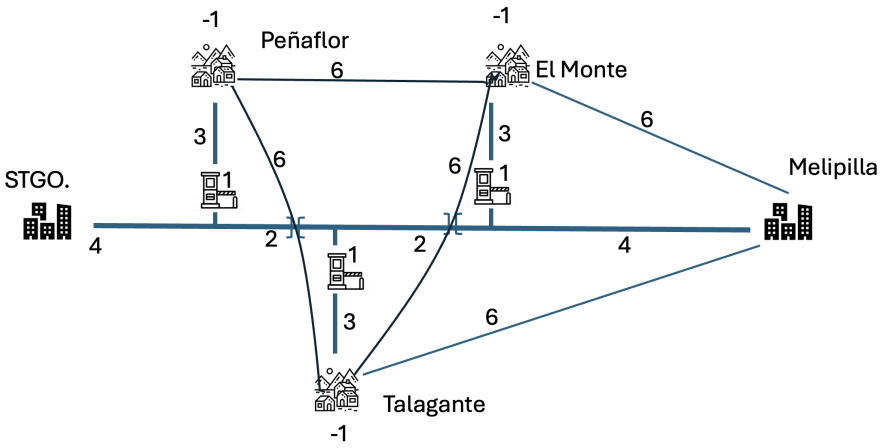
Usted es el dueño de una empresa de distribución y venta de lácteos que tiene que llevar yogur desde la ciudad de Stgo. a Melipilla. Para llegar, el conductor del camión tiene las opciones de ir por la carretera, por caminos interiores o ambos. En la figura aparecen los costos en combustible de cada segmento entre salidas laterales y el precio de los peajes. Además, se tienen las siguientes consideraciones:

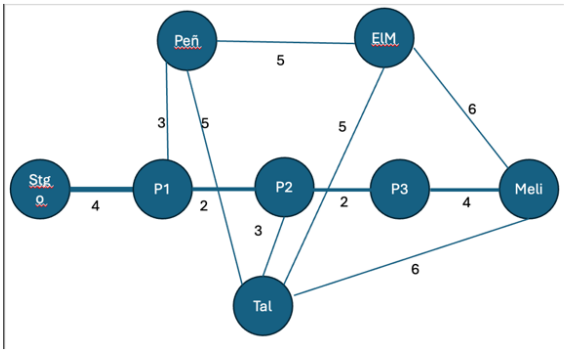
- La carretera tiene peajes laterales en la salida a cada pueblo.
- Los peajes son en un solo sentido, en la salida desde la carretera al pueblo, no a la inversa.
- El camión va vendiendo los lácteos en el viaje por lo que el costo del viaje disminuye en 1 por cada pueblo que visita (indicado con -1 en la figura).
- Los caminos interiores no se intersectan con la carretera.

(a) (2 puntos) Modele el problema como un grafo que pueda ser usado por el algoritmo Dijkstra.
Solución a) 1 pts 1 nodos-aristas, 1 costos

(b) (3 puntos) Encuentre el camino más barato para llegar de Santiago a Melipilla.
3 pts 2 pasos, 1 camino y costo

(c) (1 punto) Si con el recorrido de ida solo pasó a repartir en algunos los pueblos, modifique el problema y estrategia para que pueda recorrerlos en el camino de vuelta a Stgo. a mínimo costo. Si alcanzó a recorrerlos solo vuelva a Stgo. a mínimo costo.
Solucion: 1 pt Existen varias soluciones pero básicamente es utilizar un **algoritmo de cobertura** que textbfevite las rutas que llevan a los nodos de los pueblos ya visitados.





4. Conjuntos disjuntos

En una expedición a un lejano planeta usted descubrió una nueva familia de formas de vida parecidas a las medusas. A falta de un laboratorio mejor equipado, ha logrado establecer que dos especímenes son de la misma especie usando muestras de tejido. Si la prueba es positiva, significa que los organismos son de la misma especie.

Considere que se dispone de la secuencia de n comparaciones entre pares de especímenes y su resultado booleano, almacenadas en un arreglo $A[0 \ldots n-1]$, donde sus elementos poseen los atributos `first`, `second`, y `match` que representan respectivamente la identificación del primer especimen, del segundo y si son de la misma especie.

(a) (4 puntos) Proponga el pseudocódigo de un algoritmo `preprocess`(A) para construir una estructura de datos en tiempo $\mathcal{O}(n)$, de manera que tal estructura permita responder la consulta de comparación de especie entre dos especímenes en tiempo $\mathcal{O}(1)$ cuando A tiene n comparaciones diferentes. Proponga además el pseudocódigo de `query`(E, a, b) que responde la consulta de comparación entre a y b en tiempo $\mathcal{O}(1)$ usando la estructura E construida en su preprocesamiento. **Sugerencia:** utilice los métodos `Union` y `Find` de conjuntos disjuntos.

Solución. Interesa generar conjuntos disjuntos que representen las diferentes especies de los seres estudiados. Dado que disponemos de n comparaciones, podemos asumir que a lo más existen $2n$ especímenes individuales involucrados en dichas comparaciones y que cada uno será representado con un

Vert.	1	2	3	4
S	0,s			
p1	4,s			
p2		6, p1		
p3			8, p2	
Pe		7, p1		
El				
Ta			9,p2	
Me				12,p3

número natural correlativo. Denotamos por E a la estructura de conjuntos disjuntos (implementada como un arreglo de $2n$ celdas) poblada con el siguiente método.

```
Preprocess(A):
1    for i = 1...2n :
2        MakeSet(E, i)
3    for i = 0...n − 1 :
4        if A[i].match then
5            Union(E, A[i].first, A[i].second)
```

Luego de ejecutar el método `Preprocess`, la estructura E contiene los punteros que indican a qué conjunto pertenece cada especimen. Dado que `Union` es $\mathcal{O}(1)$, `Preprocess` es $\mathcal{O}(n)$.

Contando con E luego de llamar a `Preprocess`, las consultas se ejecutan con el siguiente método que accede a E con la función `Find`.

```
Query(E, a, b):
1    return Find(E, a) == Find(E, b)
```

La complejidad de `Find` es $\mathcal{O}(1)$, de manera que se cumple lo pedido.

Asignación de puntajes.

- 1.0 punto por inicializar cada especimen como conjunto (o por indicar que se asume que cada uno ya es un set individual, i.e. un singleton).
- 2.0 puntos por hacer unión de conjuntos cuando los especímenes tienen la misma especie.
- 1.0 punto por utilizar `Find` para construir `Query`.

Observación. Se explicitó E como argumento en los métodos de conjuntos disjuntos, pero este se puede asumir del contexto.

(b) (2 puntos) Explique cómo utilizar su solución de (a) para determinar, en base a su estructura del preprocesamiento, cuál es el grupo más numeroso de especímenes que han sido identificados como miembros de una misma especie. No requiere entregar pseudocódigo.

Solución. Una posible forma es modificar el método de preprocesamiento para llevar un conteo de integrantes de cada conjunto a medida que se procesan las comparaciones. Si existe match, se verifica si los conjuntos son diferentes y luego de unirlos, se actualiza el conteo de elementos como suma de

ambos subtotales. Gracias a que son conjuntos disjuntos, sabemos que no hay elementos repetidos. El conteo puede llevarse en un arreglo de tamaño $2n$ en paralelo al arreglo E.

Asignación de puntajes.

- 1.0 punto por idea de conteo parcial.
- 1.0 punto por describir cómo actualizar los valores al mezclar conjuntos.

Se aceptan otras propuestas siempre que sean correctas, por ejemplo, usando directamente E luego del preprocesamiento.