



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2133 — Estructuras de Datos y Algoritmos

2021 - 2

Tarea 3

Fecha de entrega código e informe: Viernes 3 de Diciembre del 2021.

Información General

Esta tarea tiene por objetivo el aplicar capacidades de análisis respecto a problemas de optimización mediante la utilización de estrategias codiciosas basadas en grafos. La tarea se compone de dos secciones independientes entre ellas. Ambas partes valen 3 puntos y el puntaje se comenta en cada sección respectiva. Es recomendable leer el enunciado con calma y tiempo para que así tengan el problema en la mente y puedan obtener las soluciones de forma más rápida y con menos presión.

Parte 1 - 60%

Objetivos

- Modelar un problema de grafos
- Utilizar y diseñar algoritmos basados en estrategias codiciosas
- Modelar un problema NP-Hard y encontrar soluciones subóptimas

(Hint: Lee el problema completo, y notarás que es más simple de lo que parece)

Introducción

Luego de sobrevivir exitosamente al percance zombie en DCCWarts, se realizó una exhaustiva investigación al respecto y concluyeron que alguien había infectado algunos puestos de Panino Fritto Wurstel Patatine. Es así como el CAH expropia toda la red de puestos y propone abrir más de estos, para minimizar los costos del grafo de puestos de Panino Fritto Wurstel Patatine y asegurar la vigilancia de estos en el campus.



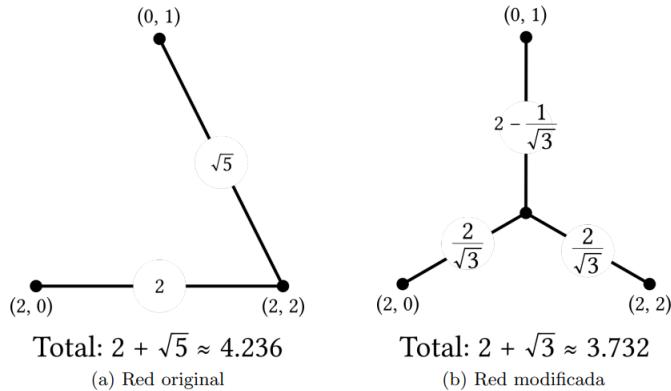
Figura 1: Panino Fritto Wurstel Patatine infectado

Problema: Rectilinear Steiner Tree

Dado n puntos en un plano \mathbb{R}^2 Se quieren conectar todos los puntos en un solo grafo tal que el costo total de la red sea mínimo. Siendo en este caso el costo de conectar dos puntos A y B, su distancia euclíadiana en el plano

$$d(A, B) = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$$

Se vio en clases que este problema se llama **Minimum Spanning Tree** (MST) y consiste en conectar los nodos de tal forma que se forme el grafo de costo mínimo. Sin embargo existen formas de mejorar el costo de un MST. Esto es, agregando m puntos nuevos al plano que disminuyan el costo general. Por ejemplo para un grafo de 3 puntos



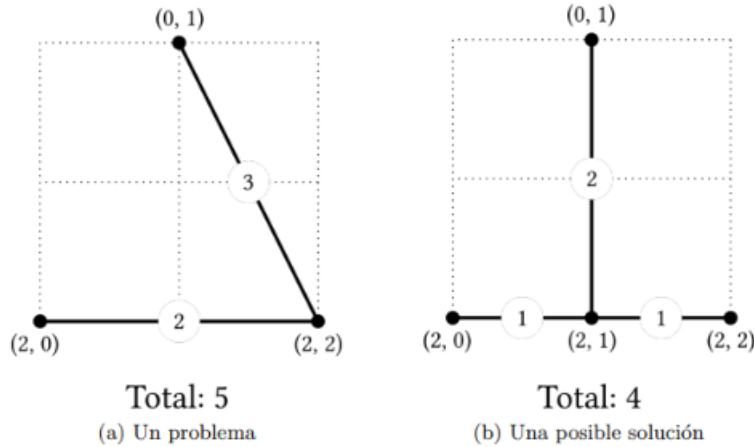
Observamos que en la red original el costo es de 4.236 dados 3 puntos. Sin embargo, si agregamos un punto intermedio m el costo disminuye a 3.732, considerablemente más bajo. Estos puntos poseen el nombre de **Steiner Points** nombrado en honor de Jakob Steiner. Y es un método usual de optimización utilizado en áreas como telecomunicaciones, logística o circuitos electrónicos. Sin embargo, puedes observar que el problema es *NP-Hard*, la forma de obtener el óptimo global sería probando todas las combinaciones posibles de Steiner Points, por lo que requerimos de algoritmos que nos permitan determinar puntos candidatos y como posicionarlos en el espacio. Así encontrando subóptimos mejores que el MST

Rectilinear Steiner Tree

Una simplificación de este problema es el rectilinear steiner tree, donde los puntos se ubican en \mathcal{N}^2 . El problema sigue siendo NP-Hard, pero permite aumentar el número de restricciones que se deben aplicar para encontrar los puntos candidatos. En este problema debido a que se ubican en posiciones discretas requiere que se utilice una métrica como la distancia [manhattan](#) para los costos

$$d(A, B) = |A_x - B_x| + |A_y - B_y|$$

Este problema es equivalente a trabajar en una grilla de puntos donde los puntos están en vértices de la grilla. Por ejemplo el caso anterior



Es posible encontrar aproximaciones al óptimo en tiempo lineal. aunque basta con quedar dentro del tiempo indicado

Deberás diseñar y escribir una solución tal que, tome un conjunto de puntos de la grilla y entregue una serie de aristas que conecte todos los puntos originales (y puntos de steiner en caso de añadir).

Ejecución

Archivo Input

Para esta parte del problema, se entregará un input iniciado por 3 números L como las dimensiones del tablero (la grilla sería $L \times L$). Luego un N que indica el número de vértices originales y por último un número C que indica el costo base del grafo esperado. Luego se presentan N líneas con las coordenadas de cada vértice en formato $Y \ X$. En resumen

- Número L que indica dimensiones del tablero
- Número N que indica el número original de vértices
- Número C que indica el costo base del MST
- N líneas con formato $Y \ X$ que indican la posición de los vértices

Ejemplo:

```
3 3 5
0 1
2 0
2 2
```

En este caso se nos indica que la grilla es de 3x3, que son solo 3 nodos y que el costo base es de 5.

Archivo Output

El output corresponde a las $n + m - 1$ aristas del MST que cubren los puntos originales (n) y los puntos de steiner (m). Para esto el output debe contener

- Número $n + m$ de vértices utilizados
- $n + m - 1$ Aristas con formato $Y_iX_iY_jX_j$ Que indica que existe una arista entre V_i y V_j

Para el ejemplo anterior

```
4
2 0 2 1
2 1 2 2
2 1 0 1
```

Acá se observa que se agrega un steiner point en la posición 2 1. Y por ende el costo total del MST sería 4 lo que es mejor que el 5 base que se tenía.

Evaluacion

Esta evaluación será particular, obtener un correcto MST base corresponderá al 50% de la nota de esta sección, para obtener el 100% deberás ir logrando mejoras porcentuales en la optimalidad de tu solución, hasta llegar al máximo, el detalle concreto se da más adelante.

Además, en caso de estar muy complicado de tiempo y si lo decides, puedes optar una forma distinta de calificación para la tarea entregando solo el MST base (osea con 0% de mejora), lo que en caso de estar correcto te garantiza un 4 en la sección de código de la tarea completa. (Ojo si haces esto, no se revisara tu parte B, por lo que tu nota máxima también será 4)

En caso de no optar a esto, la nota de esta sección se distribuye entre 1 y 8 de la siguiente forma

- Si el output no corresponde a un MST entonces tendrás un 1
- Si el costo de tu MST es igual al árbol original tendrás un 4,0
- Si el MST posee un costo entre 0% a 6% inferior al costo base, tendrás una nota hasta 8 linealmente distribuida

Parte 2 (40%)

Objetivos

- Modelar y resolver un problema a través de algoritmos codiciosos

Introducción

Luego de un arduo periodo de exámenes se acerca el fin de año académico en DCCWarts y se viene el carrete no mágico más esperado del año ... LA GALA. Sin embargo, el CAH al organizar la fiesta y empezar a decorar el gran salón se da cuenta de que al prender las luces instaladas (las cuales eran rojas, verdes y azules) venían con un hechizo que causaba que algunos no vieran las luces azules y otros no vieran las luces rojas. Quedando solo horas para el evento, Petter Potter te piden ayuda a ti, experto en algoritmos codiciosos, para que programes una solución que conecte los cables de la forma más eficiente posible y que de esta forma no existan personas que vean luces sin conectar entre sí, ya que por esta velada la magia está prohibida y luces no conectadas flotando en el salón podrían generar pánico.

Problema

Imagina que las luces en el salón están puestas en una recta infinita, donde se colocaran n luces de 3 colores distintos. Considera las luces se ubican en alguna posición i de la recta, y nunca habrá más de una luz por posición

R	G	B	G	R					
0	1	2	3	4	5	6	7	8	9

En este caso diríamos que existe una luz roja en 0, una verde en 2 y así sucesivamente. El problema consiste en conectar todas las luces de tal forma que el costo sea mínimo y las luces sigan conectadas en caso de eliminar todas las luces rojas o azules. Por ejemplo la solución del caso anterior ha de ser consistente y mantener conectados estos 2 casos.

G	B	G							
0	1	2	3	4	5	6	7	8	9

(Recta con Rojos invisibles)

R	G		G	R					
0	1	2	3	4	5	6	7	8	9

(Recta con Azules invisibles)

Es importante además aclarar que se pide que estas conexiones sean de costo mínimo. y el costo de una conexión se calcula simplemente con la distancia entre ambos puntos en la recta.

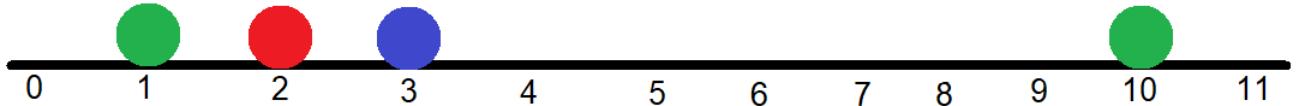
Ejemplo del problema: Consideremos el siguiente input

```

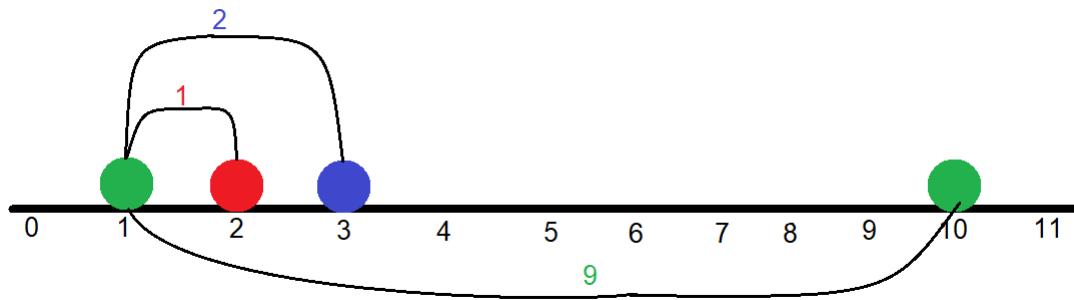
BEGIN INPUT
4
1 g
2 r
3 b
10 g
END INPUT

```

Este input indica que en nuestra recta tenemos 4 luces, que están en las posiciones $G = \{1, 10\}$, $B = \{3\}$ Y $R = \{2\}$. Gráficamente, esto se representa como



El programa ha de determinar alguna configuración de conexiones tal que el costo de resolver el problema sea mínimo. En este ejemplo en la solución es 12, y gráficamente las conexiones se ven de la siguiente forma



Ejecución

Input

Como Input se entregara un N indicando el número de puntos, y una colección de puntos en el estilo $i \ C$, con i siendo la posición en la recta infinita y C su color. Por ejemplo

```

4
1 g
3 b
2 r
10 g

```

Output

Como output deberás entregar un N indicando el número de aristas obtenidas seguido por N líneas indicando cuáles son las conexiones. En el ejemplo anterior

3
1 g 10 g
1 g 2 r
1 g 3 b

Evaluacion

La nota de esta sección completa corresponde a un 40% del total de la tarea cuyo puntaje por test se asigna de la siguiente forma

- Si los puntos no están correctamente conectados (ie que al desconectar todos los azules quedan puntos sin conectar) entonces se obtendrá un 0 en dicho test
- Si se conectan todos los puntos en un **costo óptimo**, entonces se obtiene un 1 en dicho test

Recomendación de tus ayudantes

Las tareas requieren de mucha dedicación de tiempo generalmente, por lo que desde ya te recomendamos distribuir tu tiempo considerando los plazos definidos. Así mismo, te recomendamos fuertemente que antes de empezar a programar tu tarea, leas el enunciado y te dediques a entender de manera profunda lo que te pedimos. Una vez que hayas comprendido el enunciado, dedica el tiempo que sea necesario para la planificación y modelación de tu solución, para posteriormente poder programar de manera eficiente. Estos son consejos de tus ayudantes que te pueden ayudar a pasar el ramo :)

Uso de memoria

Parte de los objetivos de esta tarea, es que implementen en la práctica el *trade-off* entre memoria y tiempo, es por esto que independiente del test, tendrán como máximo 1.2 GB de memoria RAM disponible. Pueden revisar la memoria que utiliza su programa con el comando `htop`. Además, el servidor avisará en caso de superar el máximo permitido.

Eficiencia

Se solicita que el programa sea eficiente. No podrá tomar más de 10 segundos `user + sys`, pueden utilizar `time` seguido del comando de ejecución de su programa para revisarlo.

Evaluación

La nota de tu tarea se descompone como se detalla a continuación:

- 60% Parte A y 40% parte B. donde en la parte A puedes llegar hasta el 8,0.
- Para disminuir la carga de la tarea se crea la excepción de solo realizar el MST básico de la parte A. En dicho caso puedes optar a un 4. Sin embargo, al hacer esto debes explicitarlo en el siguiente formulario [LINK AL FORM](#) y no podrás optar al puntaje de la parte B

Recordar, como se menciona arriba, que si lo deseas puedes enviar el MST del apartado 1 de forma exclusiva, en caso de estar correcto tendrás una nota 4 como nota de código.

Para esta tarea el informe es Opcional, esto es. Puedes realizarlo si deseas aumentar tu nota (si es que el informe te perjudica, no será considerado). La evaluación se describe de la siguiente forma

- 80% Nota Código
- 20 % Nota Informe en caso de beneficiar

Informe Opcional

Tu informe ha de contener como mínimo un análisis considerando los siguientes tópicos

- Ventajas de utilizar algoritmos codiciosos en ambos problemas presentados
- Para la parte 1:
 - Algoritmo para obtener el MST base y Su complejidad
 - Descripción de una estrategia para generar los Steiner Points
- Para la parte 2:
 - Algoritmo para obtener la solución al problema y su complejidad

Entrega

Código: GIT - Rama master del repositorio asignado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Informe (Opcional): SIDING - En el cuestionario correspondiente, en formato PDF. Sigue las instrucciones del cuestionario. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Atraso: A lo largo del semestre tendrás 4 días de gracia, los cuales podrás utilizar en caso de que no alcances a entregar alguna tarea en el tiempo indicado. Para esto, puedes usar un máximo de 2 días en una tarea, sin importar si tienes más días disponibles, y tendrás que avisar si quieres que se revise un commit posterior a la fecha de entrega en el formulario que se mandará el día siguiente. Cabe destacar que si entregas a las 00:01 hrs perderás un día en caso de llenar el formulario, y no será revisado ese commit en caso de que decidas no contestarlo. Por otro lado, si se te acaban los 4 días y entregas una tarea atrasada, entonces tendrás la calificación mínima **sin derecho a reclamo**. Cabe recalcar que el informe solo se puede realizar en un **máximo de 6 hojas**, de pasarse no podrán objetar por posibles descuentos.

Integridad académica

Este curso se adscribe al Código de Honor establecido por la Escuela de Ingeniería. Todo trabajo evaluado en este curso debe ser hecho **individualmente** por el alumno y **sin apoyo de terceros**. Se espera que los alumnos mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería.

¡Éxito! :)

Y para quienes llegaron hasta acá...

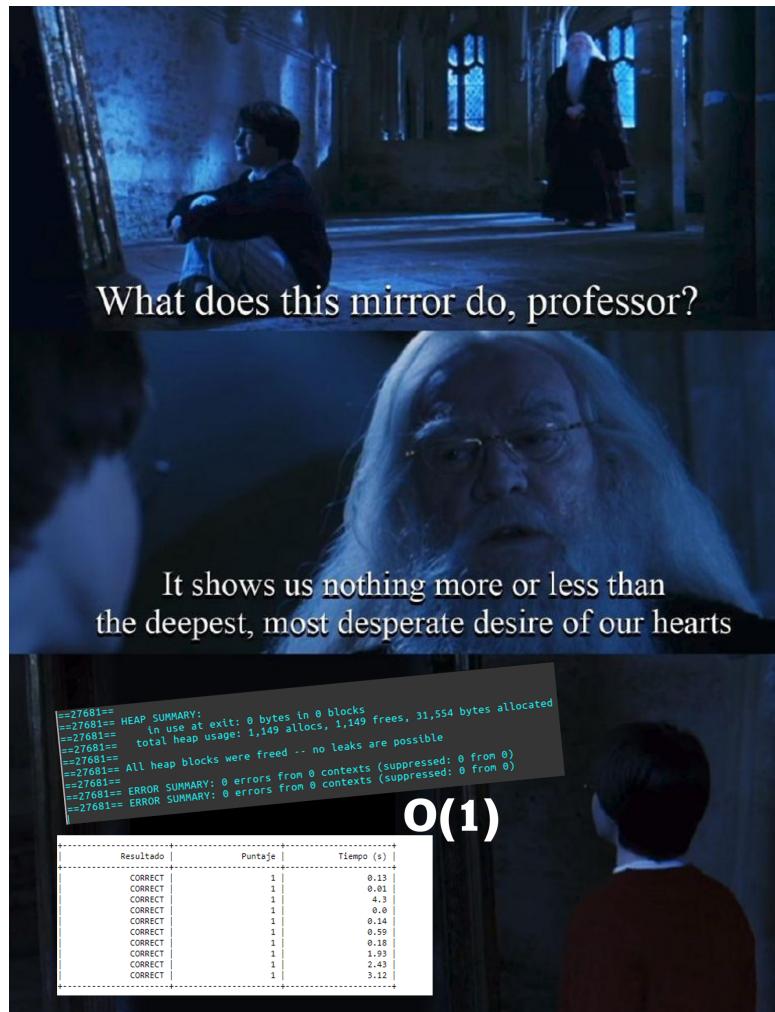


Figura 2: Meme gracioso