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

IIC2133 – Estructuras de Datos y Algoritmos

2024 - 1

Tarea 3

Fecha de entrega código: 21 de Junio

Link a generador de repos: CREAR REPOSITORIO AQUÍ

Objetivos

- Emplear diferentes técnicas algorítmicas para encontrar soluciones óptimas.
- Explorar el desarrollo de algoritmos que se emplean sobre grafos.

Introducción

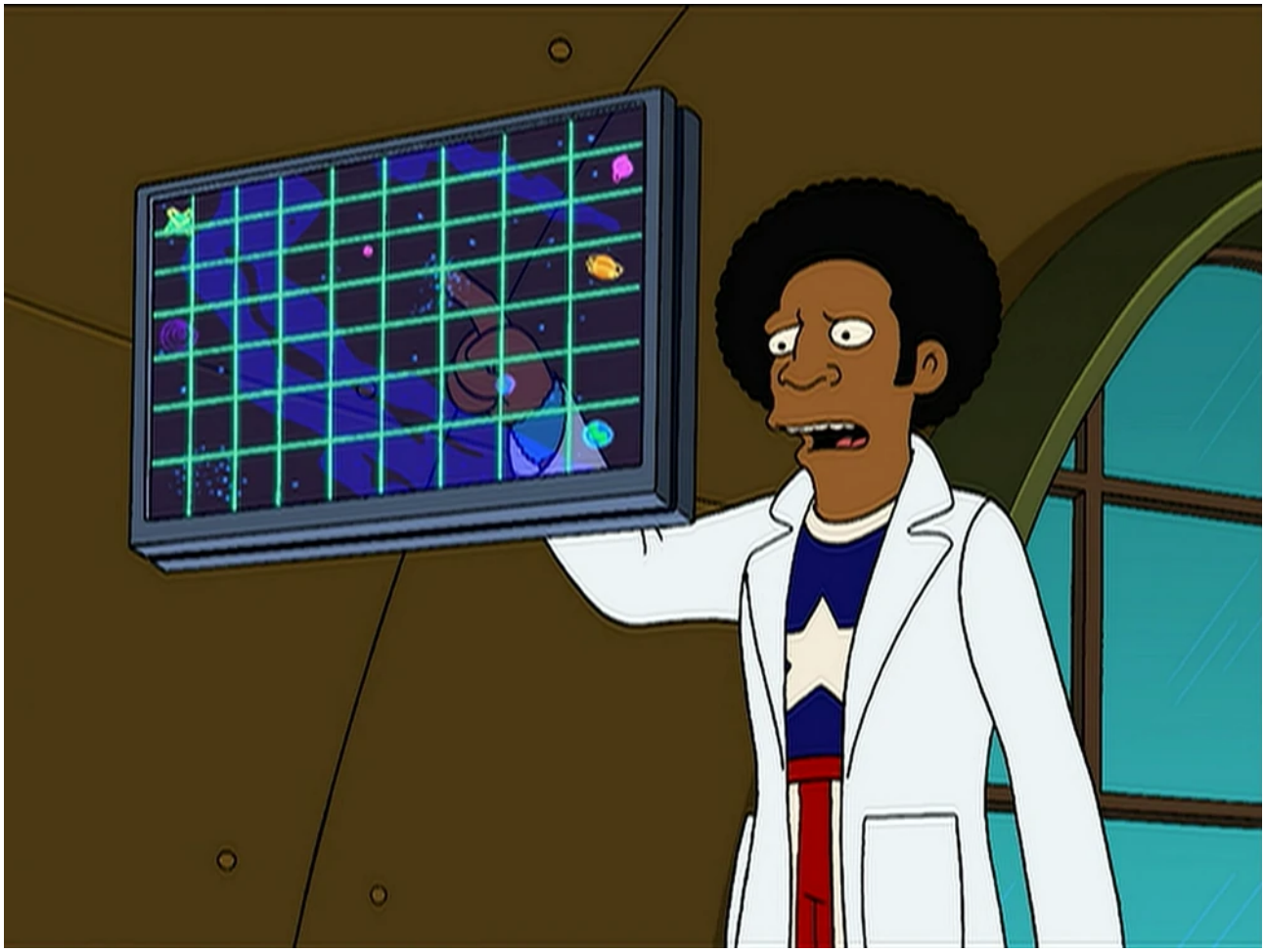

Figura 1: Profesor introduciendo a Planet EDDXPRESS nuevos espacios por explorar

Parte 1: Cerebros Codicioso

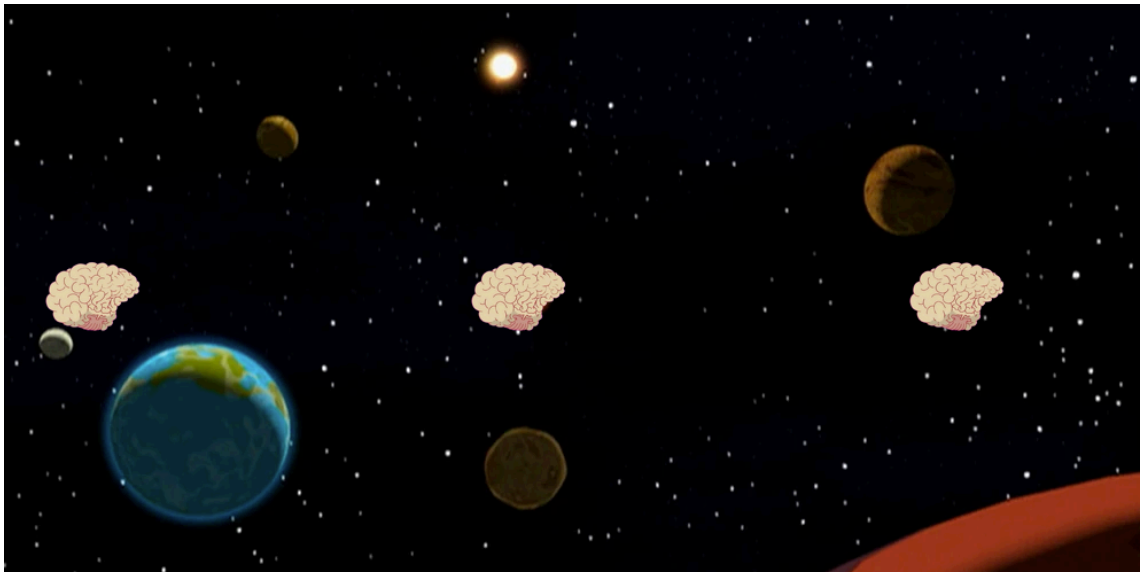

Figura 2: Cerebros ubicados estrategicamente para cubrir planetas

En un rincón distante del universo, Bender, en un estado de embriaguez, cometió un acto insensato al secuestrar al Nyctalus noctula Cerebral, una criatura sagrada de pequeño tamaño que residía en el desván del templo neuronal de la metrópolis. Este misterioso ser nocturno era venerado como un símbolo de conexión cósmica y sabiduría por los cerebros. Al descubrir el ultraje, los cerebros decidieron responder con ira, desencadenando una invasión galáctica como represalia, empeñados en encontrar a su preciada criatura en todos los rincones del universo donde Bender pudiera haberla llevado.

Los cerebros, reconociendo tu reputación como programador y defensor de todas las criaturas, incluidas las nocturnas, te solicitan ayuda para optimizar sus estrategias de guerra. Con su sabiduría en actos bélicos y tu experiencia en programación, **buscan minimizar el uso de recursos** y diseñar un plan *ambicioso* para distribuir sus naves cerebrales de la forma más eficiente posible al rededor de toda la galaxia.

Problema

Para abarcar este problema se te entregará una cantidad de planetas P seguido de las posiciones de planetas p_i, en una dimensión, y un **rango** R de cobertura de naves cerebrales. Se dice que un planeta se encuentra cubierto por los cerebros cuando hay una nave **a lo más** a una distancia R de este. Deberás obtener la cantidad mínima de cerebros C con rango R que son capaces de cubrir todos los planetas P.

Ejecución

Tu programa se debe compilar con `make` y debe generar el ejecutable `Nyctalus_search`, que se ejecuta con:

```
./nyctalus_search input.txt output.txt
```

Se leerá de un archivo de input y se escribirá en el archivo de output, entregados como argumentos del programa. En caso de querer correr el programa para ver los leaks de memoria utilizando deberás utilizar:

```
valgrind ./nyctalus_search input.txt output.txt
```

Input y Output

A continuación presentamos un ejemplo de un input y output válido para el problema.

input

```
3
1 2 -1
2
```

output

```
1
```

Las formas de visualizar el problema anterior serían las siguientes:

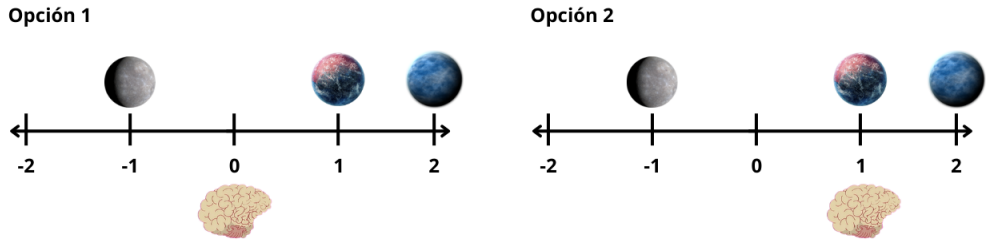

Figura 3: Distribución input de ejemplo

Al ser el rango $R = 2$ tenemos basta con solamente una nave posicionada en la posición 0 o posicionada en 1, por lo que el output en esta situación termina siendo 1

Otros ejemplo, para una cantidad óptima de cerebros igual a 2 es:

input

```
7
1 -2 -4 3 0 -5 4
2
```

output

```
2
```

Como podemos ver existen varias configuraciones que permiten posicionar 2 cerebros de modo que se cubren todos los planetas. Notar que a un planeta puede cubrirlo **al menos** un cerebro.

Otro ejemplo ilustrativo, para una cantidad 3 de cerebros es:

input

```
7
4 0 12 11 9 7 3
2
```

output

```
3
```

Parte 2: Acumulando Basura

Figura 4: Nave que transporta los diferentes tipos de basura

En el año 3000, Nueva Nueva York enfrenta un problema insólito que podría poner en riesgo el equilibrio de la ciudad: la gestión eficiente de sus basurales. La basura, en esta futurista metrópolis, no solo es una molestia, sino que se ha convertido en un recurso valioso para la producción de energía y materiales reciclables. Sin embargo, no todas las montañas de basura son iguales; **cada tipo de basural tiene una cantidad distinta de masa** que debe ser cuidadosamente contabilizada.

El profesor Farnsworth, con su inagotable afán por la ciencia y el progreso, ha ideado un plan brillante para optimizar el uso de los basurales. Ha descubierto que, mediante un cálculo preciso y una programación meticulosa, es posible determinar la cantidad mínima de basurales necesarios para obtener una masa de basura exacta M, crucial para mantener las operaciones de reciclaje y energía al máximo rendimiento.

Con la ayuda de su intrépido equipo de Planet Express, el profesor Farnsworth se embarca en una misión para desarrollar un sistema que pueda resolver este enigma.

Problema

Dado una lista B de **tipos** de basurales distintos, donde cada tipo aporta una **cantidad** b_i de masa de basura, obtén la cantidad mínima de basurales (distinto o igual tipo) que logren obtener exactamente la masa dada M.

Si es que no existe combinación que permita llegar exactamente a la masa M, se deberá imprimir -1.

Ejecución

Tu programa se debe compilar con `make` y debe generar el ejecutable `trash`, que se ejecuta con:

```
./trash input.txt output.txt
```

Se leerá de un archivo de input y se escribirá en el archivo de output, entregados como argumentos del programa. En caso de querer correr el programa para ver los leaks de memoria utilizando deberás utilizar:

```
valgrind ./trash input.txt output.txt
```

Input y Output

input

```
12
2 4 4 5 1 2 0 70 10 15 31 11
13
```

output

```
2
```

Se usa el último, que aporta 11, y el primero, que aporta 2, para llegar a 13.

input

```
7
9 2 7 2 21 13 2
12
```

output

```
6
```

Se puede usar 6 veces cualquier tipo de basural que aporta 2.

input

```
6
14 7 30 15 8 7
13
```

output

```
-1
```

No hay como llegar a 13 a partir de los tipos de basurales dados.

Parte 3: Identificación de Regiones

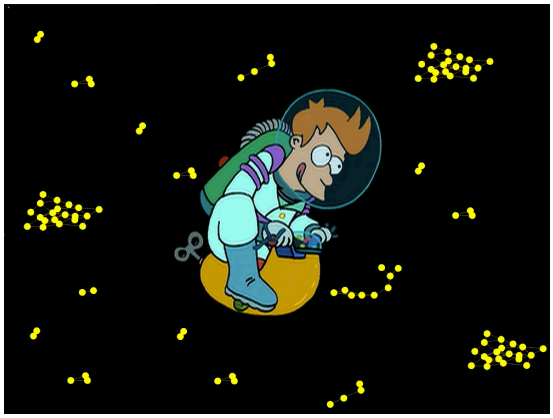

Figura 5: Fry viajando a través de grupos de estrellas

Los malvados cerebros han nuevamente atacado las oficinas de EDDXPress. Antes de los programadores puedan defenderse con sus poderosos algoritmos, estos se escondieron entre las estrellas sin dejar ningún rastro. El alto comando de la empresa quiere contraatacar lo antes posible y necesita saber cuáles grupos de estrellas pueden estar escondidos los cerebros. Por suerte, con tu avanzado conocimiento de grafos, propones modelar las rutas de comercio entre las estrellas. De esta forma puedes identificar los grupos no conectados de estas para empezar la búsqueda y acabar de una vez por todas con la tiranía neuronal.

Problema

Se te entregará una representación de un grafo no dirigido de la siguiente forma:

- Primero, un número N que indica la cantidad de nodos del grafo, donde cada uno tendrá un ID que será desde 0 a $N - 1$.
- Luego, se te entregará un número A, seguido de A líneas que representan las aristas de los nodos. Cada línea tendrá 2 IDs, y representan una conexión entre los nodos de dichos IDs.

Debes determinar cuántos grupos no conectados de nodos hay.

Ejecución

Tu programa se debe compilar con `make` y debe generar el ejecutable `find_groups`, que se ejecuta con:

```
./find_groups input.txt output.txt
```

Se leerá de un archivo de input y se escribirá en el archivo de output, entregados como argumentos del programa. En caso de querer correr el programa para ver los leaks de memoria utilizando deberás utilizar:

```
valgrind ./find_groups input.txt output.txt
```

Input y Output

input

```
10
6
0 1
2 4
5 7
2 3
4 0
9 6
```

output

```
4
```

Informe

Además del código de la tarea, se debe redactar un **breve** informe que explique las **estrategias y algoritmos relevantes** que implementarías para resolver cada parte, incluyendo la **complejidad** de tiempo y espacio de estos. Debe quedar claro **como** resolverías o resuelves cada problema que identifiques en el enunciado, mencionando las estructuras de datos y algoritmos usadas, referenciando contenido de clases si es necesario. Se espera que en la motivación se mencione por qué estas son relevantes.

Se debe entregar como archivo PDF generado por LaTeX junto a la tarea (por lo cual tienen la misma fecha de entrega) y con nombre **`informe.pdf` en la raíz del repositorio**.

Les recomendamos comenzar la tarea escribiendo parte de este informe, ya que pensar y armar un esquema antes de pasar al código ayuda a estructurarse al momento de programar y adelantar posibles errores que su solución pueda tener. Además, es ideal separar el informe en cada parte de la tarea, y seguir un hilo coherente.

Se encuentra un template en este link para el uso en la tarea. En la documentación de Overleaft se indica como clonar un template. No se aceptarán informes de más de **3 planas** (excluyendo la página de referencias), informes ilegibles, o generados con inteligencia artificial.

Evaluación

La nota de tu tarea es calculada a partir de testcases de Input/Output y además de un breve informe escrito en LaTeX que explique un modelamiento sobre como abarcar el problema, las estructuras de datos a utilizar, etc. La ponderación se descompone de la siguiente forma

- 20 % Por el informe de modelamiento, que debe incluir motivación, modelamiento y conclusión.
  - 5 % Por coherencia y seguir el formato dado.
  - 15 % Por el informe dado, siguiendo lo mínimo pedido previamente.
    En esta tarea, el puntaje es **equitativamente distribuido en cada parte**.
- 70 % a la nota entre las partes 1, 2 y 3, donde se evalúa por porcentaje de tests pasados.
  - 25 % a la nota de los tests de la parte 1
  - 20 % a la nota de los tests de la parte 2
  - 25 % a la nota de los tests de la parte 3
- 10 % Por manejo de memoria
  - 5 % Por no poseer leaks de memoria
  - 5 % Por no poseer errores de memoria

Para cada test de evaluación, tu programa deberá entregar el output correcto en **menos de 10 segundos** y utilizar menos de 1 GB de RAM[1]. La complejidad de tu solución deberá ser acorde a las competencias del curso.

Recomendación de tus ayudantes

Las tareas generalmente requieren de mucha dedicación, 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, con anticipación, el enunciado detenidamente 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, modelación y elaboración de tu informe, para posteriormente poder programar de manera más eficiente. Por tu propio bien, esperamos que no estés leyendo esto a días de la entrega.

Entrega

Código: GIT - Rama principal del repositorio asignado, que debes generar con el link dado al principio de este enunciado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Atraso: Esta tarea considera política de atraso y de cupones, especificada en el repositorio del curso.

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.

[1] Puedes revisarlo con `valgrind --tool=massif --massif-out-file=massif.out` y luego `ms_print massif.out`