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

IIC2133 – Estructuras de Datos y Algoritmos

2024 - 1

Tarea 1

Fecha de entrega código: 24 de abril

Link a generador de repos: AQUí

Objetivos

- Aplicar Heaps y Árboles de Búsqueda para resolver problemas.
- Utilizar técnicas algorítmicas para cumplir una complejidad dada.

Introducción

En el vibrante universo intergaláctico, EDDXPRESS se encuentra en una encrucijada. A pesar de su historial de entregas impecables, la competencia feroz de MomCorp amenaza con eclipsar su brillo. Con el fin de mantenerse a flote y destacarse, EDDXPRESS busca diversificar sus servicios.

La oportunidad surge en dos frentes diferentes pero igualmente prometedores. Por un lado, la empresa decide asociarse con los corredores de mercado ilegal intergaláctico, ofreciendo un servicio de entrega rápido y seguro para las transacciones en este mercado clandestino. Esto les permite incursionar en un nuevo territorio y aprovechar un nicho lucrativo, aunque riesgoso.

Por otro lado, EDDXPRESS se une al prestigioso DCCMuseum para prestar servicios de almacenamiento temporal de las cabezas de los personajes históricos del universo DCC. Esta asociación no solo les ofrece una oportunidad única de asociarse con un importante centro cultural, sino que también les permite expandir su presencia en el ámbito del almacenamiento.

Sin embargo, gestionar estos dos servicios simultáneamente presenta un desafío logístico considerable para el equipo de EDDXPRESS. Con el valor de las acciones en juego y la reputación de la empresa en peligro, necesitan un sistema que les permita mantener el control tanto de las operaciones financieras como del cuidado de las valiosas cabezas del museo.

Para resolver este problema, EDDXPRESS busca tu ayuda, dado que tu reputación de programador te procede. Por lo que es necesario que utilices tus conocimientos de programación y así desarrollar diferentes sistemas integrales que no solo gestionen las transacciones de mercado ilegal con eficacia, sino que también garantice la integridad del almacenamiento de las cabezas y la eficiencia de búsqueda de información.

Parte 1: Heap Market

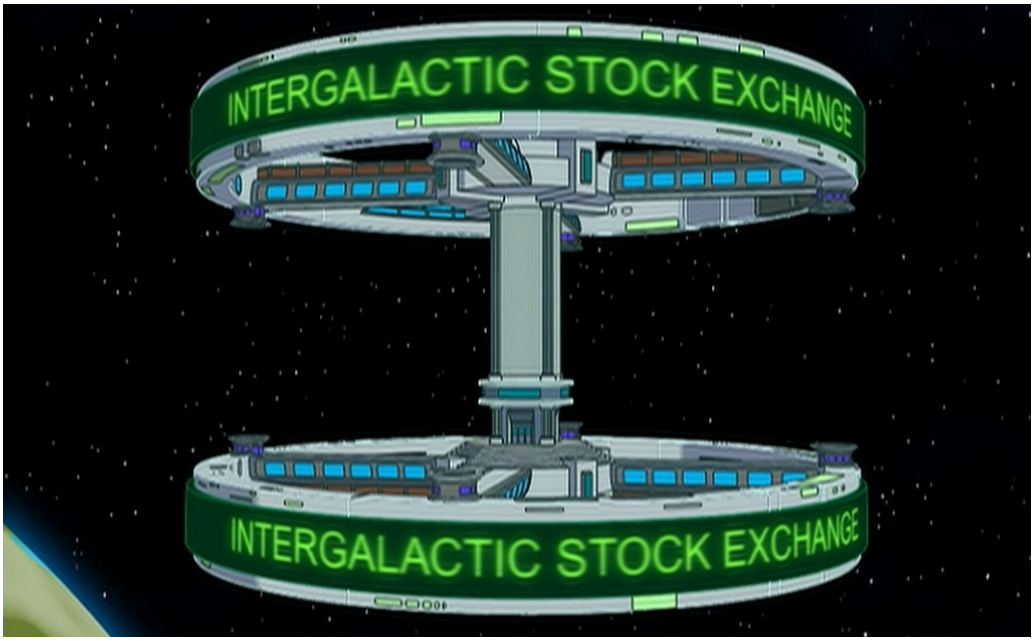

Figura 1: El mercado intergaláctico.

En el agitado y futurista mundo de Nueva Nueva York, donde los robots pasean por las calles y los alienígenas hacen negocios junto a los humanos, se encuentra el mercado de heaps más caótico de la galaxia. Conocido como Heap Market, este mercado es un crisol de órdenes de compra y venta que fluyen constantemente como un río de datos a través de las vastas redes informáticas del futuro. Aquí, los ciudadanos de todas las especies buscan obtener ganancias, ya sea comprando stocks de tecnología de vanguardia o vendiendo recursos raros extraídos de los confines del universo.

En el corazón del Heap Market, los comerciantes virtuales compiten ferozmente por ejecutar sus órdenes antes que los demás, mientras que los algoritmos de inteligencia artificial trabajan sin descanso para encontrar las mejores oportunidades de inversión. Las órdenes de compra y venta se registran en tiempo real, con montos que fluctúan tan rápido como la velocidad de la luz, y las ejecuciones se llevan a cabo en un instante, determinadas por algoritmos complejos y reglas estrictas.

Sin embargo, incluso en este mundo futurista, el caos y la imprevisibilidad son moneda corriente. A veces, múltiples órdenes llegan al mismo tiempo con el mismo monto, desatando una carrera frenética por ser el primero en ejecutar la transacción. Otras veces, las rivalidades entre especies conducen a disputas sobre el precio justo, y las negociaciones se convierten en un juego de astucia y estrategia.

Pero a pesar de todo el frenesí y la confusión, el Heap Market sigue siendo un lugar emocionante y lleno de oportunidades para aquellos lo suficientemente valientes como para sumergirse en sus profundidades. En este mundo de ciencia ficción, donde las leyes de la física y la lógica se doblan a voluntad, el mercado de heaps es un reflejo del constante flujo y cambio del universo, donde solo los más astutos pueden esperar triunfar.

Problema

Antes de ver el sistema de la simulación, es importante entender como funcionará este mercado.

- Tanto órdenes de compra y órdenes de venta, se registran con el monto (precio) en que buscan comprar o vender, y el número de la cuenta que les corresponde. Se asume cantidad unitaria[1].
- Se dice que las órdenes son activas si todavía no han sido ejecutadas.
- Un par de órdenes activas de compra y venta se ejecutan cuando la orden de compra tiene mayor o igual monto que la orden de venta.
- El monto en el que se ejecuta es el de la última orden del par ejecutado.

Puede pasar que existan varias órdenes de compra o venta con el mismo monto, en ese caso, se tiene que ejecutar la orden que llegó primero (FIFO). También, si es que existen múltiples órdenes de compra con mayor monto que una orden de venta, o múltiples órdenes de venta con menor monto que órdenes de compra, se ejecuta la orden de compra de mayor monto, o la orden de venta de menor monto, respectivamente.

Ejemplo de operaciones

Libro de órdenes en un instante	**Operación (Evento)**	**Libro de órdenes luego de la operación**	**Transferencia Realizada**
Monto | Usuario Órdenes de compra \$10 | 1 \$11 | 2 \$12 | 3 Órdenes de venta \$13 | 4 \$14 | 5 \$15 | 6	Nueva orden de compra con usuario 7 y monto \$14	Monto | Usuario Órdenes de Compra \$10 | 1 \$11 | 2 \$12 | 3 Órdenes de Venta \$14 | 5 \$15 | 6	Orden de compra con usuario 7 se ejecuta con orden de venta de usuario 4 a monto \$14
Monto | Usuario Órdenes de Compra \$10 | 1 \$11 | 2 \$12 | 3 Órdenes de Venta \$14 | 5 \$15 | 6	Nueva orden de compra con usuario 8 y monto \$12	Monto | Usuario Órdenes de Compra \$10 | 1 \$11 | 2 \$12 | 8 \$12 | 3 Órdenes de Venta \$14 | 5 \$15 | 6	No se ejecuta orden, orden nueva de usuario 8 queda luego de la del usuario 3, dado que la orden del usuario 3 llegó antes
Monto | Usuario Órdenes de Compra \$10 | 1 \$11 | 2 \$12 | 8 \$12 | 3 Órdenes de Venta \$14 | 5 \$15 | 6	Nueva orden de venta con usuario 5 y monto \$10	Monto | Usuario Órdenes de Compra \$10 | 1 \$11 | 2 \$12 | 8 Órdenes de Venta \$14 | 5 \$15 | 6	Orden de venta con usuario 5 se ejecuta con orden de compra de usuario 3 a monto \$10

Ayuda: Es posible pensar las tablas de las órdenes de compra y venta como colas distintas. En la de compra, la prioridad va desde el monto mayor hasta el monto menor, y para las de venta, es al revés, la prioridad va desde el monto menor hasta el monto mayor. Se ejecuta una orden cuando los inicios de las colas logran converger en algún monto.

[1] Las órdenes de compra o venta reales especifican cantidad y precio, aquí se asumirá que la cantidad de stocks es siempre 1.

Eventos

Se te entregarán N eventos, que pueden ser:

`STATUS`

Este comando indica imprimir el estado del mercado, por lo que debe mostrar el siguiente output, donde se muestra la orden de compra mayor activa y la orden de venta menor activa.

output

```
1 Estado del mercado
2     {oCompraMayor.monto} por {oCompraMayor.nCuenta}
3     SPREAD {oVentaMenor.monto - oCompraMayor.monto}
4     {oVentaMenor.monto} por {oVentaMenor.nCuenta}
```

Aquí, `oCompraMayor` es la orden de compra activa con el mayor monto, y análogamente, `oVentaMenor` es la orden de venta activa con el menor monto.

En casos de no haber órdenes de compra la línea 2 y 3 no se imprimirían. Similar a esto, en caso de no existir órdenes de venta, no se imprime la línea 3 y 4. Ahora, en caso de que no existan órdenes activas ni de compra ni de venta, se debe imprimir lo siguiente:

output

```
1 Mercado inactivo: No hay registros de transacciones.
```

`SELL {monto} {nCuenta}`

Este comando indica el registro de una orden de venta.

output

```
1 User {nCuenta} vende a monto {monto}
```

`BUY {monto} {nCuenta}`

Este comando indica el registro de una orden de compra.

output

```
1 User {nCuenta} compra a monto {monto}
```

Además, en base al funcionamiento descrito en la explicación del problema, al simular `SELL` o `BUY` se debe **intentar** ejecutar la orden recién registrada.

En caso de que se ejecute una orden, se debe mostrar:

output

```
1 Orden Ejecutada: {oVenta.nCuenta} -> {oCompra.nCuenta}: {monto}
```

Cada uno de los eventos debe tener complejidad a lo más $O(\log n)$, con n la cantidad actual de órdenes.

Ejecución

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

```
./market 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 ./market input.txt output.txt
```

Input y Output

Se te entregarán N eventos, los que se verían con el siguiente formato[2]:

input

```
N
SELL 1250 1
SELL 1201 2
BUY 1300 5
SELL 1250 4
BUY 900 6
SELL 1420 2
STATUS
BUY 1320 7
...
```

output

```
User 1 vende a monto 1250
User 2 vende a monto 1201
User 5 compra a monto 1300
Orden Ejecutada: 2 -> 5: 1300
User 4 vende a monto 1250
User 6 compra a monto 900
User 2 vende a monto 1420
Estado del mercado
    900 por 6
    SPREAD 350
    1250 por 1
User 7 compra a monto 1320
Orden Ejecutada: 1 -> 7: 1320
...
```

[2]En este ejemplo hacemos comentarios con #.

Parte 2: A buscar cabezas

Figura 2: Pasillo de presidentes en el museo de cabezas.

Problema

Dirigido por el excéntrico y genial Profesor Farnsworth, el Laboratorio de Cabezas es el epicentro de la innovación científica y la resolución de misterios en el universo conocido. Equipado con la última tecnología en sistemas de almacenamiento y búsqueda, el laboratorio es capaz de rastrear las cabezas perdidas en un abrir y cerrar de ojos, gracias a sus algoritmos de búsqueda especializados y sus poderosos árboles de búsqueda multidimensionales de la galaxia Omicron Persei 8.

Cada búsqueda es una nueva aventura para el equipo de élite del laboratorio, encabezado por el valiente capitán Turanga Leela, la leal e ingeniosa Amy Wong y, por supuesto, el eternamente confundido pero sorprendentemente útil Philip J. Fry. Juntos, se sumergen en las profundidades del cosmos, navegando por los rincones más remotos de la galaxia para encontrar cabezas perdidas, enfrentándose a alienígenas hostiles, paradojas temporales y bromas cósmicas a lo largo del camino.

En este universo lleno de maravillas y peligros, la búsqueda de cabezas es mucho más que una simple tarea científica; es una misión para mantener el orden y la cordura en un universo en constante cambio. Y aunque el viaje puede ser arduo y lleno de desafíos, el equipo del Laboratorio de Cabezas nunca retrocede ante la adversidad, siempre dispuesto a enfrentar lo desconocido con valentía y determinación, guiados por la luz de la ciencia y la amistad.

Cabe destacar que este problema de búsqueda debe cumplir con los principios de eficiencia y optimización aprendidos en clases, garantizando que el proceso de búsqueda sea lo más rápido y preciso posible, utilizando algoritmos y estructuras de datos avanzadas para manejar grandes volúmenes de datos con facilidad y precisión.

Eventos

Se entregarán N cabezas con los siguientes atributos:

`SAVE {id} {nombre} {año} {nRegion} {x} {y}`
Donde se tiene que

- `id` es un número único por cabeza,
- `nombre` es su nombre, que no será más largo que 64 caracteres y no tendrá espacios,
- `año` es el año de origen,
- `nRegion` es el número de la región al que pertenece, y,
- `x` e `y` son los números enteros que representan las coordenadas en la que se encuentra.

Luego se te entregaran B búsquedas que tendrás que realizar, donde los tipos de búsquedas son:

`WITH-ID {id}`
Búsqueda por ID. Como el ID es único, se debe mostrar uno o ningún resultado.

`WITH-YEAR {año}`
Retorna todas las cabezas con el año de origen dado. El orden de los resultados debe ser por el **orden de llegada**, es decir, el que fue registrado antes debe ir antes.

`WITH-YEAR-DISTRICT {año} {nRegion}`
Retorna todas las cabezas con el año de origen y número de región dado. Los resultados deben presentarse por orden de llegada.

`IN-X-RANGE {A} {B}`
Retorna todas las cabezas con $A \leq x < B$. El orden de los resultados debe ser primero por **menor** valor `x`, y luego, por orden de llegada.

`IN-CIRCLE {x} {y} {r}`
Retorna todas las cabezas que estén dentro del círculo con centro (x, y) y radio r, incluyendo el perímetro. El orden de los resultados debe ser primero por `x` y luego `y`. Puedes asumir que no se repite la posición.

El formato de retorno de todas las consultas debe ser:

output

```
{BUSQUEDA_REALIZADA}: {CANTIDAD_DE_RESULTADOS}
{resultado1.id} {resultado1.nombre}
{resultado2.id} {resultado2.nombre}
...
```

Hint: Puedes usar distintos tipos de árboles para cada una de las búsquedas. Investiga de como se puede trabajar con múltiples valores en los árboles de búsqueda. Aunque toda esta parte se puede resolver con los árboles vistos en clases, puede ser útil investigar otro tipo de árboles de búsqueda, como RangeTree.

Importante: Deberán usar estructuras de datos con complejidad de búsqueda promedio a lo más $O(log(n))$.

Bonus: Más info en este link

Ejecución

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

```
./hsearch 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 ./hsearch input.txt output.txt
```

Input y Output

El archivo de input comenzará con el número `N` de cabezas a registrar, y luego las `N` cabezas con todos sus atributos, Luego, se entregará el número B de búsquedas, y las B búsquedas correspondientes.

input

```
N
SAVE 27 EricCartman 1865 2 64 72
SAVE 4 AbrahamLincoln 1865 23 30 802
...
B
With-ID 27
WITH-YEAR 1865
WITH-YEAR-DISTRICT 1865 2
IN-X-RANGE 50 200
IN-CIRCLE 150 700 150
...
```

output

```
With-ID 27: 1
27 EricCartman
WITH-YEAR 1865: 2
27 EricCartman
4 AbrahamLincoln
WITH-YEAR-DISTRICT 1865 2: 1
27 EricCartman
IN-X-RANGE 50 200: 2
4 AbrahamLincoln
27 EricCartman
...
```

Informe

Además del código de la tarea, se debe redactar un **breve** informe que explique como se pueden implementar algunas estructuras y algoritmos del enunciado. Les recomendamos **comenzar la tarea escribiendo 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.

Este informe se debe entregar en un archivo PDF escrito usando LaTeX junto a la tarea (por lo cual tienen la misma fecha de entrega), con nombre `informe.pdf` en la raíz del repositorio. En este se debe explicar al menos los siguientes puntos:

1. Estructuras de datos usadas y sus características (nombre, complejidad de operaciones relevantes).
2. Como se implementan en memoria las estructuras usadas.
3. Complejidad de las operaciones de Heap Market.
4. Consideraciones y estrategias usadas en los árboles de búsqueda.
5. Otras aplicaciones de las estructuras de datos usadas (donde se suelen usar, motivación).

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 2 planas (excluyendo la página de referencias), informes ilegibles, o generados con inteligencia artificial.

Nota: Es importante destacar que este informe no necesita seguir una estructura convencional de responder pregunta por pregunta. Se brinda total libertad en el formato de presentación, pero es crucial que incluyan los puntos mencionados anteriormente para obtener el puntaje completo. La evaluación se centrará en la comprensión y la aplicación efectiva de los conceptos de estructuras de datos y búsqueda eficiente en el contexto de la tarea más que en la conformidad con un formato específico de informe.

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
- 70 % a la nota entre las partes 1 y 2, donde se evalúa por % de tests pasados.
  - 25 % a la nota de los tests de la parte 1
  - 45 % a la nota de los tests de la parte 2
- 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[3]. Ademas, en esta tarea cada sección indica algún tipo de complejidad algorítmica la cual deberas respetar. De lo contrario, recibirás 0 puntos en ese test.

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 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. Estos son consejos de tus ayudantes que te pueden ayudar a pasar el ramo.

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.

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