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

IIC2133 — Estructuras de Datos y Algoritmos
2023 - 2

Tarea 2

Fecha de entrega código: 30 de octubre del 2023 a las 23:59
Link a generador de repos: https://classroom.github.com/a/mqF_eCSc

---

Objetivos

- Identificar y emplear algoritmos apropiados para resolver los problemas dados.
- Utilizar algoritmos y estrategias importantes en la ciencia de la computación.
- Familiarizarse con la importancia de la complejidad computacional.

Figura 1: Shrek firmando para no tener que usar C (no aprende EDD y termina mal)

Parte 1: Un mundo alternativo

Dado el efecto dominó causado por el contrato firmado, Rumpelstiltskin pudo transformar Muy Muy Lejano en su reino a su gusto. Poder controlar todo es el sueño de muchos, pero rápidamente se fue dando cuenta de que crear cada detalle de su reino es muy abrumador: necesitará más magia para poder generar todo lo que falta crear.

Figura 2: Castillo de Rumpelstiltskin, rodeado de tierra decierta

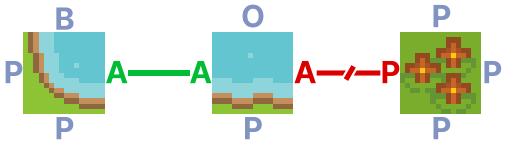
Él sabe que eres muy habilidoso con modelos mágicos de generación para resolver problemas, así que te pide que crees un proceso mágico que permita ir completando rápidamente las partes faltantes del mundo. Te da una lista de posibles opciones que deberás usar, y referencias que deberás tomar, inspiradas de una vuelta atrás al mundo original de Muy Muy Lejano.

Problema

Dado un mapa de tamaño dado, un set de *tiles* a usar, y *tiles* iniciales en el mapa, deberás rellenar todos los cuadrantes con *tiles* que calcen con sus vecinos.

Tiles

Tile (o tesela) es una sección cuadrada que se usa para tileado de mapas (como en juegos o mapas web). Para el ejercicio, no nos preocuparemos de la sprites, pero cada *tile* deberá tener vecinos con sentido:



Cada *tile* contiene 4 tipos de bordes (arriba, derecha, abajo, izquierda). Para que 2 *tiles* puedan estar juntas, sus bordes deben tener el mismo tipo. Se usará letras A-Z (en inglés, sin Ñ) para representar los tipos.

Tileset y Mapa

Para la generación del mapa, se te entregará la siguiente información:

- **Ancho** y **Alto** del mapa a tildear.
- El **tileset** (lista de *tiles* a usar) a utilizar. Se entregará inicialmente un número `n`, y por cada una de las siguientes `n` líneas, las 4 letras de los tipos de bordes (arriba, derecha, abajo, izquierda).
- **Tiles iniciales**, en el formato columna, fila e índice del *tile* correspondiente.

Código Base y Ejecución

Tu programa debe compilar con `make` para generar el ejecutable `worldGen`, que se use de la siguiente manera:

```
./worldGen input.txt output.txt
```

El código base posee solo un archivo `main.c` encargado de manejar la lectura inicial del archivo. El resto de la lógica debe ser creado por ustedes.

Ejemplo Input

```
4 3
6
A C A C
A C C C
C A C C
C C A C
C C C A
A A A A
2
1 1 2
3 0 5
```

Se entrega un mapa de 4 de ancho y 3 de alto. Luego se entregan 6 tiles que podrán ser utilizadas para rellenar el mapa, y luego 2 tiles en el formato (columna, fila, índice).

Ejemplo Output

```
0 1 2 5
0 2 4 0
0 2 4 0
```

Se ve que la posición $(1, 1)$ y $(3, 0)$ tiene el tile 2 y 5 respectivamente. Una versión alternativa de **ver** el output pedido es añadiendo los bordes correspondientes.

```
 A  A  C  A
C0CC1CC2AA5A
 A  C  C  A
 A  C  C  A
C0CC2AA4CC0C
 A  C  C  A
 A  C  C  A
C0CC2AA4CC0C
 A  C  C  A
```

Parte 2: Conteo de escuadrones

Hoy, con el nuevo reino, Rumpelstiltskin se ha colapsado con tantas tareas, que ni con la ayuda de FiFi ha logrado cachar bien como hacerlas todas. Tratando de ocultar su desesperación, fue a la oficina de la líder de las brujas a pedir ayuda a último minuto, especialmente con el conteo de la gran cantidad de escuadrones y pandillas que él quería controlar.

Figura 3: Rumpelstiltskin en su trono junto a brujas, fingiendo que tiene todo al día

Afortunadamente, las brujas tenían un As bajo la manga para este último desafío: un hechizo que les permitiría en un chasquido registrar las apariciones de los diferentes escuadrones. Las brujas te ven desocupado, así que te pasan ocultado el truco en estos párrafos, para que puedas realizar tú mágicamente las cuentas.

Problema

Se te entregará una cantidad indefinida de operaciones a utilizar para llevar un conteo de escuadrones, añadiendo o consultando la cantidad de registros de un escuadrón, hasta que se cierre el programa.

Escuadrones

Haciendo referencia a los escuadrones, son una lista de diferentes tipos de soldados, **representados con letras A-Z** (en inglés, sin Ñ). Los escuadrones se entregan de la siguiente forma:

```
3 S B G
```

El primer carácter representa la cantidad de tipos de soldados, y el resto serán los tipos. Los escuadrones **pueden estar desordenados** (no importa el orden), por lo que los siguientes escuadrones son equivalentes:

```
3 B G S
3 S G B
```

Operaciones

Añadir (+) Aumenta en 1 la cantidad de veces que se ha registrado el escuadrón dado.

Ejemplo: `+ 4 D C B A`
Esto significa que se añade el escuadrón `4 A B C D`

Consultar (`?`) Deberá imprimir la cantidad de veces que se ha registrado el escuadrón dado.

Ejemplo: `? 4 B A D C`
Esto significa que se consulta por el escuadrón `4 A B C D` y se espera que el output sea el número de ocurrencias.

Salir (`q`) Termina el programa.

Código Base y Ejecución

Tu programa debe compilar con `make` para generar el ejecutable `squads`, que se use de la siguiente manera:

```
./squads input.txt output.txt
```

El código base posee solo un archivo `main.c` encargado de manejar la lectura inicial del archivo. El resto de la lógica debe ser creado por ustedes.

Ejemplo Input

```
+ 3 S B G
+ 2 O O
? 1 O
+ 3 B G S
+ 1 S
? 3 G B S
q
```

Ejemplo Output

```
0
2
```

Se imprime por cada `?`:

- `0` dado que no se registró el escuadrón `1 O`
- `2`, porque `3 G B S` fue registrado como `3 S B G` y `3 B G S` (notar que tienen diferente orden)

Competencia (bonus)

Dado que se busca medir eficiencia, se realizara una competencia premiando así a las 10 mejores tareas (En base a su tiempo total de ejecución) con los siguientes premios:

- 1° Lugar: 5 décimas + cupón de atraso adicional
- 2° y 3° lugar: 5 décimas
- 4°, 5° y 6° lugar: 3 décimas
- 7°, 8°, 9° y 10° lugar: 2 décimas

Importante: para poder clasificar al leaderboard, se debe poseer 100 % de correctitud en los tests
Los primeros tres lugares además aparecerán en un leaderboard público para la posterioridad del curso
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]. De lo contrario, recibirás 0 puntos en ese test.

Documento de diseño

La tarea contempla un documento de diseño que posee ponderación en la nota. En particular, el documento es evaluado de forma binaria. Asignando puntaje completo en caso de realizar las tres secciones, y ningún puntaje en caso contrario.
La idea del documento son varias. Primero es incentivar el análisis del enunciado y del problema en el inicio. Es **fuertemente recomendado realizar gran parte del documento antes de empezar a programar** la solución (ve este link). Por otro lado, la idea es permitir realizar recorrecciones manuales de código. Donde se asigna puntaje parcial a la implementación realizada.

IMPORTANTE: Si no se realiza el documento de diseño, NO se podrá optar a recorrección manual.

Contenidos mínimos

El documento de diseño ha de ser llenado en el template `docs/design.md` ubicado en el repositorio base de la tarea.
No se espera que el documento sea formal, sino que se espera que queden como registros del proceso de diseño de su solución. Es por esto que ha de contener como mínimo las siguientes secciones:

1. Análisis del enunciado: Breve análisis identificando las estructuras y sus relaciones.
2. Planificación de solución: una idea a rasgos generales sobre como será abordado el problema, identificando que Estructura de datos se utilizara.
3. Justificación: Sección donde se justifican las decisiones de diseño consideradas para la implementación de la tarea

Entrega documento de diseño

El documento de diseño se entrega en conjunto con el repositorio de la tarea, el contenido principal ha de ir en el archivo `docs/design.md`. En caso de querer dejar anexos como imágenes o diagramas, han de ser añadidos en la misma carpeta.

[1] Puedes revisarlo con el comando htop o con el servidor

Evaluación y Cálculo de Nota

La tarea contempla dos secciones principales que contribuyen a la nota. La sección práctica, compuesta por resolver correctamente los tests y la parte teórica.

Parte Práctica (80 %)

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[2]. De lo contrario, recibirás 0 puntos en ese test.

- Parte 1: 50 %
- Parte 2: 40 %
- No Leaks de Memoria: 5 %
- No Errores de Memoria: 5 %

En el caso de leaks y errores. Puedes chequear que tu programa no los posea utilizando la herramienta `valgrind`.

Evaluación Tests

Los tests se evaluaran de la siguiente forma

- Parte 1:
  - 1.0 punto, si el output es igual al esperado
  - 0, en otro caso
- Parte 2:
  - 1.0 punto, si el output es igual al esperado
  - 0.8 punto en caso de que al menos 95 % del output sea igual
  - 0.7 punto en caso de que al menos 90 % del output sea igual
  - 0.6 punto en caso de que al menos 85 % del output sea igual
  - 0, en otro caso

Parte Teorica (20 %)

La parte teórica contempla el documento de diseño y el cuestionario.
Documento de diseño (20 %): Su evaluación es binaria. Ósea, obtiene puntaje máximo en caso de realizarlo completo y puntaje mínimo en caso contrario. Solo se evalúa la presencia de las secciones, no es importante que sea detallado.

Cuestionario (80 %): El cuestionario posee un total de 25 puntos obtenibles, de los cuales se consideran 20 para la nota máxima. Su formula es la siguiente:

$$nota = 6 \cdot \frac{min(puntaje, 20)}{20} + 1$$

El enlace al cuestionario se enviará dentro del plazo de la tarea mediante un anuncio en Canvas.

[2]Puedes revisarlo con el comando htopm, en el servidor, o con `valgrind --tool=massif`.

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, veas este link 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 :)

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.

Política de atraso: Para esta tarea **aplica** la política de atraso del curso, definida de la siguiente forma:

- Cada día de entrega atrasada, descontará 7 décimas de la **nota máxima obtenible**. Con un máximo de 3 días
- Todos los estudiantes parten con 2 cupones de atraso. El uso de un cupón *elimina* un día de atraso

En resumen, el código para calcular la nota es el siguiente:

```
double nota_con_atraso(double nota, int dias_de_atraso, int cupones_usados) {
    int atraso_efectivo = dias_de_atraso - cupones_usados;

    if (atraso_efectivo > 3) { return 1.0; }

    return min(nota, 7.0 - 0.7 * atraso_efectivo);
}
```

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.