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

IIC2133 — Estructuras de Datos y Algoritmos
2022 - 2

Tarea 2

Fecha de entrega código: 21 de Octubre
Link a generador de repos: Ir a github

Objetivos

- Aplicar técnicas de hashing para optimizar consultas sobre conjuntos de datos
- Analizar y diseñar funciones de hash considerando criterios sobre la calidad y velocidad de la misma.

Introducción: Árboles del Riddler

Últimamente, Batman ha estado luchando contra el Riddler (o Acertijo en español), y terminó por encontrar su guarida. Sin embargo, la guarida está llena de árboles, por lo que es muy difícil para Batman encontrar evidencia. Nuevamente, por su trauma con los árboles (que le recuerdan a su árbol familiar), te ha solicitado ayudarlo para evitar seguir gastando sus bati-millones en bati-psicólogos.

Batman ha visto una foto de como luce el árbol favorito del Riddler y te la ha entregado codificada para que tú sepas que hay que buscar. Para ayudar a Batman debes encontrar todos los sub-árboles que coincidan con el árbol favorito del Riddler, pues en esos es más probable que el riddler haya guardado objetos sensibles que le puedan servir a Batman como evidencia.

En particular, los árboles tienen nodos negros y blancos, los cuales permiten distinguir un árbol de otro.

Figura 1: Una foto del Riddler junto a su árbol favorito.

Problema: Búsqueda del árbol del Riddler

Todos los árboles fueron almacenados como un árbol binario completo. Es decir, un árbol binario donde, dada su altura h, tendrá $n = 2^h - 1$ nodos. Denotaremos este árbol por Γ.

Lo particular del árbol Γ, es que solo almacenan un valor binario 1 o 0. Donde 0 indica que el nodo del árbol es negro y 1 si es que es blanco.

Para facilitar la visualización de ejemplos, diremos que un nodo con valor 0 corresponde a un nodo pintado y 1 corresponde a un nodo no pintado.

Por lo tanto, el siguiente árbol es un ejemplo de Γ.

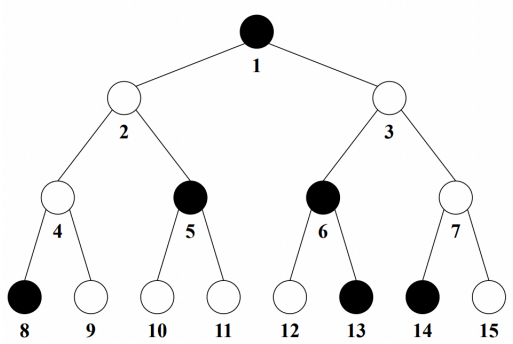

Figura 2: Γ con $h = 4$. Los nodos están numerados de 1 a n. Llamaremos a este número *el ID del nodo.*

Ahora, dado Γ con altura h_Γ. El problema a resolver será encontrar las ocurrencias de distintos árboles $\gamma_1, \gamma_2, \ldots, \gamma_k$ en Γ. Considerando qué $1 \leq h_i \leq h_\gamma \ \forall i \in \{1, \ldots, k\}$.
Definiremos como *ocurrencia* de γ_i sobre Γ a algún subárbol de Γ de altura h_i, cuyos nodos coinciden en color con los de γ. Por ejemplo

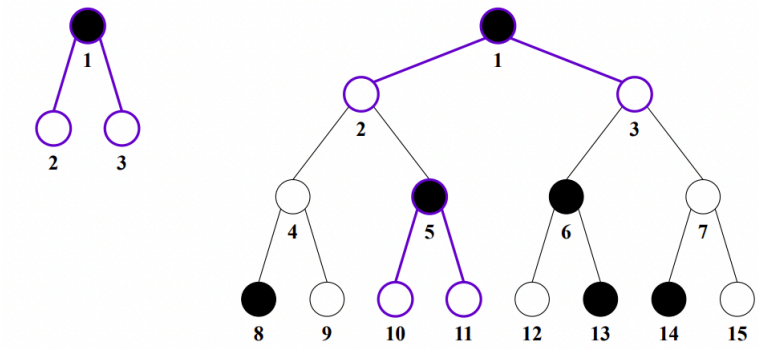

Figura 3: A la izquierda el árbol γ a buscar, con $h_\gamma = 2$. A la derecha, sus *ocurrencias* en Γ

Luego formalizando. Denotamos las ocurrencias en Γ de un subárbol γ por el *ID del nodo* de Γ que coincide con la raíz de γ. En el ejemplo anterior, las ocurrencias de γ en Γ serían entonces los ID 1 y 5. Además, es posible que las ocurrencias de γ se traslapen, como muestra el siguiente ejemplo:

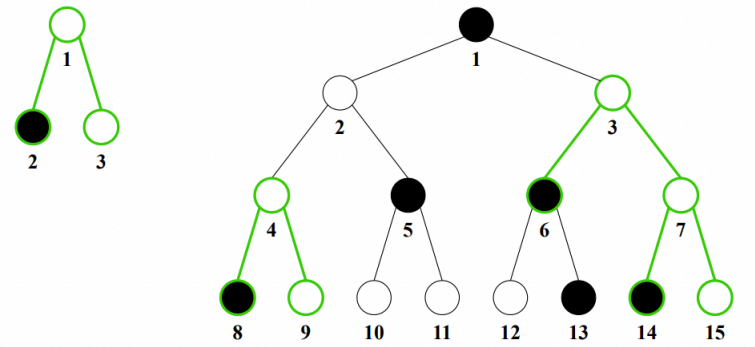

Figura 4: Las ocurrencias de γ en Γ serian los ID 3, 4 y 7. Notar que 3 y 7 se traslapan

Además, un árbol γ puede **no** aparecer en Γ. Por ejemplo:

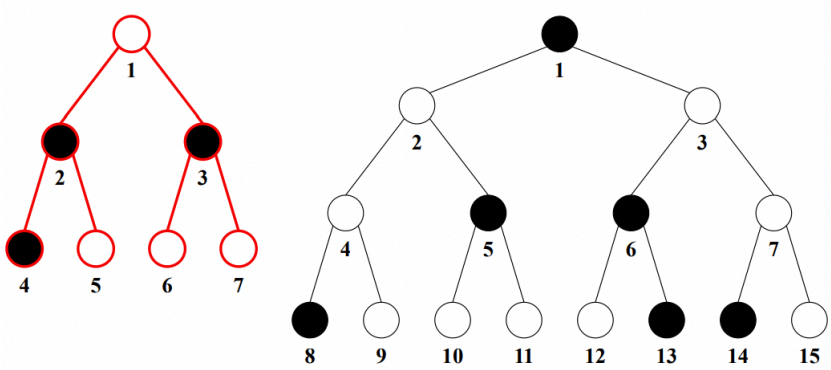

Figura 5: En este caso diremos que la ocurrencia es el ID -1, ya que no existe

Finalmente, tu tarea será escribir un programa en el lenguaje C que, dada una descripción de Γ, pueda encontrar las ocurrencias de K árboles γ. Para esto **deberás** utilizar tablas y funciones de hash. Donde es tu responsabilidad diseñar la función adecuada al problema que pueda codificar de forma eficiente los datos entregados. Se recomienda fuertemente la utilización de una función de hash incremental[1]

También tendrás que implementar una *Hash Table* apropiada, analizando los parámetros asociados a esta con el fin de disminuir la memoria utilizada y el tiempo de búsqueda. tomando en consideración los criterios de factor de carga, tipo de direccionamiento, entre otros.

[1]Significa que a partir del hash de N, sea *rapido* calcular el de N+1

Ejecución

Tu programa se debe poder compilar con el comando `make` y generar un ejecutable de nombre `riddler`. Que se debe ejecutar con el siguiente comando

```
./riddler input.txt output.txt
```

Donde `input` es el archivo inicial con el árbol original y una serie de consultas. Y `output` es la ruta al archivo de output. En esta tarea, el código base solo contendrá un archivo `Makefile` y un archivo `main.c` con lo básico de C, pero que no contendrá lectura de input.

Input

El input se estructura de la siguiente forma

- Una línea que describe el árbol Γ. Esto es, un número n que indica la cantidad de nodos. Seguido por el color de los n nodos, ordenando por índice. Negro = 0, Blanco = 1
- Una línea indicando el número K de consultas
- K líneas que contienen un árbol γ_i a consultar. Este árbol se describe de la misma forma que Γ en la primera línea

Por ejemplo, para el input Γ de ejemplo, y las 3 consultas mostradas:

```
15 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1
3
3 0 1 1
3 1 0 1
7 1 0 0 0 1 1 1
```

La primera línea indica que es un árbol de 15 nodos y luego la descripción del árbol. Luego se indica el número de consultas. Finalmente cada una de las consultas. Se subirán tests adicionales a los del repositorio base durante los siguientes días. Podrán encontrar el link directo desde el repositorio del curso

Output

El output debe consistir de K líneas, cada una con los ID de las ocurrencias del árbol γ correspondiente en el input. Estos ID se imprimen ordenados de menor a mayor. Por ejemplo, para el input anterior de 3 consultas, el resultado es el siguiente:

```
1 5
3 4 7
-1
```

Gotta go fast (Competencia)

Para darles la oportunidad de obtener décimas adicionales, vamos a realizar una competencia. A los mejores diez alumnos, es decir con los tiempos menores, se darán 8 décimas al primer lugar, 5 décimas al segundo, tercer y cuarto lugar y al resto se darán dos décimas de bonus.

Evaluación

La nota de la tarea será evaluada por tests automatizados, donde el puntaje de cada test individual depende del porcentaje de consultas calculadas de forma correcta.
Una consulta correcta corresponde a que se reportaron correctamente **todos** los ID's respuesta de la misma. Luego, en un test donde se obtuvo X % de las consultas correctas, el puntaje es dado por:

- 0 si $X < 75\,\%$
- 0.5 si $75 \leq X < 85$
- 0.7 si $85 \leq X < 95$
- 0.8 si $95 \leq X < 1$
- 1 si $X = 1$

Finalmente, la nota de la tarea se desglosa como:

- 90 % por el resultado de los tests.
- 5 % por no poseer leaks de memoria.
- 5 % por no poseer errores de memoria.

Importante: 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.

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 trabajen solicitando y liberando memoria manualmente. Para evaluar esto, usaremos *valgrind*. Se recomienda fuertemente ver los videos de este repositorio.

Para asegurarte que no tengas errores de memoria debes correr tu programa con:
`valgrind ./riddler input.txt output.txt`

y el output debe contener
"`All heap blocks were freed -- no leaks are possible`" y
"`ERROR SUMMARY: 0 errors from 0 contexts`"

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

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: La política de atraso del curso considera dos criterios:

- Utilización de **cupones**[3] de atraso. Donde un cupón te proporciona 24 horas habiles adicionales para entregar tu tarea sin penalización a tu nota
- Entrega atrasada sin cupón donde se aplica un descuento *suave*[4]. Calculada de la forma

$$N_f = min(70 - 0{,}7 \cdot d^{1{,}3}, N_o)$$

  Donde d es la cantidad de días atrasados, N_f la nota final y N_o la nota obtenida en la tarea

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] Recuerda que solo tienes 2 cupones para todo el semestre
[4] Es un descuento a la nota máxima posible