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

Tarea 1

IIC2133 - Estructuras de Datos y Algoritmos

Segundo semestre, 2016

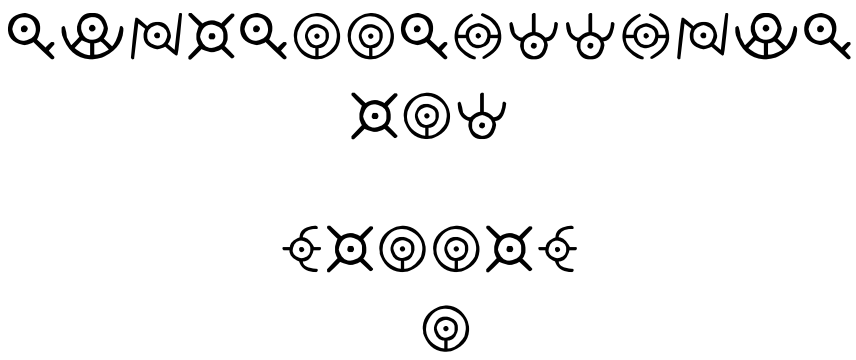
Entrega: Jueves 8 de Septiembre

Objetivos

- Modelar un problema de planificación con backtracking y llevarlo a la práctica
- Construir y hacer uso de estructuras de datos para optimizar un algoritmo básico
- Analizar el comportamiento de un problema exponencial

Introducción

A lo largo del archipiélago de Aloha, en medio del mar, se encuentran diversas rocas con inscripciones misteriosas, como las que siguen:



Durante mucho tiempo no se pudo dilucidar su significado, pero este último año se encontraron las notas del desaparecido explorador Steven Stone, las cuales explican el propósito de las inscripciones. Según Stone, se trata de un juego de los antiguos, donde la solución permitiría encontrar las ruinas perdidas de Pok**edd**o.

Problema

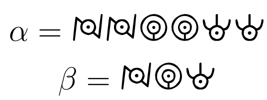
Cada inscripción sigue la misma estructura: una secuencia de símbolos α, y debajo de esta, otra secuencia de símbolos β. El juego consiste en extraer el **primer** elemento de la secuencia β e insertarlo en la secuencia α, siguiendo las siguientes reglas:

1. Si la inserción **formó** un grupo de **3 o más** símbolos iguales adyacentes, **ese** grupo se elimina y se sigue al paso 2. Si no, el símbolo queda ahí y pasa a formar parte de la secuencia α.
2. Las dos cadenas que quedan cierran el espacio que dejó el grupo anterior. Si esto **forma** un grupo de **2 o más** elementos adyacentes del mismo símbolo, **ese** grupo se elimina y se repite el paso 2. Si no, termina el proceso de inserción.

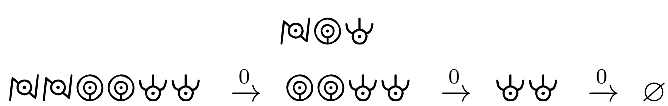
Esto debe repetirse hasta que la secuencia α quede completamente vacía, sin importar si sobran elementos en la secuencia β. Dado que siempre se usa el primer elemento de β, solo importa en qué posición de α se inserta. Esta secuencia ordenada de índices es la solución al problema. Usaremos el siguiente formato para representar la resolución de un problema:

$$\begin{array}{c}\beta \\ \alpha_0 \xrightarrow{x_1} \alpha_1 \xrightarrow{x_2} \alpha_2 \xrightarrow{x_3} \dots \xrightarrow{x_n} \varnothing\end{array}$$

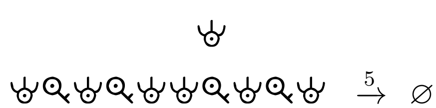
Donde α_i es el i-ésimo estado de α, y x_i es el índice de la i-ésima inserción, y β es β. A continuación se muestran algunos ejemplos:

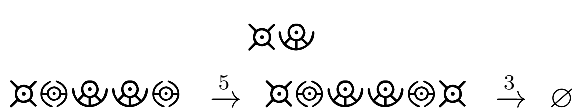


Extrayendo el primer símbolo de β, e insertándolo en la posición 0, se obtendría la secuencia  , por la regla 1) quedaría en  . Se repite lo mismo con los siguientes elementos de β:



Otros ejemplos dados distintos α y β:





Debes escribir un programa en C que, dado dos secuencias de símbolos α y β, busque la secuencia de inserciones ordenadas de los elementos de β dentro de la secuencia α de manera de que la secuencia α quede completamente vacía. Pueden sobrar elementos de β.

Input

Tu programa deberá recibir los siguientes parámetros:

1. La ruta del archivo que contiene las secuencias α y β
2. La ruta del archivo donde deberás imprimir tu respuesta

de la siguiente manera:

```
./solve puzzle.txt solucion.txt
```

El archivo de input consiste en dos lineas: la primera contiene la secuencia α y la segunda la secuencia β

Output

Tu archivo de output deberá tener $n+1$ líneas, donde la primera línea corresponde al número de inserciones hechas y las siguientes n líneas indican las inserciones x_i de tu solución en orden.

Por ejemplo, si el input fuera

```
ABCCB
AC
```

Un output correcto sería

```
2
5
3
```

Informe

Deberás entregar un informe donde analices el problema. Se espera lo siguiente:

- (1pto) De qué forma se puede usar backtracking para este problema.
- (2.5pts) Analizar teóricamente como cambia el desempeño de backtracking al representar las secuencias como arreglos
- (2.5pts) Proponer por lo menos una poda que se pueda aplicar al problema, y justificar su beneficio.

Evaluación

La nota de tu tarea está descompuesta en distintas partes:

- 70 % a que las respuestas generadas sean correctas. Se probarán distintos casos de prueba.
- 30 % a tu informe.

Los test tendrán un tiempo límite de 10 segundos c/u. Si tu programa no ha entregado en ese tiempo será cortado y tendrás 0 puntos en ese test.

Entrega

Deberás entregar tu tarea en el repositorio que se te será asignado; asegurate de seguir la estructura inicial de éste.

Se espera que tu código compile con `make` dentro de la carpeta **Programa** y genere un ejecutable de nombre `solve` en esa misma carpeta

Se espera que dentro de la carpeta **Informe** entregues tu informe, con el nombre *Informe.pdf*

Por cada regla que no cumplas se te aplicará un descuento porcentual a tu nota final.

Se recogerá el estado de la rama `master` de tu repositorio, 1 minuto pasadas las 24 horas del día de entrega. Recuerda dejar ahí la versión final de tu tarea. No se permitirá entregar tareas atrasadas.

Bonus

A continuación, formas de aumentar la nota obtenida en tu tarea. Estos aplicarán solo si la nota involucrada es mayor o igual a 4.

Crazy Fast (+2.5 a la nota de *Informe*)

Propón una heurística para decidir a donde hacer las inserciones y justifica por qué es un aporte. Esta no debe perjudicar notoriamente a algún tipo de instancia del problema.

Buen uso del espacio y del formato (+5 % a la nota de *Informe*)

La nota de tu informe aumentará en un 5 % si tu informe está bien presentado y usa el espacio y formato a favor de entregar la información. A juicio del corrector.

Anexo

Como equipo de ayudantes decidimos incluir un par de consejos para poder abordar el problema. Lo principal es que modeles cuidadosamente tu algoritmo antes de ponerte a escribir código. Para ayudar a esto, trata de responder a las siguientes preguntas:

- ¿Donde calza backtracking en todo esto?
- ¿Cuál sería la condición de término? ¿Habrá una forma O(1) de revisarla?
- ¿Qué decisión se debe tomar en cada iteración?
- ¿Qué implica tomar una decisión? ¿Existen distintos casos?
- ¿Como hacer para deshacerla? ¿Existen distintos casos?
- ¿Qué podas y/o heurísticas se podrían aplicar? ¿Es barato revisarlas?
- ¿Existirá alguna forma de representar el problema de manera de que las inserciones y eliminaciones dentro de la secuencia no resulten tan costosas? Hint: Lista Doblemente Ligada

Asegurate de leerte la guia de backtracking subida en el repositorio del curso. Este problema calificaría como un problema de "planificación" para lo ahí descrito.

Y recuerda: