INF-221 Algoritmos y Complejidad, 2022-1

Tarea 1

Profesor: Diego Arroyuelo
Ayudantes: Cristian Jara
cristian.jaras@sansano.usm.cl

Fecha de Inicio: 2 de mayo, 2022
Fecha de Entrega: 25 de mayo, 2022
Plazo máximo de entrega: 5 días.

Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con el ayudante **ANTES** de comenzar la tarea. No se permiten de ninguna manera grupos de más de 3 personas. Pueden usarse los lenguajes de programación C, C++, Python, y Java.

1. Problema 1: Terrenos Inundados

Una compañía constructora ha comprado una gran área de terreno en el sur del país. Dadas las condiciones meteorológicas y geográficas de la zona, el terreno posee lagunas. Aunque en principio puede parecer una desventaja, el poseer lagunas internas es un beneficio para el paisaje, y por lo tanto el valor de las propiedades que se construyan será más elevado.

Los ingenieros de la compañía dividieron el terreno en una grilla (o matriz) conformada por celdas de tamaño uniforme. Cada celda contiene ya sea agua o tierra firme, pero no ambas. La pregunta que los ingenieros quieren responder es: dada una celda que contiene agua (identificada por su número de fila y columna en la grilla), ¿Cuál es el área de la laguna que contiene a dicha celda? En este caso, el área de una laguna se calcula como la cantidad de celdas que la conforman. Las celdas diagonales son consideradas adyacentes. Su solución debe ser lo más eficiente posible. Diseñe e implemente un **algoritmo de fuerza bruta recursiva** para resolver este problema.

Formato de Entrada

Los datos serán leídos desde la entrada standard, en donde cada línea tendrá el siguiente formato. La primera línea contiene un único entero positivo, indicando el número de casos de prueba que le siguen. Luego de la primera línea hay una línea en blanco. Además, hay una línea en blanco después de cada caso de prueba. Cada caso de prueba consiste de $0 < n \leq 999$ líneas, cada una conteniendo una secuencia de caracteres '`L`' (tierra firme) y '`W`' (agua), de largo $0 < m \leq 999$ cada una. Estas n líneas representan a la grilla de $n \times m$. Luego le siguen $k > 0$ líneas, cada una de las cuales contiene un par de enteros i y j, que representan a una celda de la grilla que contiene agua, y serán usados para hacer las consultas sobre la matriz. Un ejemplo particular de entrada es el siguiente:

```
2

LLWLLLLLLLLLLLLWLLL
WLLLLLLLWLLLLWWLWLLL
LLWWLLLLWWLWLWLLLLWL
LWWLLLWWLWLLLLLWLLLL
4 2
2 1
1 3
2 9

WLL
LLL
LWL
WLW
LWW
LLL
1 1
3 2
```

Hint: para probar su programa de una mejor manera, ingrese los datos de entrada con el formato indicado en un archivo de texto (por ejemplo, el archivo `input-1.dat`). Luego, ejecute su programa desde la terminal, redirigiendo la entrada estándar como a continuación:

```
./problema1 < input-1.dat
```

De esta manera, evitará tener que entrar los datos manualmente cada vez que prueba su programa.

Formato de Salida

La salida del programa debe mostrarse a través de la salida standard. Para cada caso de prueba, la salida debe seguir la siguiente descripción. Las salidas de dos casos de prueba consecutivos deben ser separados por una línea en blanco.

Para cada par de enteros i y j presentados como entrada, se debe imprimir una línea que indique el área de la laguna que contiene a la celda con fila i y columna j de la grilla.

```
4
1
1
6

1
5
```

2. Problema 2: Encontrando Diferencias entre dos Secuencias

Se quiere implementar una herramienta que permita mostrar todas las diferencias que existen entre dos secuencias de texto. En particular, dadas dos secuencias de caracteres $s[1..n]$ y $t[1..m]$, se deben mostrar pares de substrings (uno de cada secuencia) que difieren. En particular, para que esto sea útil para el usuario, se debe mostrar **la mínima cantidad de pares de substrings** que indiquen todas las diferencias entre s y t. Resuelva este problema de manera eficiente definiendo un algoritmo de programación dinámica.

Hint: para definir los pares de substrings a mostrar, se debe encontrar una serie de caracteres que aparezcan en ambas secuencias de entrada (s y t). Esos caracteres particionan a s y t en substrings correspondientes que difieren, los cuales deben mostrarse. El problema es encontrar dichos caracteres de tal manera de minimizar la cantidad de pares a mostrar.

Formato de Entrada

La entrada de datos será por la entrada estándar (`stdin`), y contendrá varios casos de prueba. La primera línea de la entrada contiene un entero K indicando la cantidad de casos a probar ($1 \leq K \leq 10{,}000{,}000$). Luego, le siguen los K casos, cada uno con el siguiente formato. La primera línea comienza con un entero n, y es seguido por una secuencia s de n caracteres. Ambos datos están separados por un único espacio en la entrada. Puede asumir $1 \leq n \leq 1{,}000{,}000$. La segunda línea del caso comienza con un entero m, y es seguido por una secuencia t de m caracteres. Ambos datos están separados por un único espacio en la entrada. Puede asumir $1 \leq m \leq 1{,}000{,}000$.

Un ejemplo de entrada es el siguiente:

```
3
6 ABCLGH
6 AELFHR
6 AGGJAB
7 GZJZAMB
16 Este es un texto
18 Este es otro texto
```

Formato de Salida

La salida se hará por la salida estándar (`stdout`). La salida comienza con el valor K, indicando la cantidad de casos de prueba (es el mismo valor K de la entrada). Luego, para cada uno de los casos de prueba de la entrada, deben imprimirse todos los pares de substrings que difieren en las secuencias de entrada. El formato a usar es el siguiente. Si la cantidad de pares a mostrar para un caso es L, se debe imprimir el valor L, y a continuación se deben mostrar L líneas, cada una conteniendo el correspondiente par de strings, separados por un único espacio. Sólo se deben mostrar los substrings que difieren en ambas secuencias. Los substrings mostrados pueden ser vacíos, aunque sólo tiene sentido mostrar los pares en los que al menos unos de los substrings no es vacío.

La salida correspondiente al ejemplo mostrado anteriormente es:

```
3
3
BC E
G F
 R
4
A
G Z
 Z
 M
1
un otro
```

Note que en la tercera línea del primer caso, el substring correspondiente a la primera secuencia es vacío. Algo similar ocurre en la primera línea del segundo caso, en donde el substring correspondiente a la segunda secuencia es vacía.

3. Entrega de la Tarea

La entrega de la tarea debe realizarse enviando un archivo comprimido llamado

`tarea1-apellido1-apellido2-apellido3.tar.gz`

(reemplazando sus apellidos según corresponda), o alternativamente usando formato zip, en el sitio Aula USM del curso, a más tardar el día 25 de mayo, 2022, a las 23:59:00 hrs (Chile Continental), el cual contenga:

- Los archivos con el código fuente necesarios para el funcionamiento de la tarea.
- `NOMBRES.txt`, Nombre y ROL de cada integrante del grupo.
- `README.txt`, Instrucciones de compilación en caso de ser necesarias.
- `Makefile`, Instrucciones para compilación automática.