Sopa de letras

Objetivos.

- Aprender a implementar el TAD Trie.
- Aprender a usar un Trie para resolver una sopa de letras.
- Aprender a usar el tipo std::map
- Aprender a usar el tipo std::pair
- Aprender a usar el tipo std::stack

Descripción.

Un Trie [1] es un árbol multicamino especializado en recuperar las claves almacenadas basándose en prefijos. En un Trie todo nodo (salvo la raíz) representa un prefijo de clave formado por el camino desde la raíz a dicho nodo y el subárbol que tiene a dicho nodo como raíz almacena todas las clave que tienen como prefijo el definido por el nodo raíz.

Vamos a sacar partido de esta cualidad para resolver de forma eficiente una sopa de letras.

Como ya sabes, una sopa de letras es un tipo de puzle formado por una cuadrícula rellena con letras, de forma que, un número de palabras conocidas están contenidas en la cuadrícula siguiendo una secuencia lineal de casillas y, el resto de casillas se rellenan con letras de forma aleatoria. El juego consiste en localizar las palabras en el menor tiempo posible. La Figura 1 muestra un ejemplo de sopa de letras.

SOPA DE LETRAS DE PAÍSES DEL MUNDO

h	o	x	k	n	i	a	i	c	e	r	g
g	a	c	m	a	e	l	i	h	c	j	z
a	s	d	a	r	c	u	j	p	p	f	a
i	u	p	r	g	r	i	y	a	o	r	i
l	e	o	r	e	a	b	a	n	l	a	s
a	c	r	u	n	i	d	n	d	o	n	u
r	i	t	e	t	n	b	i	o	n	c	r
t	a	u	c	i	a	p	h	r	i	i	x
s	r	g	o	n	m	l	c	r	a	a	h
u	r	a	s	a	e	c	u	a	d	o	r
a	b	l	c	g	l	i	c	u	b	a	a
v	e	c	a	n	a	d	a	i	n	z	s

Francia
Chile
Ecuador
Portugal
Marruecos
Alemania
Australia
Argentina
Canada
Suecia
Cuba
China
Rusia
Polonia
Andorra
Grecia

(*obtenidad de: https://pasatiemposimprimibles.blogspot.com/2019/05/sopa-de-letras-de-nombres-de-paises-del.html)

Figura 1. Un ejemplo de sopa de letras. Se trata de localizar todas las naciones listadas. en la cuadrícula en el menor tiempo posible.

Para resolver la sopa de letras, lo primero que tenemos que hacer es crear un Trie insertando la lista de palabras a buscar como claves del Trie.

Lo segundo sería ir recorriendo la cuadrícula (por filas y columnas), consultando en el Trie si la letra de dicha cuadrícula es prefijo de alguna de alguna clave. Si es así aplicamos el algoritmo de la Figura 2 que escanea de forma recursiva la tabla siguiendo un prefijo de clave.

```
Algorithm scanCell(row:Int, col:Int, dx:Int, dy:Int, TrieNode node,
soup:AlphabetSoup, result:Pair[String, Stack[Pair[Int,Int]]])

   1. If this node is a leaf node (Leaf nodes has value != "").
   1.1 Save the word in the first item of the scan_result pair and finish
recursion.
   2. Else, we can have tree cases.
   2.1.1 This is the first letter of word (dx==dy==0). Scan for all the 3x3
neighbourhood for the next letter.
   2.2.1 We've have a prefix, so we need scan the next letter in the
direction (dy, dx) if this letter is child of this node (recursion). Else
return not found result.first = "")
   2.3 if a word was found (first item of the scan_result pair != ""), we
push the current
         cell's coordinates [row,col] into the second item of scan_result.
```

Figura 2. Algoritmo para escanear una celda (row, col) siguiendo la dirección (dy, dx) a partir del nodo. El resultado se almacena en un par con el primer valor la palabra encontrada y la segunda una pila con las coordenadas de las celdas donde está la palabra.

Así si utilizamos la sopa de la Figura 1, cuando estudiemos la celda (9,5) con valor 'e', encontraremos que es un prefijo de clave ('ecuador') y pasaremos a llamar a nuestro algoritmo de escaneo de celda dy=0, dx=0 y nodo trie.root().child(soup[9][5]). De esta forma se procederá a escanear ahora para el rango rows=(8,9,5) x cols=(4,5,6). En este proceso la celda (9,6) con valor 'c' es un hijo del nodo para el prefijo 'e', por lo que se aplica de forma recursiva el algoritmo sobre este nodo (prefijo 'ec') en la celda (9,6) con dy=(9-9)=0 y dx=(6-5)=1. Obseva como a partir de la primera letra encontrada, (dy,dx) indican ya la dirección a seguir para el resto de la palabra, es decir, ahora en este nodo (prefijo 'ec') ya sólo se estudia la celda (9+dy=**9**, 6+dx=**7**) y se repite el proceso hasta que se llega a un nodo hoja que representa la clave "ecuador".

Detalles de implementación.

Para implementar el tipo TrieNode vamos a utilizar un mapa std::map [2] para representar los enlaces con los nodos hijos, de tal forma que el mapa asociará una letra con un enlace a un nodo hijo.

Para devolver el resultado de un escaneo de una palabra usaremos un par std::pair [3] de una cadena std::string y una pila std::stack [4]. La pila almacena a su vez pares (std::pair) de valores (fila,columna) con las coordenadas de la rejilla para cada letra de la palabra encontrada.

Puntuación de la práctica:

- tests_trie: 3.
- tests_retrieve: 3
- tests_alphabet_soup: 4

Referencias.

[1] Trie: en.wikipedia.org/wiki/Trie

[2] std::map: http://www.cplusplus.com/reference/map/map/

[3] std::pair: https://www.cplusplus.com/reference/utility/pair/pair/

[4] std::stack: http://www.cplusplus.com/reference/stack/stack/