Evaluación Formativa 3
INFO088 - Taller Estructuras de Datos y Algoritmos

Académico: Héctor Ferrada.
Instituto de Informática, Universidad Austral de Chile.
Noviembre 18, 2021

Ejecución: ./problema n k

Para este problema usted dispone de la clase BST_ADN que es un árbol de búsqueda para cadenas, cuya estructura para sus nodos es la siguiente:

```
struct BinaryNode {
    char* word; // clave del nodo
    int m; // largo del arreglo word
    BinaryNode *izq;
    BinaryNode *der;
};
typedef struct BinaryNode nodo;
```

Usted deberá modificar la inserción estudida en clases para el BST con claves enteras. Ahora debe considerar que las claves son arreglos de caracteres provenientes del alfabeto APHABET[0...LEN] (declarado en la cabecera). Se pide:

1. **[1.5 Pts.]** En el **main**, genere n palabras de largo m, cada m debe ser aleatorio entre 1 y k inclusive; y cada caracter debe ser tomado aleatoriamente desde APHABET. Cree el árbol t y por cada palabra generada invoque a su método de inserción. Luego imprima t con el método ***printInorder***(), que ya está codificado. Finalice imprimiendo la cantidad de caracteres que tiene el árbol, para esto invoque al método ***sizeTree***(), que también ya ha sido codificado.

2. **[2.5 Pts]** Cree el método **bool BST_ADN::insert**(nodo **$root$, char *$word$, int m). Tome como referencia el $insert()$ visto en clases de la clase BST y adapte el método para que inserte la clave $word[m]$ en el subárbol de raíz $root$. Recuerde que si la clave está duplicada debe retornar false. Haga uso del método **compareWords**$(w1, m1, w2, m2)$ que compara los arreglos, de tipo char, $w1[m1]$ y $w2[m2]$ retornando -1 si $w1 < w2$, cero si son iguales o 1 si $w2 < w1$.

3. **[2 Pts.]** Cree el método **int BST_ADN::compareWords**(char* $w1$, int $m1$, char* $w2$, int $m2$), que realiza la comparación lexicográfica entre los arreglos de caracteres $w1[m1]$ y $w2[m2]$. Retorna -1 si $w1 < w2$, 0 si son iguales ó 1 si $w1 > w2$. Si una palabra es prefijo de la otra, entonces considere a la más corta como la menor.
   Por ejemplo $ala < alaba$, $casita < caso$, $dinosaurio < oso$, etc.

En la carpeta BST, esta el código fuente del método BST::insert() visto en clases para que se guíe con la codificación de su método.

Nota. Trabaje solo con arreglos de tipo char y no string, tampoco puede utilizar funciones existentes en el lenguaje para el tratamiento de strings o cadena de caracteres. Solo se permite $atoi()$ para recuperar los argumentos del programa.

Ejemplo de ejecución para $n = 10$ y $k = 4$:

```
$ ./problema 10 4
Palabra: [GCTC]  INSERTADA !!
Palabra: [GACC]  INSERTADA !!
Palabra: [TGT]  INSERTADA !!
Palabra: [GAGA]  INSERTADA !!
Palabra: [T]  INSERTADA !!
Palabra: [T] NO se pudo insertar !!
Palabra: [GG]  INSERTADA !!
Palabra: [TTT]  INSERTADA !!
Palabra: [GG] NO se pudo insertar !!
Palabra: [CTC]  INSERTADA !!

 Árbol t =
[CTC]
[GACC]
[GAGA]
[GCTC]
[GG]
[T]
[TGT]
[TTT]

El árbol tiene 24 caracteres en total
### Fin Problema ###
```