

Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Pauta P1 y P4 Control 2 CC3001
Profesor: Patricio Poblete
Fecha: 5 de Julio de 2009

1. Pregunta 1

Para esta pregunta había que hacer una rotación simple hacia la izquierda, tal como aprendieron con los AVL. Recalcular el peso eficientemente consistía en utilizar los pesos ya conocidos de los nodos y sub-árboles ilustrados, de manera que no reimplementaran el método calcularPeso del control 1. Además debían verificar que los sub-árboles ilustrados no fueran nulos como dijo el profesor durante el control, de lo contrario, ejecutar las instrucciones p.der.izq.peso y p.der.der.peso arrojarían una excepción del tipo NullPointerException. Una manera de resolver el problema es la siguiente:

```
public NodoArbol rotarPesar(NodoArbol p){
    //A es p.izq
    //B es p.der.izq
    //C es p.der.der
    //pueden ser nulos!
    int pesoA,pesoB,pesoC;
    //inicializo en 0 los pesos en caso de ser nulos
    pesoA = pesoB = pesoC = 0;
    //actualizo mis variables si los arboles existen
    if(p.izq != null) pesoA = p.izq.peso;
    if(p.der.izq != null) pesoA = p.der.izq.peso;
    if(p.der.izq != null) pesoC = p.der.der.peso;
    //debo crear una referencia para hacer la rotación, escojo a p.der
    nodoArbol d = p.der;
    //hago la rotación
    p.der = d.izq;
    d.izq = p;
    //actualizo los pesos eficientemente (usando A,B y C)
    p.peso = 1 + pesoA + pesoB; //A == p.izq, B == p.der
    d.peso = 1 + p.peso + pesoC; //p == d.izq, C == d.der
    //y retorno el que antes era el hijo derecho
    return d;
}
```

2. Problema 4

Para este problema se les pedía ilustrar los pasos para realizar un par de búsquedas usando KMP en la primera, BMH y BMS en la segunda. Para KMP debían encontrar la función de fracaso, y para el resto debían solamente aplicar el algoritmo correspondiente en la búsqueda.

2.1. Parte (a): KMP

2.1.1. Índices del patrón a buscar

a	b	a	b	b	a	a
1	2	3	4	5	6	7

La función de fracaso consiste en encontrar el tamaño del más largo de los sufijos que sea igual a un prefijo para los sub-strings incrementales del patrón de búsqueda. Por ejemplo, el largo del sufijo más largo que calza con un prefijo cuando seleccionamos los primeros 4 caracteres del patrón de búsqueda consiste

en el string 'ab' pues hay un prefijo al inicio idéntico, de forma que el valor de la función de fracaso para el sub-string de largo 4 del patrón vale 2, el tamaño del sufijo.

2.1.2. Función de fracaso de KMP

j	1	2	3	4	5	6	7
f(j)	0	0	1	2	0	1	1

2.1.3. Algoritmo aplicado

a	b	a	b	a	b	b	a	b	a	b	b	a	a	
1	2	3	4	X										
			f(4)=2	3	4	5	6	X						
							f(6)=1	2	3	4	5	6	7	calce!

Para aplicar el algoritmo de búsqueda hay que notar en qué índice falló el calce de caracteres. Luego de esto se alinea el patrón según el índice que indica la función de fracaso y se sigue comparando a partir de éste en la cadena de caracteres. El proceso se repite sucesivamente hasta llegar al final del texto o hasta encontrar un calce.

2.2. Parte (b): BMH

En BMH basta alinear el carácter de la cadena que causa el error con la primera ocurrencia de la letra en el patrón de búsqueda. Si no existe dicho carácter entonces se mueve completamente el patrón.

2.2.1. Algoritmo aplicado

I	N	G	E	N	I	<u>E</u>	R	I	A	_	Y	C	I	E	N	<u>C</u>	I	A	
C	I	<u>E</u>	N	C	I	A													
			_	C	I	E	N	C	I	A									
											C	I	E	N	<u>C</u>	I	A		
												C	I	E	N	C	I	A	calce!

2.3. Parte (c): BMS

Este algoritmo es muy parecido a BMH, salvo que se compara el carácter que falló del patrón con la letra siguiente de la cadena de caracteres.

2.3.1. Algoritmo aplicado

I	N	G	E	N	I	<u>E</u>	R	I	A		Y	C	I	<u>E</u>	N	C	I	A	
C	I	E	N	C	I	<u>A</u>													
											C	I	E	N	C	I	<u>A</u>		
												C	I	E	N	C	I	A	calce!