Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Ejercicios 2

Estructuras de datos básicas

1. En muchos lenguajes se utilizan estructuras anidadas por paréntesis. Hacer un algoritmo que lee un archivo de texto plano y determine si todas las estructuras anidadas se cierran apropiadamente.
Por ejemplo

```
[[](){})]{([])}
```

es correcta y

```
{[)(]} , {([)}
```

son incorrectas.

2. Idear un método peek() para retornar el elemento en el tope de la pila sin eliminarlo. Proponer su implementación para la pila basada en arreglo y para la basada en lista enlazada.

3. La estructura Bag analizada en clase no tiene forma de eliminar elementos.
- Dar una implementación de la operación remove(Item) que remueve un item de la bolsa en caso de estar presente. Si no existe no hace nada. Asumir una implementación de Bag basada en arreglo.
- Dar una implementación de la operación remove(Item) para cuando la bolsa está implementada por medio de una lista enlazada.

4. Escribir un procedimiento que tome como entrada una lista enlazada y devuelva una nueva lista con los elementos en orden inverso.

5. La lista doblemente enlazada ofrece la posibilidad de recorrer los elementos en ambos sentidos. Proponer una implementación de la estructura PilaCola que implementa al mismo tiempo las funcionalidades de la Pila y de la Cola.

6.* Escribir un procedimiento que tome como entrada una lista enlazada y retorne una nueva lista con los elementos en orden aleatorio.

Análisis de algoritmos

1. Obtener aproximaciones *tilde* para las siguientes expresiones. Indicar el orden de crecimiento de cada una.

a) $N+1/N^2$
b) $1+1/N$
c) $(1+1/N)(1+2/N)$
d) $2N^3-15N^2+N$
e) $lg(2N)/lg(N)$
f) $lg(N^2+1)/lg(N)$
g) $N^{100}/2^N$

2. Determinar la función T(N) que describe el tiempo requerido por los siguientes algoritmos en función de los tiempos requeridos por las operaciones elementales.

a. Obtener el máximo de un vector

```
int max=datos[0];
for(int i=1; i<datos.length; i++)
     if (datos[i]>max) max=datos[i];
return max;
```

b. Contar elementos repetidos en una lista

```
int count=0;
for(Node i=first; i.hasNext(); i=i.next) {
     for(Node j=i.next(); j.hasNext(); j=j.next) {
          if (j!=i && i.item.equals(j.item)) count++;
     }
}
return count;
```

3. Determinar el orden de crecimiento (en función de N) de los siguientes fragmentos de código. Seleccionar el modelo de costo representativo, estimar la frecuencia del modelo de costo y su orden de crecimiento.

a.

```
int sum=0;
for(int n=N; n>0; n/=2)
     for(int i=0; i<n; i++)
          sum++;
```

b.

```
int sum=0;
for(int i=1; i<N; i*=2)
     for(int j=0; j<i; j++)
          sum++;
```

c.

```
int sum=0;
for(int i=1; i<N; i*=2)
     for(int j=0; j<N; j++)
          sum++;
```

d. Multiplicar dos matrices

```
public double[][] product(double[][] a, double[][] b) {
     double[][] c = new double[a.length][b[0].length];
     for(int i=0; i<a.length; i++)
          for(int j=0; j<b[0].length; j++) {
               for(int k=0; k<a[0].length; k++) {
                    c[i][j] += a[i][k]*b[k][j];
               }
          }
     return c;
}
```