UNLP. Facultad de Informática.

Algoritmos y Estructuras de Datos

Cursada 2025 (Redictado)

Práctica 4

Análisis de algoritmos

Ejercicio 1

Debido a un error en la actualización de sus sistemas, el banco AyED perdió la información del estado de todas sus cuentas. Afortunadamente logran recuperar un backup del día anterior y utilizando las transacciones registradas en las últimas 24hrs podrán reconstruir los saldos. Hay poco tiempo que perder, el sistema bancario debe volver a operar lo antes posible.

Las transacciones se encuentran agrupadas en consultas, una consulta cuenta con un valor y un rango de cuentas consecutivas a las que hay que aplicar este cambio, por ejemplo la consulta (333..688 = 120) implica sumar $120 a todas las cuentas entre la número 333 y la número 688 (inclusive). Entonces, la recuperación de los datos consiste en aplicar todas las consultas sobre el estado de las cuentas recuperadas en el backup del día anterior.

El equipo de desarrollo se pone manos a la obra y llega a una solución rápidamente (Algoritmo **procesarMovimientos**). Toman cada consulta y recorren el rango de cuentas aplicando el valor correspondiente, como muestra el siguiente algoritmo.

```
Consultas.comenzar()
While(!consultas.fin()){
        Consulta = consultas.proximo();
        for(i = consulta.desde; i < consulta.hasta; i++){
                cuenta[i] = cuenta[i] + consulta.valor;
        }
}
```

Escriben la solución en pocos minutos y ponen en marcha el proceso de recuperación. Enseguida se dan cuenta que el proceso va a tardar muchas horas en finalizar, son muchas cuentas y muchos movimientos, la solución aunque simple es ineficiente. Luego de discutir varias ideas llegan a una solución (Algoritmo **procesarMovimientosOptimizado**) que logra procesar toda la información en pocos segundos. Ambos algoritmos se encuentran en el archivo Ejercicio 1 - rsq_tn_ayed.zip del material adicional.

a.- Para que usted pueda experimentar el tiempo que demora cada uno de los dos algoritmos en forma empírica, usted debe ejecutar cada uno de ellos, con distintas cantidades de elementos y completar la tabla. Luego haga la gráfica para comparar los tiempos de ambos algoritmos. Tenga en cuenta que el algoritmo posee dos constantes CANTIDAD_CUENTAS y CANTIDAD_CONSULTAS, sin embargo, por simplicidad, ambas toman el mismo valor. Sólo necesita modificar CANTIDAD_CUENTAS .

Nº Cuentas (y consultas)	procesarMovimientos	procesarMovimientosOptimizado
1.000		
10.000		
25.000		
50.000		
100.000		

b.- ¿Por qué **procesarMovimientos** es tan ineficiente? Tenga en cuenta que pueden existir millones de movimientos diarios que abarquen gran parte de las cuentas bancarias.

c.- ¿En qué se diferencia **procesarMovimientosOptimizado**? Observe las operaciones que se realizan para cada consulta.

Aunque los dos algoritmos se encuentran explicados en los comentarios, no es necesario entender su funcionamiento para contestar las preguntas.

Ejercicio 2

La clase BuscadorEnArrayOrdenado del material adicional (Ejercicio 2 - Tiempo.zip) resuelve el problema de buscar un elemento dentro de un array ordenado. El mismo problema, lo resuelve de dos maneras diferentes: búsqueda **lineal** y búsqueda **dicotómica**.

Se define la variable cantidadElementos, la cual se va modificando para determinar una escala (por ejemplo de a 100.000 o 1.000.000, dependiendo de la capacidad de cada equipo). Realice una tabla con el tiempo que tardan en ejecutarse ambos algoritmos, para los distintos valores de la variable. Por ejemplo:

N	Lineal	Dicotómica
100.000		
200.000		
300.000		
400.000		
500.000		
600.000		

Ejercicio 3

En la documentación de la clase arrayList que se encuentra en el siguiente link https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Se encuentran las siguientes afirmaciones

- "The size, isEmpty, get, set, iterator, and listIterator operations run in constant time."
- "All of the other operations run in linear time (roughly speaking)"

Explique con sus palabras por qué cree que algunas operaciones se ejecutan en tiempo constante y otras en tiempo lineal.

Ejercicio 4

Determinar si las siguientes sentencias son verdaderas o falsas, justificando la respuesta utilizando notación Big-Oh.

a. 3^n es de $O(2^n)$
b. $n + \log_2(n)$ es de $O(n)$
c. $n^{1/2} + 10^{20}$ es de $O(n^{1/2})$
d. $\begin{cases} 3n+17, n < 100 \\ 317, n \geq 100 \end{cases}$ tiene orden lineal
e. Mostrar que $p(n)=3n^5 + 8n^4 + 2n + 1$ es $O(n^5)$
f. Si p(n) es un polinomio de grado k, entonces p(n) es $O(n^k)$.

Ejercicio 5

Se necesita generar una permutación random de los n primeros números enteros. Por ejemplo [4,3,1,0,2] es una permutación legal, pero [0,4,1,2,4] no lo es, porque un número está duplicado (el 4) y otro no está (el 3). Presentamos tres algoritmos para solucionar este problema. Asumimos la existencia de un generador de números random, ran_int (i,j) el cual genera en tiempo constante, enteros entre i y j inclusive con igual probabilidad (esto significa que puede retornar el mismo valor más de una vez). También suponemos el mensaje swap() que intercambia dos datos entre sí.

```
public class EjercicioPermutaciones {
	private static Random rand = new Random();

	public static int[] randomUno(int n) {
		int i, x = 0, k;
		int[] a = new int[n];
		for (i = 0; i < n; i++) {
			boolean seguirBuscando = true;
			while (seguirBuscando) {
				x = ran_int(0, n - 1);
				seguirBuscando = false;
				for (k = 0; k < i && !seguirBuscando; k++)
					if (x == a[k])
						seguirBuscando = true;
			}
			a[i] = x;
		}
		return a;
	}

	public static int[] randomDos(int n) {
		int i, x;
		int[] a = new int[n];
		boolean[] used = new boolean[n];
		for (i = 0; i < n; i++) used[i] = false;
		for (i = 0; i < n; i++) {
			x = ran_int(0, n - 1);
			while (used[x]) x = ran_int(0, n - 1);
			a[i] = x;
			used[x] = true;
		}
		return a;
	}
```

```
	public static int[] randomTres(int n) {
		int i;
		int[] a = new int[n];
		for (i = 0; i < n; i++) a[i] = i;
		for (i = 1; i < n; i++) swap(a, i, ran_int(0, i - 1));
		return a;
	}


	private static void swap(int[] a, int i, int j) {
		int aux;
		aux = a[i]; a[i] = a[j]; a[j] = aux;
	}

	/** Genera en tiempo constante, enteros entre i y j con igual probabilidad.
	 */
	private static int ran_int(int a, int b) {
		if (b < a || a < 0 || b < 0) throw new IllegalArgumentException("Parámetros
inválidos");
		return a + (rand.nextInt(b - a + 1));
	}

	public static void main(String[] args) {
		System.out.println(Arrays.toString(randomUno(1000)));
		System.out.println(Arrays.toString(randomDos(1000)));
		System.out.println(Arrays.toString(randomTres(1000)));
	}
}
```

a. Analizar si todos los algoritmos terminan o alguno puede quedar en loop infinito.
b. Describa con palabras la cantidad de operaciones que realizan.

Ejercicio 6

a.- Supongamos que tenemos un algoritmo de $O(\log^2 n)$ y disponemos de 1 hora de uso de CPU. En esa hora, la CPU puede ejecutar el algoritmo con una entrada de tamaño n= 1024 como máximo. ¿Cuál sería el mayor tamaño de entrada que podría ejecutar nuestro algoritmo si disponemos de 4 horas de CPU?

b.- Considerando que un algoritmo requiere T(n) operaciones para resolver un problema y la computadora procesa 10.000 operaciones por segundo. Si $T(n) = n^2$, determine el tiempo en segundos requerido por el algoritmo para resolver un problema de tamaño n=2.000.

Ejercicio 7

Para cada uno de los algoritmos presentados:
a. Expresar en función de **n** el tiempo de ejecución.
b. Establecer el orden de dicha función usando notación Big-Oh.

```
1) for(int i = 0; i < n + 100; ++i) {
     for(int j = 0; j < i * n ; ++j)
         sum = sum + j;
     for(int k = 0; k < n + n + n; ++k)
         c[k] = c[k] + sum;

   }
```

```
2) int i,j;
    int x = 1;
    for (i = 0; i <= n^2; i=i+2)
    for (j = n; j >= 1; j-= n/4)
      x++;
```

```
3) public static void calcular(int n) {
       int i, j, r = 0;
       for ( i = 1; i < n; i= i+2)
          for (j = 1; j <= i; j++ )
                r = r + 1;
       return r;

}
```

Ejercicio 8

a.- ¿Qué hace cada uno de los métodos que aparece a continuación?

b.- Calcule la función de tiempo de ejecución y el orden de la misma para cada método

c.- ¿Cuál de los dos métodos es más eficiente? ¿Por qué?

```
static public int pIter(int x, int n){
       int potencia;
       if  (n == 0)
              p = 1;
       else {
              if  (n == 1)
                     p  =  x;
```

```
                else{
                        p  = x;
                        for (int  i  = 2 ; i <=  n ; i++) {
                                p *= x ;
                        }
                }
        }
        return  p;
}
```

```
static public int pRec( int x, int n){
        if( n == 0 )
                return 1;
        else{
                if( n == 1)
                        return x;
                else{
                        if ( (n  %  2 ) == 0)
                                return  pRec (x * x, n / 2 );
                        else
                                return pRec (x * x, n / 2) * x;
                }
        }
}
```

Ejercicio 9

a. Exprese la función del tiempo de ejecución de cada uno de los siguientes algoritmos, resuélvala y calcule el orden.

b. Compare el tiempo de ejecución del método 'rec2' con el del método 'rec1'.

c. Implemente un algoritmo más eficiente que el del método 'rec3'. (es decir, que el T(n) sea menor).

```
static public int rec1(int n){
        if (n <= 1)
                return 1;
        else
                return (rec1(n-1) + rec1(n-1));
}
```

```
static public int rec2(int n){
        if (n <= 1)
                return 1;
        else
                return (2 * rec2(n-1));
}
```

```
static public int rec3(int n){
```

```
                if  ( n == 0 )
                        return  0;
                else {
                        if ( n == 1 )
                                return 1;
                        else
                                return (rec3(n-2) * rec3(n-2));
                }
        }
```

Ejercicio 10

Calcule el tiempo de ejecución de los métodos buscarLineal y buscarDicotomica de la clase BuscadorEnArrayOrdenado. Compare el tiempo con los valores obtenidos empíricamente en el ejercicio 2.

Ejercicio 11

Calcule el tiempo de ejecución de procesarMovimientos y procesarMovimientosOptimizado del ejercicio 1. Compare el tiempo con los valores obtenidos empíricamente.

Ejercicio 12

Resuelva la función de tiempo de ejecución de los siguientes segmentos de código y calcule el orden.

```
int recur (int n){
if (n == 1)
        return  1;
else
        return(recur(n/2)* recur(n/2));

}
```

```
int recursivo(int n){
if (n <= 5)
        return  1;
else
        return (recursivo (n-5));

}
```

```
int recursivo(int n){
if (n <= 7)
  return  1;
else
  return (recursivo (n/8));

}
```

Ejercicio 13

Considere el siguiente fragmento de código:

```
int count = 0; int n = a.length;
        for (int i = 1; i <= n; i = i*2)   {
                for (int j = 0; j < n; j+=n/2)  {
                 a[j]++;
                }
        }
```

Este algoritmo se ejecuta en una computadora que procesa 1.000 operaciones por segundo. Determine el tiempo **aproximado** (considerando el orden de ejecución del algoritmo) que requerirá el algoritmo para resolver un problema de tamaño n=4096.

Ejercicio 14

Dado el siguiente algoritmo:

```
public static int recursivo( int n ) {

    if (n  <=  1){
        return 1;
    }

    int i  = 1;
    while (i <=  n) {
        i*=2;
    }
   return n  +  recursivo(n/2)  *  n/2;
}
```

Calcular la función de tiempo de ejecución y el orden de ejecución de la misma

Ejercicio 15

Dado el siguiente fragmento de código:

```
int resta = algún valor;

for (int i = n * n; i <= n * n * n; i += n * n)

    for (int j = 1; j <= i; j++)

        resta = resta - j;
```

Calcular la función de tiempo de ejecución y el orden de ejecución de la misma.

Ejercicio 16

¿Cuál es el orden de ejecución del siguiente método? Calcule la función de tiempo de ejecución

```
void fun(int n, int arr[])
{    int i = 0, j = 0;
    for (; i < n; ++i)
        while (j < n && arr[i] < arr[j])
            j++;
}
```

Ejercicio 17

¿Cuál es el orden de ejecución del siguiente fragmento de código? Calcule la función de tiempo de ejecución

```
for (int i = 0; i < n*n ; i ++)
      for (int j = i % n; j >= 0; j--)
            System.out.println("j:"+j);
```

Ejercicio 18

Dado el siguiente algoritmo exprese, desarrolle e indique el orden de la función de tiempo de ejecución T(n):

```
public static void metodo(int n){
      int i = 1;
      int j = n;
      while (i =< n){
            while (j > 1){
              for (int k = i; k < i+10;  k++){
                     System.out.println(k*i*j);
              }
              j = j / 2;
            }
            i = i + 2;
      }
}
```