Evaluación áulica 1 de Estrategias Algorítmicas. Fecha: jueves, 14 de septiembre de 2021.

Nombre del alumno: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Aciertos: \_\_\_\_\_\_\_\_\_\_\_\_

1) El siguiente método elimina todos los números negativos de un arreglo recorriendo todos los positivos posteriores a la izquierda. En los espacios sobrantes de la derecha se guardan ceros. Efectúa un análisis a priori de la complejidad del algoritmo en términos de los siguientes costos, para los casos mejor y peor:

* *c*1: costo de una lectura/escritura en el arreglo.
* *c*2: costo de una comparación con algún dato del arreglo.
* *c*3: costo de una operación con índices/tamaños.

void **eliminarNegativos**(int[] array) {

int last = array.length – 1;

[-3, -4, -1, 5, 7, 6]

[-4, -1, 5, 7, 6, 0]

[-1, 5, 7, 6, 0, 0]

[ 5, 7, 6, 0, 0, 0]

[ 5, 7, 6, 0, 0, 0]

[ 5, 7, 6, 0, 0, 0]

for (int i = 0; i <= last; ) {

if(array[i] < 0) {

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

array[j – 1] = array[j];

}

array[last] = 0;

last --;

} else {

i ++;

}

}

}

1.a) Para dar con las ecuaciones definitivas, llena la siguiente tabla. *6 puntos.*

|  |  |  |  |
| --- | --- | --- | --- |
| Instrucción | Costo | Veces (mejor) | Veces (peor) |
| int last = array.length – 1; |  |  |  |
| for (int i = 0; i <= last; ) { |  |  |  |
| if(array[i] < 0) { |  |  |  |
| for (int j = i + 1; j < last; j ++) |  |  |  |
| array[j – 1] = array[j]; |  |  |  |
| array[last] = 0; |  |  |  |
| last --; |  |  |  |
| } else { |  |  |  |
| i ++; |  |  |  |

1.b) Antes de escribir la ecuación, platícame cómo es el arreglo que induce el peor caso. No escribas ejemplos, generaliza la respuesta. *3 puntos.*

Escriba aquí la respuesta.

1.c) ¿El algoritmo es estable? Justifica tu respuesta. *2 + 3 puntos.*

Escriba aquí la respuesta.

1.d) Aquí escribe las ecuaciones que expresan el tiempo de ejecución en los casos mejor y peor, utilizando los costos arriba descritos. *6 puntos.*

1.e) De acuerdo a las ecuaciones anteriores, caracteriza el tiempo de ejecución el algoritmo utilizando notación asintótica. *4 puntos.*

1.f) De la misma forma, caracteriza el tiempo de ejecución del algoritmo en términos de comparaciones y de movimientos, por separado, utilizando notación asintótica. *6 puntos.*

2) Analiza el siguiente método y contesta las preguntas de abajo. Cuando se pregunte tiempo de ejecución, justifica breve y claramente las ecuaciones mostradas. Utiliza notación asintótica.

int **process** (List<String> words) {

PriorityQueue<String> pq = new PriorityQueue<>();

for(int i = 0; i < words.size(); i ++) {

String w = words.get(i);

pq.offer(w);

}

int F = 1, f = 1;

String u = null;

while(!pq.isEmpty()) {

String w = pq.poll();

if(w.equals(u)) {

f ++;

if(f > F) F = f;

} else {

f = 1;

}

u = w;

}

return F;

}

2.a) ¿Cuál es el tiempo de ejecución *T*(*N*) del primer ciclo for, suponiendo que *words* tiene *N* elementos y fue creado como un *ArrayList*? Justifica tu respuesta. *8 puntos.*

Escriba aquí la respuesta.

2.b) Misma pregunta, pero ahora *words* es un *LinkedList*. Justifica tu respuesta. *8 puntos.*

Escriba aquí la respuesta.

2.c) ¿Qué cambios sugieres en ese ciclo para que el tiempo de ejecución no sea afectado por el tipo de lista utilizado? Es decir, que sea siempre el mejor posible. *6 puntos.*

Escriba aquí la respuesta.

2.d) Con la mejora anterior, ¿cuál es el tiempo de ejecución total del algoritmo? *8 puntos.*

Escriba aquí la respuesta.

3) El siguiente método intercambia todos los valores adyacentes (*i*, *j*) de un arreglo tal que *i* es un índice par, *j* es un índice impar. Previamente hizo un procesamiento desconocido con el arreglo. El método *processData* tardó **8** segundos en acabar su ejecución con un arreglo **100 mil** elementos. Sin embargo, tardó alrededor de **32** segundos en acabar con un arreglo de **200 mil** elementos. ¿Qué tipo de algoritmo tiene el método *prepareData* y por qué? ¿constante, logarítmico, lineal, cuadrático, …? *Valor: 5 puntos.*

void **processData**(int[] array) {

*prepareData* (array);

for (int i = 0; i < array.length; i += 2)

*swap*(array, i, i + 1);

}

Escriba aquí la respuesta.

4) El siguiente método tardó **498** milisegundos en acabar su ejecución con un arreglo de **500,000** elementos. Sin embargo, tardó **997** milisegundos en acabar con un arreglo de **1 millón** de elementos. ¿Qué tipo de algoritmo tiene el método *processElement* y por qué? *Valor: 5 puntos.*

void **processData**(int[] array) {

for (int i = 0; i < array.length; i ++)

*processElement* (array, i);

}

Escriba aquí la respuesta.

5) Escribe la ecuación que expresa el tiempo de ejecución de cada caso descrito.

1. ¿Cuántos movimientos entre datos efectúa *Inserción* para ordenar arreglos que siguen esta serie? *4 puntos*
2. ¿Cuántas comparaciones entre datos efectúa *Inserción* para ordenar arreglos de *N* elementos que siguen la serie de arriba? *4 puntos*
3. ¿Cuántos movimientos de datos efectúa *Selección* para ordenar arreglos de *N* elementos que siguen la serie de arriba? *4 puntos*
4. ¿Cuántas comparaciones entre datos efectúa *Heapsort* para crear el max-heap a partir de un arreglo de *N* elementos que está ordenado de mayor a menor? *4 puntos*
5. ¿Cuántas iteraciones ejecuta *Conteo* para ordenar un arreglo que contiene los primeros *N* múltiplos de 12: {12(1), 12(2), …, 12(*N*)}? *4 puntos*

6) Se conoce que el tiempo de ejecución de un algoritmo para ciertas entradas es: . Resalta (subraya, colorea) aquellas notaciones que no contradicen a este tiempo de ejecución.

*Valor: 10 puntos.*

7) Te piden reacomodar una hilera de *N* árboles colocados a un costado de una avenida principal de forma que los árboles más bonitos y frondosos estén lo más adentro de la ciudad. El reacomodo implica un proceso costoso de extracción y replantación. No existe espacio disponible para mantener a más de un árbol fuera de la tierra. ¿Cuál algoritmo visto en clase podrías adaptar a este problema para minimizar los costos y por qué éste sí y cada uno de los otros no? *Valor: 4 + 6 puntos.*

Escriba aquí la respuesta.

8) El siguiente método escribe en un archivo los valores que tendrá cada inversión de una lista durante los próximos 30 días del ejercicio, incluyendo la inversión en el día 0. Describe el tiempo de ejecución en términos del número de inversiones *N* y considerando el número de llamadas a la función *addToFile*. Caracteriza el tiempo de ejecución obtenido utilizando la(s) notación(es) asintótica(s) más adecuada(s). *Valor: 5 puntos.*

void **calcularInteres**(double[] inversiones) {

for (int i = 0; i < inversiones.length / 2; i ++) {

*addToFile*(inversiones [i]);

for (int j = 1; j <= 30; i++) {

inversiones [i] \*= 1.0083;

*addToFile*(inversiones[i]);

}

}

}

9) El siguiente método suma los dígitos de un número entero positivo *N*. Si *N* = 345, devuelve 5+4+3=12. Describe su tiempo de ejecución en términos de *N*. *Valor: 5 puntos.*

int **sumDigits**(int n) {

int sum = 0;

while (n > 0) {

sum += n % 10;

n /= 10;

}

return sum;

}