ITC-ADA-C1-2023: Assignment #3

Luis Ballado
luis.ballado@cinvestav.mx

CINVESTAV UNIDAD TAMAULIPAS — February 9, 2023

1 Considerando el algoritmo recursivo mostrado a continuación, responda las siguientes preguntas

Algorithm 1: Algoritmo Misterio

```
entrada: Un arreglo A[0..n − 1] de números reales
if n==1 then
    return A[0];
else
    temp ← Misterio(A[0..n − 2])
    if temp <= A[n-1] then
        return temp;
    else
        return A[n-1];
```

Pregunta 1

¿Qué calcula el algoritmo?
El algoritmo recursivo calcula el elemento de minimo valor del arreglo

Pregunta 2

¿Cuál es el parámetro que indica el tamaño de la entrada del algoritmo? *n, siendo este el tamaño del arreglo de entrada. Apartir de este valor se puede llegar al caso base.*

Pregunta 3

¿Cuál es la operación básica del algoritmo?
La comparación es la operación básica de tiempo constante $O(1)$ en cada llamada, el algoritmo compara el valor minimo del subarreglo con el último elemento del subarreglo

Pregunta 4

¿Cuáles son el mejor caso y el peor caso para este algoritmo? *El mejor caso ocurre cuando el valor minimo es encontrado en la primera llamada donde el arreglo de entrada es pequeño A[0] y seria de tiempo constante $O(1)$*
El peor caso ocurre cuando el minimo valor es encontrado en la última llamada a la función. En este caso el algoritmo hace ***n*** *llamadas a función, siendo n el tamaño del arreglo. Convirtiendose en una complejidad lineal $O(n)$*

Pregunta 5

Proporcione una expresión matemática (relación de recurrencia), en función del tamaño de la entrada del algoritmo, que permita calcular cuántas veces se ejecuta la operación básica en este algoritmo.
T(n) = 1 cuando n = 1
T(n) = T(n-1)+1 cuando n > 1
T(n-1)+1 = T(n-2)+2 = T(n-3)+3 podemos decir que la recurrencia esta expresada como: $T(n-i)+i$
al ser $n-i=0$ *para llegar al caso base* $n=i$
T(0)+n = n+1; por lo tanto $T(n) \in \Theta(n)$

Pregunta 6

Resuelva la relación de recurrencia propuesta mediante substitución hacia atrás
Partiendo de la relación propuesta: $T(n) = T(n-1) + O(1)$
$T(n)$ *es el tiempo de complejidad del algoritmo para tamaños de entrada n,* $T(n-1)$ *es el tiempo de complejidad para tamaños* $n-1$
$O(1)$ *tiempo de complejidad por las comparaciones. La relación de recurrencia mediante substitución hacia atrás:*
$T(n) = T(n-1) + O(1)$
$T(n) = (T(n-2) + O(1)) + O(1)$
$T(n) = (T(n-3) + O(1) + O(1)) + O(1)$
$T(n) = (T(n-4) + O(1) + O(1) + O(1)) + O(1)$
podemos concluir que $T(n) = (T(1) + O(1) + O(1) + ... + O(1)) + O(1))$ donde $T(n)$ es el tiempo de complejidad para un array de tamaño n, $T(1)$ es la complejidad para tamaño 1 por la comparación constante

Pregunta 7

¿Cuál es la clase de eficiencia? $T(n) \in O(n)$

2 Dado el problema de encontrar el determinante de una matriz A de nxn , desarrolle los siguiente puntos:

Pregunta 8

Programe las versiones iterativa y recursiva del algoritmo para resolver el problema

2.1 Implementación

tarea1.cpp

```
#include <iostream>
#include <vector>

//Complejidad funcion principal O(n^2)
//por el doble for que recorre los arreglos
int main(){

  std::vector<int> a;
  a.push_back(2);
  a.push_back(5);
  a.push_back(5);
  a.push_back(5);

  std::vector<int> b;
  b.push_back(2);
  b.push_back(2);
  b.push_back(3);
  b.push_back(5);
  b.push_back(5);
  b.push_back(7);

  //vector de resultados
  std::vector<int>arr;

  int last_index = 0;

  //Usando fuerza bruta                    // O(n^2)
  for (int i = 0; i < a.size(); i++){      // O(n)
    for (int j = i; j < b.size(); j++){    // O(n)

      //std::cout << a[i] << "<-a comparacion b->"<< b[
    last_index] << "\n";

      if (a[i] == b[last_index]){
        arr.push_back(a[i]);
        break; // romper ciclo cuando sean iguales
      }

      last_index = i+1; //indice auxiliar para avanzar

    }
  }

  //--------------------
  // Imprimir resultado
  //--------------------

  std::cout << "El resultado es: \n";       // 1
  for(int i = 0; i<arr.size(); i++){        // n
    std::cout << arr[i] << "\n";
  }

}
```

ver código en github

Ejecutar desde una terminal

Command Line

```
$ g++ -o ./tarea1 ./tarea1.cpp
$ ./tarea1
```

Pregunta 9

Analice matemáticamente cada versión el algoritmo por separado usando las metodologías vistas en clase. *versión recursiva* $T(n) = T(n-1) + O(n^2)$ *donde* $O(n^2)$ *es el tiempo de complejidad del calculo de las submatrices, y* $T(n-1)$ *tiempo de complejidad cuando se reduce la matriz* $n-1$ *en un análisis por cofactores*
El tiempo de complejidad del cálculo de submatrices puede estar expresado por $O(n^2)$ ya que cada elemento pertenece a la primera fila, se va creando una submatriz de tamaño reducido $n-1xn-1$ para ir llegando al caso base.
$T(n) = T(n-1) + O(n^2)$
$T(n) = (T(n-2) + O(n^2)) + O(n^2)$
hasta llegar al caso base
$T(n) = (T(1) + O(n^2) + O(n^2) + ... + O(n^2) + O(n^2))$
podemos concluir que $T(n) = T(1) + n * O(n^2)$
$T(n) = O(n^3)$, pero para un peor caso donde la matriz es muy grande,
el algoritmo en su versión recursiva $T(n) \in \Theta(n!)$

Pregunta 10

En base a los resultados obtenidos en el punto anterior determine cuál de los dos algoritmos es más eficiente *Respuesta aqui*

Pregunta 11

Para cada uno de los dos algoritmos desarrollados aplique el método descrito en el apartado "Doubling ratio experiments" del libro Algorithms de Sedgewick y Wayne, generando instancias de tamaños 1000,2000,etc; hasta lograr un radio de 2^b, ejecutando 20 pruebas con cada tamaño y con cada algoritmo. Registre sus resultados. *Respuesta aqui*

Pregunta 12

Para cada algoritmo comparado realice una tabla con 5 predicciones, posteriores al tamaño con que se logró obtener el radio 2^b *Respuesta aqui*

Pregunta 13

Para cada algoritmo comparado grafique los siguientes resultados de sus ejecuciones *Respuesta aqui*

Pregunta 14

Con base en los experimentos realizados y considerando un tiempo máximo de ejecución sobre su computadora de 7 días, ¿Cuál es el tamaño máximo de entrada que puede resolver cada algoritmo analizado? *Respuesta aqui*

Pregunta 15

Conclusiones respecto al orden de crecimiento de cada algoritmo observado empíricamente y constrástelas contra los resultados de sus análisis matemático *Respuesta aqui*