Algoritmos y Estructuras de Datos - Parcial Módulo 2 - TE

Lunes 28 de noviembre de 2022 - TEMA 1

Apellido	Nombre	Legajo	Corrigió
[redacted]			

Ejercicio 1	Ejercicio 2	Ejercicio 3	Ejercicio 4	Total
4	1	2	1 50	8 50

Ejercicio 1 -- 4 puntos

Dado el siguiente algoritmo **calcule** el T(n) correspondiente,

```
public int met(int n){
        int c = 0;
        if (n >= 2){
            c += met(n-1)* met(n-1) + n;
        }
        return c;
}
```

Ejercicio 2 -- 2 puntos

a) Expresar la función T(n) del siguiente método (no es necesario desarrollar),

```
public static void calculo(int n){
        int i, j, k, x = 0; cte.
        for ( i=1; i <=n; i=i*2)
            for ( j=n; j >=1; j=j-2)
                for (k=0; k <= n; k++)
                    x=x*2; cte.
}
```

b) Expresar el O(n) del método del inciso anterior (no es necesario justificar),

Ejercicio 3 -- 2 puntos

Suponga que dispone de un algoritmo X, que resuelve un problema de tamaño n, y su función de tiempo de ejecución es $T(n)=n^2\log_2(n)$. Este algoritmo se ejecuta en una computadora que procesa 49.152 operaciones por cada segundo. Determine el tiempo que requerirá el algoritmo para resolver un problema de tamaño n=4.096. Justifique su respuesta.

$T(4096) = (4096)^2 . \log_2(4096)$

$T(4096) = 201326592$

49.152 ——— 1 seg.

201326592 ——— 4096 segundos.

MODULO 2 - TE

1) $T(n) \begin{cases} 1 & n<2 \\ 2T(n-1)+cte & n \geq 2 \end{cases}$ ✓

sea $n \geq 2$

paso ① $2T(n-1)+cte$

paso ② $2[2T(n-2)+cte]+cte$

$4T(n-2)+3cte$ ✓

paso ③ $4[2T(n-3)+cte]+3cte$

$8T(n-3)+7cte$

paso general $[2^i T(n-i)+(2^i-1)cte]$

caso base

$n-i<2$ ✗ aca deberia ir igualar a 1.0

$n-2 \leq i$

reemplazando

$2^{n-2}T(n-n-2)+(2^{n-2}-1)cte$

$2^{n-2}.T(1)+(2^{n-2}-1)cte$

$2^{n-2}.1+(2^{n-2}-1)cte$

$\frac{2^n}{2^{-2}}+\left(\frac{2^n}{2^{-2}}-1\right)cte$

$T(n) = 2^n.\frac{1}{4}+2^n.\frac{1}{4}.cte-cte$

2) a) $T(n)=\sum_{i=1}^{\log_2(n)}\left[\sum_{j=1}^{n/2}.\left(\sum_{k=0}^{n} cte\right)\right]$ ✓

b) $O(n)$ ✗ $n^2 \log_2 n$

Ejercicio 4 -- 2 puntos

1) ¿Cuál de las siguientes sentencias es **verdadera** de acuerdo con la **definición formal de BigOh**?

(a) $n \log n \in O(n^2)$
(b) $(n^2 + n) \in O(n^2)$
(c) $3^n \in O(4^n)$
(d) Todas las opciones anteriores
(e) Ninguna opción es correcta

2) Dado el siguiente algoritmo,

```
public static void calculo(int n){
 int i, j, k = 0;
 int x = 1;
 for (i = 1; i <= n; i=i*2){
        x=x*2;
        for (k = 5; k <= n+4; k++)
              x=x*2;
 }
}
```

¿Cuál es el orden de ejecución?

(a) $O(\log^2(n))$
(b) $O(n*\log(n))$
(c) $O(n+\log(n))$
(d) $O(n^2)$
(e) Ninguna de las opciones anteriores

3) Dado un árbol binario que contiene n elementos, ¿cuál es el orden de ejecución en el peor caso para consultar la existencia de un elemento en el árbol?

(a) O(log(n))
(b) O(n * log(n))
(c) $O(n^2)$
(d) O(n)
(e) O(1)

4) Indique cómo quedaría la función del T(n) del siguiente algoritmo

```
public int test(int n){
        int sum = 0;
        for (int k=1; k<=n; k=k*2)
            for (int j=1; j<=n/2; j++)
                sum++;
        return sum;
}
```

(a) $T(n)= c1 + c2 * \log_2^2 n$
(b) $T(n)= c1 * n^2 + c2$
(c) $T(n)= n^* \log^2_2 n + c1$
(d) $T(n)= c1 + (n/2) * \log_2 n$

Algoritmos y Estructuras de Datos - Curso 2022 - Parcial Módulo 3 - Grafos
Lunes 28 de noviembre de 2022 - TEMA 1

Ejercicio 1	Ejercicio 2	Ejercicio 3	Total
1	1	2	5

Ejercicio 1 -- 5 puntos

Un grupo de amigos ecuatorianos asistieron el domingo 20 al primer partido del mundial en la ciudad de Jor, en el cual se enfrentaron Ecuador-Qatar.
Luego de disfrutar el encuentro, contrataron un auto en una agencia turística por una cantidad de kilómetros, con el objetivo de conocer por fuera el resto de los estadios.
La agencia les entregó un mapa que contiene los 8 estadios que están distribuidos en 5 ciudades diferentes.
Este mapa se puede modelizar con un grafo sin dirección, donde cada vértice representa un estadio y las aristas las rutas que los conectan. De cada estadio se conoce: su nombre y el nombre de la ciudad a la que pertenece. De cada arista la cantidad de kilómetros.

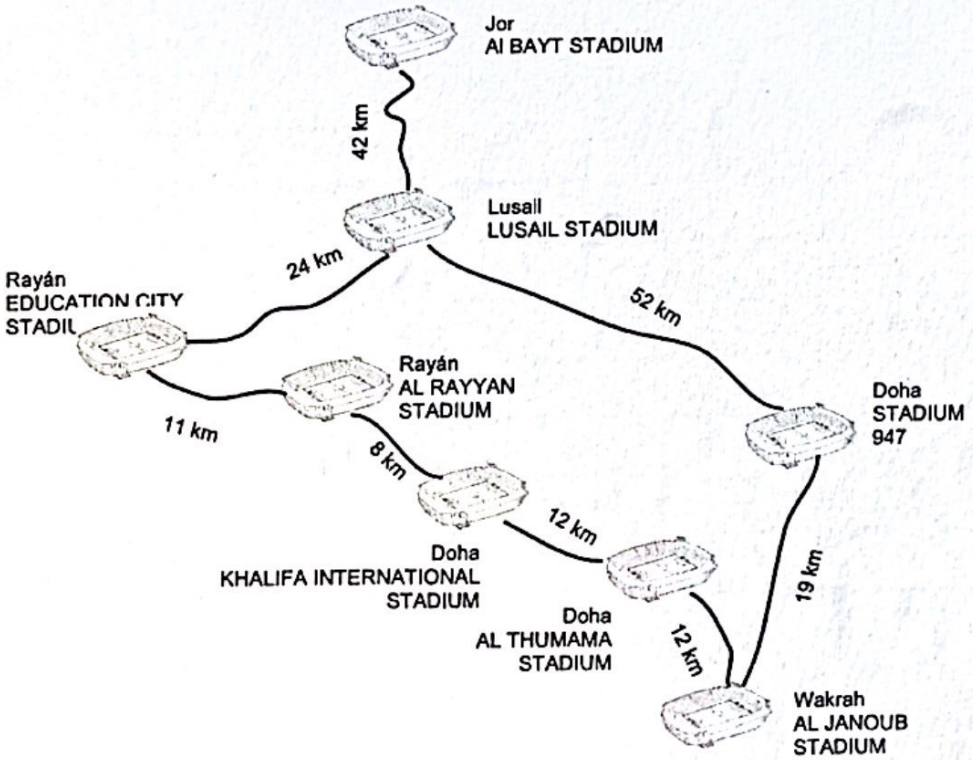
Se debe implementar el siguiente método:

ListaGenerica <String> **estadios** (Grafo<???> mapaEstadios, String estadioOrigen, int cantKm)

El cual recibe el mapa de los estadios, el nombre del estadio de la ciudad de Jor, "Al BAYT STADIUM" y la cantidad de kilómetros contratados.

El algoritmo debe retornar una lista con los nombres de los estadios que pueden recorrer como **máximo** en esa cantidad de kilómetros.
Tenga presente que para la cantidad de kilómetros contratados pueden existir distintos caminos posibles, por lo cual debe retornar el que visite la **MAYOR cantidad de estadios**.

Por ejemplo, si el mapa entregado a los turistas fuese el siguiente y la cantKm = 100.



MÓDULO 3 - Grafos

ejercicio (1)

```
public ListaGenerica<String> estadios (Grafo<Vertice<Estadios>> mapaEstadios, String estadioOrigen,
        int cantKm) {
    boolean [] marca = new boolean [mapaEstadios.listaDeVertices().tamanio()+1];
    boolean encontre = false;
    ListaGenerica<Vertice<Estadio>> vertices = mapaEstadios.listaDeVertices();
    Vertice<Estadio> aux;
    vertices.comenzar();
    int indice = -1;
    while (!vertices.fin() && !encontre) {
        aux = vertices.proximo()
        if (aux.dato.getNombreEstadio().equals(estadioOrigen)) {
            indice = aux.posicion();
            encontre = true;
        }
    }
    int ciudadMax=0;
    Max max = new Max();
    int kmActual = cantKm;
    ListaGenerica<String> recorridoActual = new ListaEnlazadaGenerica<String>();
    recorridoActual.agregarFinal(mapaEstadios.listaDeVertices().elemento(indice).dato().getNombreEstadio());
    recorrido(indice, estadios, ciudadMax+1, recorridoActual, marca, kmActual, max);

    return max.recorridoMaximo();
}

private void recorrido(int i, Grafo<Vertice<Estadios>> estadios, int ciudadesMax, ListaGenerica<String> recorridoActual,
        boolean [] marca, int kmActual, Max max) {
    marca[i] = true;
    if (ciudadesMax > max.getCantCiudades()) {                    (*)1
        max.setCantCiudades = ciudadesMax;
        max.actualizarRecorrido(recorridoActual);
    }
    else {
        Vertice<Estadio> v = estadios.listaDeVertices.elemento(i);
        ListaGenerica<Arista<int>> ady = estadios.listaDeAdyacentes(v);
        ady.comenzar();
        while (!ady.fin()) {
            Arista<int> aux = ady.proximo();
            int j = aux.verticeDestino().posicion();
            if (!marca[j] && aux.peso() >= kmActual) {
                recorridoAct.agregarFinal(estadios.listaDeVertices().elemento(j).getDato().getNombreE());
                recorrido(j, estadios, ciudadesMax+1, recorridoActual, marca, kmActual-aux.peso(), max)
                recorridoAct.eliminarEn(recorridoAct.tamanio());
            }
        }
    }
}
```

(*)1 deberías actualizar cuando no te alcanzan más los kms no cada vez que entrás así no se hace un set

??

(*)2 así como marcas, ya que buscas el mejor camino, deberías desmarcar.

```
public class Estadio{
    private String nombreE, nombreC;
    public Estadio(String estadio, String ciudad){
        this.nombreE = estadio;
        this.nombreC = ciudad;
    }

    public void setNombreE (String est){
        this.nombreE = est;
    }

    public String getNombreE (){
        return this.nombreE
    // getters/setters de ciudad no implementado x no ser necesario.
}

public class Mox{
    private int mox;
    private ListaGenerica<String> estadios;

    public Mox (){
        this.mox = 0;
        this.estadios = new ListaGenerica<String>(); enlazada
    }

    public void setCantCiudades (int cant){
        this.mox = cant;
    }
    public int getCantCiudades (){
        return this.mox;

    public void actualizarRecorrido (ListaGenerica<String> nuevo){

        estadios.comenzar();
        while (!estadios.fin()){
            estadios.eliminarEn (estadios.tamanio());
        }
        nuevo.comenzar();
        while (!nuevo.fin()){
            estadios.agregarFinal(nuevo.proximo());
        }
    }
}
```

La lista por retornar sería:

Al BAYT STADIUM, LUSAIL STADIUM, EDUCATION CITY STADIUM, AL RAYYAN STADIUM, KHALIFA INTERNATIONAL STADIUM, AL THUMAMA STADIUM

Con un total de 97 kilómetros recorridos (42+ 24 + 11 + 8 +12).

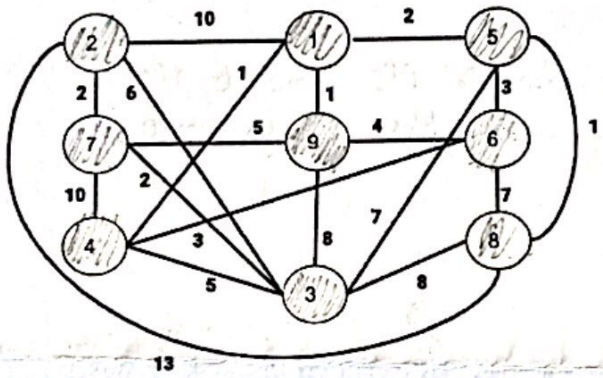
Otro camino que se podría hacer sería: Al BAYT STADIUM, LUSAIL STADIUM, STADIUM 947. Con 94 km recorridos. Pero **NO** recorre la **MAYOR** cantidad de estadios.

Notas

- No se puede pasar 2 veces por el mismo lugar
- Use los métodos de Grafo y Listas vistos en clase.
- NO recorra más de una vez el grafo y los caminos hallados.
- El signo de ??? debe ser completado con un tipo que Ud. considere adecuado.
- Seguro existe un camino que tiene al menos el nombre del estadio origen.

Ejercicio 2 -- 3 puntos

Se desea ejecutar el algoritmo de **Dijkstra** sobre el siguiente grafo pesado, a partir del vértice '2'.

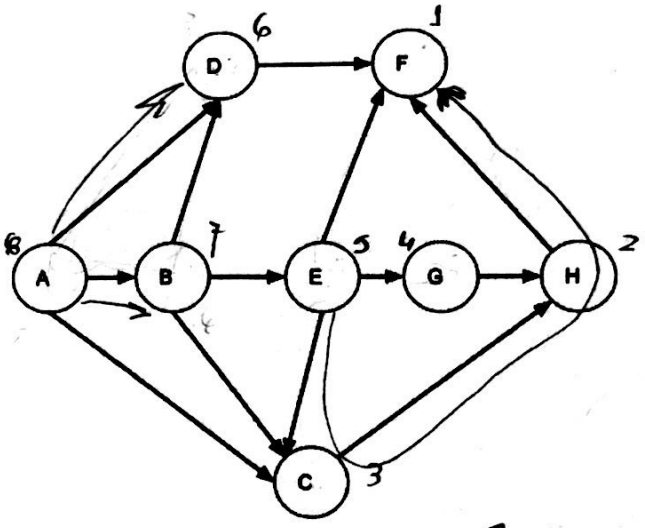


Muestre todos los pasos intermedios, indicando el orden en que se van procesando los vértices.

Orden en que toma el vértice	Vértices v	Distancia (2,v)	Previo	Visitado
5°	1	∞ 10 8	2 9	0 1
1°	2	∞ 0		0 1
3°	3	∞ 8 4	7 2 7	0 1
6°	4	∞ 12 11 9	7 3 1	0 1
7°	5	∞ 11 11 10	3	0 1
8°	6	∞ 11	9	0 1
2°	7	∞ 2	2	0 1
9°	8	∞ 13 12 13 12 11	2 3 2 3 5	0 1
4°	9	∞ 7	7	0 1

Ejercicio 3 – 2 puntos

Indicar cuál es la ordenación topológica para el siguiente grafo dirigido acíclico, utilizando la estrategia que trabaja con un recorrido DFS comenzando por el vértice E (tanto los vértices como los adyacentes se procesan en forma ascendente).



1) Realizo el recorrido DFS comenzando por E y voy en primo de post orden y enumerándolos ascendentemente,

mi pila será

E_5
G_4
C_3
H_2
F_1

me quedan unos nodos por recorrer entonces tomaré el vértice A

A_8
B_7
D_6
E_5
G_4
C_3
H_2
F_1

finalmente desapilo y tendré mi ordenación topológica

SALIDA => A B D E G C H F