Segundo parcial – 23 de mayo de 2024

P.P.

Nombre y Apellido: Szapowalo Joan **Calificación:** Aprobado

(5 hojas en total)

IMPORTANTE: Resolver cada ejercicio en una hoja separada, indicando claramente el número del ejercicio. Colocar en cada hoja su nombre y apellido. Puede resolver en lápiz solo si no hay borrones y el texto se lee fácilmente sobre la hoja.

Ej 1- Dado un árbol binario que proviene de la transformación de un bosque de árboles generales, determinar si en el bosque todos los árboles verificaban que su grado era igual al grado de su raíz.

Ej 2- (*Utilizar TDA N-ario*) Dado un árbol N-Ario de enteros determinar mediante una función int, si existe algún nodo (no hoja) que verifique que el grado de todos sus hijos es creciente.
Si la solución es void, el puntaje obtenido será no mayor a 1,75p

Ej 3- Dada la matriz de alcance ALC de un digrafo G = (V, E) con |V|=N, obtener la cantidad de nodos que sin tener bucle son alcanzados y alcanzan la misma cantidad de vértices.
El tratamiento de la matriz debe ser completamente recursivo.
No debe recorrer la matriz más de una vez

Ej 4- Dada la matriz de adyacencia de un dígrafo con vértices etiquetados de A a E, y los vectores D y P generados al aplicar parcialmente Dijkstra en dicho dígrafo partiendo desde A. Sabiendo que S ya contiene A y D (agregados en ese orden), se pide completar la ejecución del algoritmo

$$MAdy = \begin{bmatrix} 2 & 15 & 0 & 13 & 0 \\ 0 & 0 & 4 & 2 & 7 \\ 14 & 0 & 0 & 3 & 2 \\ 0 & 33 & 8 & 0 & 0 \\ 3 & 1 & 4 & 7 & 0 \end{bmatrix}$$

D = [0 15 21 13 INF] P = [0 0 D 0 INF]

Serán considerados al calificar este examen la eficiencia de las soluciones y el uso de las características del lenguaje C y de los principios de la programación estructurada. En los ejercicios 1 a 3, desarrollar un main.c completo que incluya declaraciones, inicializaciones, invocaciones y resultados obtenidos. En los ejercicios 1 y 3, declarar tipos.

*Para **aprobar** es necesario obtener al menos **5p**; **4,5p** de los cuales deben obtenerse entre los ejercicios **1 a 3***

*Para acceder al coloquio de **promoción** es necesario obtener al menos **6p** y obtener al menos **el 50% de cada ejercicio** y **al menos 5,5 entre los ejercicios 1 a 3***

Ej 1 (3 p)	Ej 2 (3 p)	Ej 3 (3p)	Ej 4 (1 pto.)	FINAL
3	3	2,50	1	9,50

FELICITACIONES!

Szapowalo Joan
hoja ①

HOJA N°
FECHA

1) Arbol binario que proviene de un bosque, determinar si en el bosque todos los arbdes verificaban que su grado era igual al grado de su raiz.

```
# include <stdio.h>
# include <stdlib.h>
typedef struct nodoA{
    int dato;
    struct nodoa *izq, *der;} nodoA;
typedef nodoA *arbol;
int main (){
    arbol bosque;
    :
    CargaBosque(&bosque);
    :
    Determina(bosque) ? printf ("El bosque verifica"):
                        Printf ("El bosque no verifica");
    Printf ("que para cada arbol su grado era igual al de su raiz");
    return 0;
}
```

NOTA Corregir

```
int arbolCumple(arbol a, int GrRaiz){
    arbol aux;
    int grado;
    if (a!=NULL){
        aux = a->izq;
        grado = 0;
        while(aux!=NULL && grado<=GrRaiz)
            grado++;
            aux = aux->der;
        }
        if(grado<=GrRaiz && aux==NULL)
            return arbolCumple(a->izq, GrRaiz) && arbolCumple(a->der, GrRaiz);
        else
            return 0;
    }
    else
        return 1;
}
int Determina(arbol bosque){
    arbol aux;
    int GrRaiz, Cumple=1;
    While(bosque!=NULL && Cumple){
        aux = bosque->izq;
        GrRaiz = 0;
        While(aux!=NULL){
            GrRaiz++;
            aux = aux->der
        }
        if(GrRaiz)
            Cumple == arbolCumple(bosque->izq, GrRaiz)
        bosque = bosque->der;
    }
    return Cumple;
}
```

Modularizar

Modularizar

NOTA

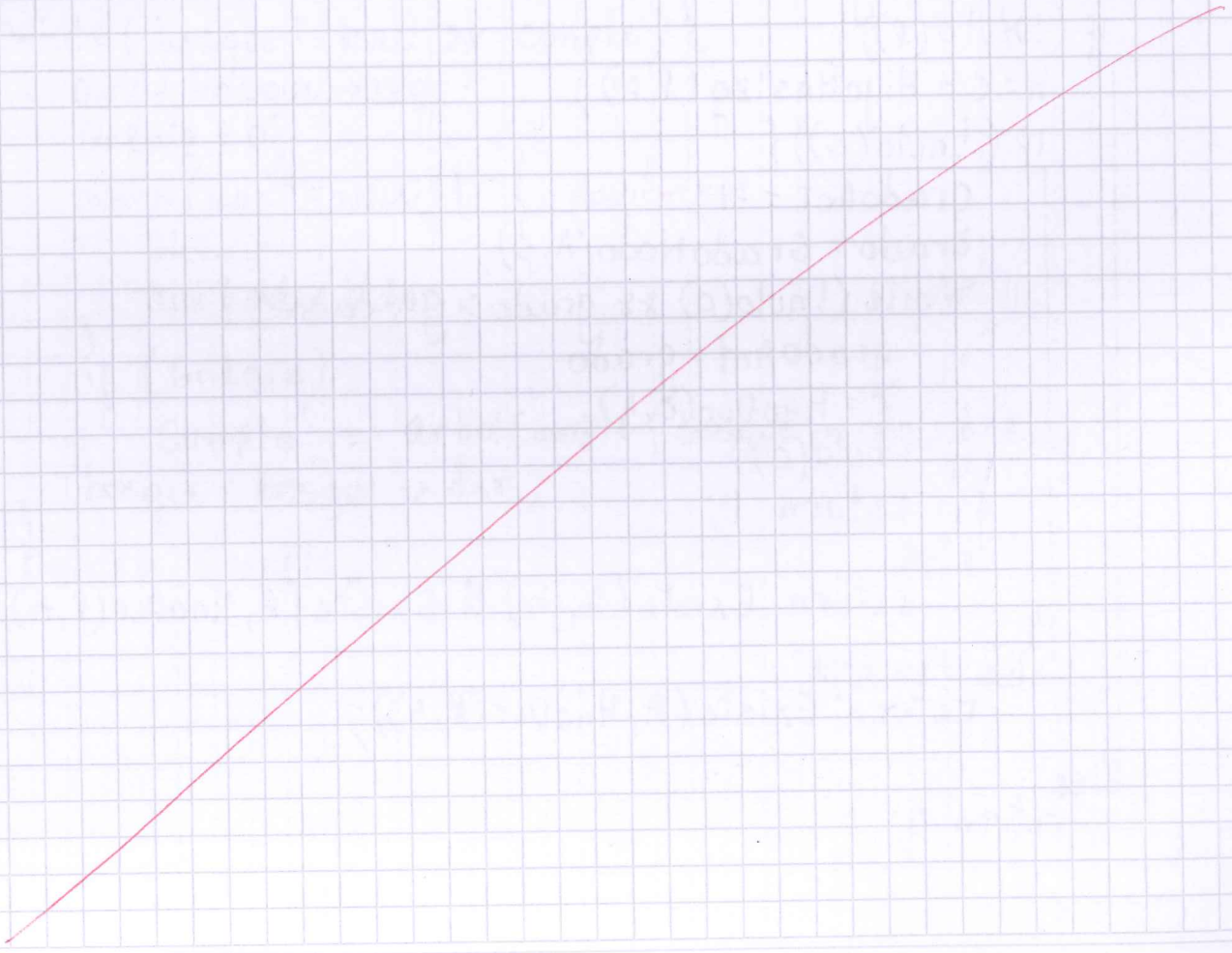
Szapowalo Joan
hoja ②

HOJA N°
FECHA

2)
```
int GradoNodo(ArbolNA, posicion p){
    int grado;
    posicion c;
    if(!nulo(p)){
        c=HijoMasIzq(p,A);
        grado = 0;
        while(!nulo(c)){
            grado++;
            c=HnoDer(c,A);
        }
        return grado;
    }
    else                    //se considera que el arbol nulo tiene grado -1
        return -1;  //esto podría ser de utilidad, pero no para este prob.
}
int Existe(ArbolNA, posicion p){
    posicion c, h;
    int grado, gradoAnt;
    if(!nulo(p)){
        h=c=HijoMasIzq(p,A);
        if(!nulo(c)){
            gradoAnt = -1;
            grado = GradoNodo(A,c);
            while(!nulo(c) && grado > gradoAnt){
                gradoAnt = grado;
                c=HnoDer(c,A);
            }
            if(nulo(c))
                return 1;
            else
                return Existe(A,h) || Existe(A,HnoDer(p,A));
        }
        else //es hoja
            return Existe(A,HnoDer(p,A));
    }
    else
        return 0;
}
```

NOTA

```
# include <stdio.h>
# include <stdlib.h>
# include "Narios.h"
int main(){
    ArbolN A;
    :
    Carga(&A);
    :
    if (Existe(A, Raiz(A))
        Printf ("El arbol tiene");
    else
        Printf ("El arbol no tiene");
    printf ("un nodo no hoja cuyo grado de los hijos es creciente");
    return 0;
}
```



NOTA

Szapowalo Joan
hoja ③

alcanza

HOJA N°

FECHA

3) Dada la mat de alcance ($|V| = N$) de un digrafo obtener la cant de ~~bucles~~ nodos que sin tener bucle son alcanzados y alcanzan la misma cant de vertices.

```
int NodosCumplen(int MatAlc[][MAX], int i, int j, int N, int Alc, int EsAlc){
    if (i < N)
        if (j < N)                                   MatAlc
            if (Mat[i][i] != 0) // tiene bucle => No analizo dicho vertice
                return NodosCumplen(MatAlc, i+1, 0, N, 0, 0);
            else
                return NodosCumplen(MatAlc, i, j+1, N, Alc + MatAlc[i][j], EsAlc+
                                                                    +MatAlc[j][i]);
        else
            return Alc == EsAlc + NodosCumplen(i+1, 0, N, 0, 0);
    else
        return 0;
}
```

5 parametros
(falta 1: MatAlc)

dos: visito 2 veces c/celda.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
int main(){
    int MatAd[][MAX], ALC[][MAX], N;
    :
    Carga(MatAd, &N);
    Genera(ALC, N);
    :
    Printf("La cantidad de vertices sin bucle que son alcanzados
            y alcanzan la misma cantidad es: %d",
            NodosCumplen(ALC, 0, 0, N, 0, 0));
    return 0;
}
```

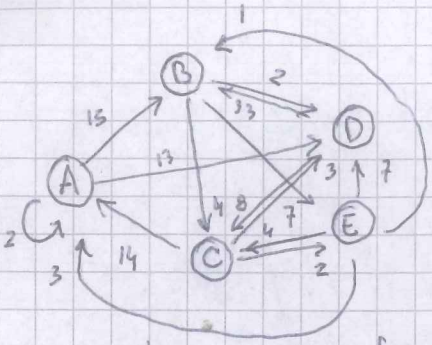
cant. filas?

NOTA

Joan

Dijkstra

$$MAdy = \begin{bmatrix} 2 & 15 & 0 & 13 & 0 \\ 0 & 0 & 4 & 2 & 7 \\ 14 & 0 & 0 & 3 & 2 \\ 0 & 33 & 8 & 0 & 0 \\ 3 & 1 & 4 & 7 & 0 \end{bmatrix}$$



Se tiene hasta el momento: $S = \{A, D\}$

$D = [0 \ 15 \ 21 \ 13 \ \infty]$

$P = [0 \ 0 \ D \ 0 \ \infty]$

Donde ya ha sido analizado D para encontrar el camino mínimo a todo vertice desde A.

Se agrega entonces para su analisis al vertice con menor costo que no haya sido analizado hasta el momento...

1) $S = \{A, D, B\}$ $VV = [1 \ 1 \ 0 \ 1 \ 0]$

$D[C] = \min(21, 15 + 4) = \boxed{19}$

$D[E] = \min(\infty, 15 + 7) = \boxed{22}$

$D = [0 \ 15 \ 19 \ 13 \ 22]$

$P = [0 \ 0 \ B \ 0 \ B]$

Agregamos a S un vértice que aun no haya sido analizado para ver si reduce el costo pasar por el para llegar a algun vertice no analizado todavia

2) $S = \{A, D, B, C\}$ $VV = [1 \ 1 \ 1 \ 1 \ 0]$

$D[E] = \min(22, 19 + 2) = \boxed{21} \Rightarrow$

$D = [0 \ 15 \ 19 \ 13 \ 21]$

$P = [0 \ 0 \ B \ 0 \ C]$

~~Vemos que los vectores P y D se mantienen sin cambios ya que pasar por C no reduce ningun costo~~

3) $S = \{A, D, B, C, E\}$ $VV [1 \ 1 \ 1 \ 1 \ 1]$

El algoritmo de Dijkstra concluye, una vez incluido el ultimo vertice.

NOTA