Universidad del Rosario | Escuela de Ingeniería, Ciencia y Tecnología | MACC

Algoritmos y Estructuras de Datos – Grupo 3

Parcial 1

Simulador de Fraude

La DIAN se encuentra actualmente en proceso de revisión fiscal de grupos empresariales, para poder identificar fraudes. Esta entidad ha detectado que existen irregularidades entre los montos totales de las transacciones entre las empresas que proveen los bancos y los recibos de las transacciones que la empresa le reporta a la DIAN. Para esto, le pide que diseñe una solución informática para entrenar a nuevos agentes de aduana, en donde genere situaciones ficticias para que ellos puedan identificar los casos. Esta es la información que debe tener en cuenta para crear el simulador de escenarios:

- Cada grupo empresarial del simulador cuenta con *num_emp* empresas, las cuales realizan transacciones entre sí (compras y ventas). Una empresa no va a tener transacciones consigo misma. Se puede definir un número de empresas diferente en cada ejecución.
- Se debe generar una lista que simule todas las transacciones que se han realizado entre las empresas. Cada transacción debe incluir:
    - Empresa origen
    - Empresa destino
    - Monto de la transacción

  Esta información se almacenará en una lista enlazada que contiene un objeto de tipo Transaccion y el puntero al siguiente elemento. La lista puede tener más punteros si lo desea. Usted debe definir las clases Transaccion y Lista, con base en la plantilla provista y el material de clase. Recuerde que no debe haber ninguna transacción que tenga valores de origen y destino iguales. Para esto genere un origen aleatorio y tantos destinos sean necesarios hasta que encuentre uno diferente.

- Con base en las transacciones, se calculará la matriz de *num_emp* x *num_emp*, que simula la información que el banco proveería sobre el monto total de las transacciones que se han realizado entre las empresas. Cada celda representa el valor de cuanto se ha trasladado entre la primera empresa (origen) a la segunda empresa (destino).
- Para simular la ocurrencia de fraude, se inducen errores en algunas transacciones (aquellas en donde la operación módulo entre el índice y una variable llamada *error* sea igual a 0) reduciendo su valor en un 25% (monto*0.75).
- Con este nuevo conjunto de transacciones alteradas, se procede a calcular una segunda matriz, de manera similar a como se calculó la original.
- Se comparan las matrices celda por celda y se escribe por pantalla en aquellas ubicaciones en donde hay diferencias.

Ejemplo

int num_emp = 4; //Número de empresas

int num_trans = 10; //Número de transacciones

int error = 3; //Variable para incluir errores en las transacciones

- Lista de transacciones

(2,3,44281)

(1,3,44477)

(3,2,12485)

(1,2,50454)

(2,3,62939)

(0,2,63023)

(0,3,47957)

(3,2,50842)

(0,2,82411)

(2,0,6122)

- Matriz original

0	0	145434	47957
0	0	50454	44477
6122	0	0	107220
0	0	63327	0

- Transacciones alteradas

(2,3,33210)

(1,3,44477)

(3,2,12485)

(1,2,37840)

(2,3,62939)

(0,2,63023)

(0,3,35967)

(3,2,50842)

(0,2,82411)

(2,0,4591)

- Matriz alterada

0 0 145434 35967

0 0 37840 44477

4591 0 0 96149

0 0 63327 0

- Presentación de diferencias

Error en total de transacciones entre empresas 0 y 3 por -11990

Error en total de transacciones entre empresas 1 y 2 por -12614

Error en total de transacciones entre empresas 2 y 0 por -1531

Error en total de transacciones entre empresas 2 y 3 por -11071

Plantilla

```
#include<iostream>

using namespace std;

class Transaccion{
  /*
  Definir atributos
  */

  public:

  Transaccion(){
      //Inicializar atributos en 0
  }

  Transaccion(/*parametros de entrada*/){
      //Inicializar atributos con parámetros de entrada
  }

  int get_atributo(){
      return atributo;
  }

  void set_aributo(int a){
      atributo = a;
  }

  Transaccion& operator=(const Transaccion& f) {
        atributo = f.atributo;
        return *this;
    }


  string to_string() {
        return "("+std::to_string(atributo) + "," + std::to_string(atributo)+
"," + std::to_string(atributo)+")";
    }

  friend std::ostream& operator<<(std::ostream& os, Transaccion& b) {
        return os << b.to_string();
    }

};

//Definición de lista con transacciones
```

```
class Lista{
    //Atributos

public:
    //Métodos
    Lista(){
        //Inicializar lista vacía
    }

};

int main()
{
    srand(1234);
    int num_emp = 4; //Número de empresas
    int num_trans = 100; //Número de transacciones
    int error = 16; //Variable para incluir errores en las transacciones
    int monto_max = 100000; //Variable para definir el máximo valor de un
monto

    int** mat_bank; //Declaración de la matriz con información reportada por
el banco, calculada con las transacciones originales
    int** mat_emp; //Declaración de la matriz con información alterada,
reportada por la empresa, calculada con las transacciones alteradas


    Lista t = new Lista();

    //Creación de transacciones. Valide que no haya transacciones con un mismo
origen y destino, y que el monto no pase del máximo establecido: monto =
rand()%100000;


    //Imprimir lista de transacciones originales
    for(int i=0; i<num_trans; i++){
        cout<<t.get(i)<<endl;
    }

    //Calcular monto total de transacciones entre empresas en mat_bank

    //Imprimir la matriz mat_bank

    //Alterar las transacciones cuyo indice%error == 0, disminuyendo su valor
en 25% (monto*0.75)

    //Imprimir lista de transacciones alteradas
```

```
    //Calcular monto total de transacciones alteradas entre empresas en
mat_alt

    //Imprimir la matriz mat_alt

    //Imprimir las diferencias encontradas entre ls matrices
    for(int i=0; i<num_emp; i++){
        for(int j=0; j<num_emp; j++){
            if(mat2[i][j] != mat[i][j])
                cout<<"Diferencia en total de transacciones entre empresas
"<<i<<" y "<<j<<" por "<<(mat_alt[i][j]-mat_bank[i][j])<<endl;
        }
    }

    return 0;
}
```

Nota

Si su programa funciona correctamente y genera los valores en el orden esperado, los valores de su implementación deberían ser iguales a los del ejemplo, pues el generador de números aleatorios tiene una semilla idéntica definida por la función srand(1234). No es obligación que sean iguales, pero si lo son, podría ser un indicador de que tiene la solución correcta.