Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación de estructuras de datos y algoritmos fundamentales (Gpo 602)

Act 4.3 - Actividad Integral de Grafos

Adela Alejandra Solorio Alcázar A01637205
Luz Patricia Hernández Ramírez A01637277
Marcela Beatriz De La Rosa Barrios A01637239

18/11/2023

Act 4.3 - Actividad Integral de Grafos

El código presentado demuestra una implementación práctica de la teoría de grafos para analizar registros de una bitácora. La estructura se centra en la clase `Graph`, que utiliza una lista de adyacencia para representar las conexiones entre nodos, donde cada nodo es una sección de una dirección IP. La elección de una estructura de grafo es fundamental para comprender las relaciones y conexiones entre diferentes partes de una red.

La lectura del archivo se realiza a través de un flujo de entrada. Cada línea se procesa para extraer información clave y la dirección IP se divide en secciones, y se añaden bordes al grafo conectando las secciones consecutivas. Este enfoque permite visualizar las relaciones de conectividad entre diferentes partes de la dirección IP.

La función `outDegree` de la clase `Graph` calcula el grado de salida de cada nodo en el grafo. Los resultados se almacenan en un vector y se ordenan de manera descendente según el grado de salida. Se imprime el top 10 de direcciones IP con los mayores grados de salida, brindando una comprensión visual de las partes de la red que están más frecuentemente conectadas.

En conjunto, este código no solo implementa conceptos avanzados de grafos, sino que también aborda problemáticas específicas, como la identificación de direcciones IP particulares y la visualización de las conexiones más fuertes en la red.

El código implementado se observa así:

```cpp
#include <iostream>
#include <fstream>
#include <list>
#include <vector>
#include <sstream>
#include <algorithm>
#include <string>
using namespace std;

const int maxVert = 999;

struct Node {
    int section;
    int outDegree;
    Node (int s, int o) {
        section = s;
        outDegree = o;
    }
};

class Graph {
    private:
        int numVert;
        list<int> *adjList;

    public:
        Graph(){
            numVert = 0;
            adjList = new list<int>[maxVert+1];
        }
```

```cpp
        void addEdge(int startVertex, int endVertex){
            if (startVertex >= 0 && startVertex < maxVert && endVertex >= 0 && endVertex < maxVert) {
                adjList[startVertex].push_back(endVertex);
            } else {
                cerr << "Vértices no permitidos" << endl;
            }
        }

        void printAdjList(){
            for(int i = 0; i <= maxVert; i++){
                cout << i << ": ";
                for(list<int>::iterator it = adjList[i].begin(); it != adjList[i].end(); it++){
                    cout << *it << " ";
                }
                cout << endl;
            }
        }

        int outDegree(){
            int outDegree;
            vector<Node> outDegreeVector;


            for(int i = 0; i <= maxVert; i++){
                outDegree = adjList[i].size();
                Node node(i, outDegree);
                outDegreeVector.push_back(node);
            }

            sort(outDegreeVector.begin(), outDegreeVector.end(),
```

```
            sort(outDegreeVector.begin(), outDegreeVector.end(),
            [](const Node &a, const Node &b) {return a.outDegree > b.outDegree;});

            for(int i = 0; i < 10; i++){
                cout << "IP segment: " << outDegreeVector[i].section << ", Out-degree value: " << outDegreeVector[i].outDegree << endl;
            }

            int maxOutDegree = outDegreeVector[0].outDegree;

            return maxOutDegree;
        }

};

int main(){


    ifstream inputFile("bitacora.txt");
    if (!inputFile.is_open()) {
        cerr << "Error: No se pudo abrir el archivo" << endl;
        return 1;
    }

    Graph g;

    string mes, fecha, ipAddress, razon, ipComplete;
    int dia, startVertex, endVertex;

    cout << "Direcciones IP donde se encuentra el boot master: " << endl;

    while (inputFile >> mes >> dia >> fecha >> ipAddress) {
        getline(inputFile, razon);
        razon = razon.substr(1);
        ipComplete = ipAddress.substr(0, ipAddress.find(":")); //obtener el ip address completo, sin port number
        istringstream ipStream(ipComplete);
```

```
        istringstream ipStream(ipComplete);
        int ipSections[4];
        char dot;
        for (int i = 0; i < 4; i++) {
            ipStream >> ipSections[i];
            if (i < 3) {
                ipStream >> dot;
            }
        }

        bool contains60 = false;

        for (int i = 0; i < 3; i++) {
            startVertex = ipSections[i];
            endVertex = ipSections[i+1];
            g.addEdge(startVertex, endVertex);

            if (startVertex == 60 || endVertex == 60) {
                contains60 = true;
            }
        }

        if (contains60) {
            cout << mes << " " << dia << " " << fecha << " " << ipAddress << " " << razon << endl;
        }

    }

    cout << "\nTop 10 direcciones IP con out-degree value más grande: " << endl;

    g.outDegree();

    inputFile.close();

    return 0;
```

Salida del código:

```
Direcciones IP donde se encuentra el boot master:
Sep 1 18:43:34 839.60.816.51:6940 Failed password for illegal user test
Aug 8 18:51:12 176.60.269.48:4608 Illegal user
Aug 21 16:34:35 54.56.965.60:5144 Failed password for illegal user guest
Jun 7 04:55:38 807.68.577.60:4579 Illegal user
Jul 21 00:37:42 996.60.573.10:5084 Failed password for illegal user test
Aug 18 05:33:56 893.60.774.73:4681 Failed password for illegal user guest
Jun 28 15:11:59 30.19.459.60:5420 Failed password for root
Aug 20 21:39:49 34.60.781.17:4657 Failed password for admin
Sep 17 07:45:06 846.60.350.94:5481 Illegal user
Jun 8 00:42:26 740.34.674.60:5400 Failed password for root
Oct 30 17:51:22 113.83.60.73:5479 Failed password for admin
Oct 7 12:33:35 879.4.904.60:4344 Failed password for admin
Jun 16 13:01:48 952.60.500.33:4029 Failed password for admin
Oct 12 23:54:21 163.14.81.60:6680 Failed password for admin
Oct 16 01:51:25 859.60.373.98:6073 Failed password for root
Sep 29 07:43:40 232.62.564.60:5504 Failed password for admin
Oct 30 13:55:03 594.72.60.18:6594 Failed password for illegal user test
Sep 1 23:00:31 362.60.26.61:6035 Failed password for root
Aug 2 11:09:49 714.60.385.52:4758 Failed password for root
Sep 24 22:41:39 82.65.970.60:5519 Failed password for root
Oct 12 20:10:28 851.86.222.60:4730 Illegal user
Jun 4 04:56:40 478.25.977.60:5573 Illegal user
Aug 13 10:57:40 33.18.843.60:6065 Failed password for illegal user guest
Jul 12 10:34:55 511.60.700.28:5327 Illegal user
Jul 29 03:17:35 483.5.784.60:6225 Failed password for root
Oct 5 09:38:58 111.13.814.60:5158 Failed password for admin
Sep 23 13:10:39 810.79.60.12:4185 Failed password for root
Oct 13 13:36:30 427.60.586.6:6080 Failed password for admin
Oct 2 11:28:41 159.60.163.11:5020 Failed password for root
Jul 8 15:12:11 225.60.869.38:5253 Failed password for admin
Jul 26 14:50:36 808.60.579.45:5778 Failed password for root
Jun 17 08:17:11 443.86.20.60:5349 Failed password for illegal user test
Aug 18 23:02:29 398.60.8.25:6010 Failed password for admin
Jul 3 03:15:19 637.60.429.21:5315 Failed password for illegal user test
Jun 17 01:20:53 60.42.722.83:5607 Failed password for illegal user test
Jul 22 01:15:14 840.10.936.60:5837 Failed password for illegal user test
Aug 8 13:50:36 986.54.60.96:5188 Failed password for root
Sep 28 02:55:12 32.60.799.22:6686 Failed password for admin
Sep 22 03:36:02 138.20.60.79:4587 Illegal user
```

```
Sep 15 22:21:18 971.60.714.7:6926 Failed password for admin
Sep 29 07:15:40 536.88.816.60:6260 Illegal user
Jun 16 03:57:52 175.60.573.54:5123 Failed password for illegal user guest
Jul 20 18:12:11 271.60.672.56:5815 Failed password for illegal user guest
Aug 15 15:37:35 469.59.261.60:5995 Failed password for illegal user test
Aug 12 03:52:53 852.13.726.60:4947 Illegal user
Jul 1 20:49:22 520.83.293.60:5882 Failed password for illegal user guest
Jun 8 11:01:59 950.41.483.60:4265 Failed password for illegal user test
Oct 6 11:52:46 931.60.665.65:6626 Failed password for admin
Jun 28 00:14:38 559.4.230.60:6675 Failed password for root
Jun 28 16:44:11 694.97.72.60:5110 Failed password for illegal user guest
Aug 15 10:56:52 249.60.323.62:4250 Failed password for illegal user test
Oct 15 23:17:12 968.38.745.60:6524 Failed password for admin
Oct 21 02:45:16 323.63.445.60:6634 Failed password for root
Sep 5 07:53:30 516.60.820.81:6117 Illegal user
Sep 10 17:27:44 792.84.876.60:5026 Failed password for illegal user guest
Sep 28 19:12:16 550.60.596.54:6289 Illegal user
Jul 24 08:05:56 873.60.909.1:5088 Failed password for root
Jun 10 06:25:44 548.60.200.47:4816 Failed password for illegal user guest
Jun 19 21:23:37 485.18.729.60:4911 Failed password for illegal user test
Sep 3 21:07:41 942.91.378.60:4331 Illegal user
Jul 19 11:06:25 401.56.642.60:5921 Failed password for illegal user guest
Aug 15 01:55:39 72.60.979.26:6356 Failed password for illegal user guest
Sep 10 00:18:32 712.69.109.60:4109 Failed password for admin
Aug 21 22:39:04 192.60.649.42:4341 Failed password for root
Jul 30 16:01:18 800.60.400.67:4005 Failed password for illegal user test
Aug 26 13:44:41 390.79.891.60:4702 Illegal user

Top 10 direcciones IP con out-degree value más grande:
IP segment: 60, Out-degree value: 236
IP segment: 13, Out-degree value: 234
IP segment: 5, Out-degree value: 234
IP segment: 75, Out-degree value: 231
IP segment: 11, Out-degree value: 231
IP segment: 53, Out-degree value: 230
IP segment: 25, Out-degree value: 230
IP segment: 18, Out-degree value: 228
IP segment: 16, Out-degree value: 228
IP segment: 42, Out-degree value: 226
```

(Se imprimen varias más entradas)

Reflexión individual:

Marcela De La Rosa: En esta actividad exploramos cómo la elección de utilizar gráficos, específicamente gráficos acíclicos dirigidos (DAG), nos brinda una forma eficiente de organizar las diferentes partes de una dirección IP. Estos diagramas proporcionan una forma clara de representar conexiones entre nodos evitando bucles, por ello, desarrollamos un programa que calculara y mostrara los 10 nodos principales, proporcionando información sobre las partes más conectadas de la red.

En nuestro ejemplo, la función "bootMaster" está configurada para encontrar el número 60, el nodo con el mayor número de conexiones salientes y su información correspondiente. Este

análisis sobre la utilidad de los grafos nos fue útil para descubrir patrones que se pueden dar en el proceso de segmentado. En conclusión, la capacidad que tienen los grafos de representar relaciones y extraer datos de conjuntos destacó en el contexto de las IPs.

Adela Solorio: El uso de grafos para resolver esta situación problema es conveniente porque permiten hacer una representación efectiva de las relaciones entre las direcciones IP, permitiendo visualizar de manera clara y estructurada la relación entre las diferentes partes de la dirección IP; este tipo de estructura de datos también hacen más eficiente hacer búsquedas (como se hizo con el out-degree), lo cual lo vuelve más conveniente; finalmente, los grafos facilitan el encontrar patrones y tendencias en conjuntos de datos (como se hizo con el boot master), lo cual lo vuelve conveniente para situaciones donde se busca entender el comportamiento o identificar irregularidades.

Luz Patricia Hernández: Durante el desarrollo de este proyecto centrado en algoritmos de grafos,tuvimos que realizar muchas pruebas minuciosas del código. Aprendimos a identificar y corregir errores relacionados con la lógica, la estructura de datos y la sintaxis de la programación orientada a objetos, aplicándolos a situaciones complejas.
El enfoque práctico de conceptos de grafos era para analizar registros mediante patrones y relaciones en conjuntos de datos y nos proporcionó una visión más tangible de su potencial en el mundo real. Usar un grafo dirigido acíclico (DAG) resultó fundamental para modelar eficientemente las conexiones entre diferentes partes de las direcciones IP por su naturaleza flexible pero estructurada. En resumen, esta experiencia consolidó mi comprensión de los grafos como herramienta para representar relaciones complejas.

Referencias

Grafos. (n.d.). Universidad de Granada. Recuperado de https://ccia.ugr.es/~jfv/ed1/c++/cdrom4/paginaWeb/grafos.htm