UTILIZAÇÃO DE ALGORITMOS EM GRAFOS PARA RESOLVER UM PROBLEMA DE CENTRALIDADE EM REDES COMPLEXAS

AEDs III

CAIO FERNANDO DIAS
FELIPE DE GODOI CORRÊA
MATHEUS REIS DE LIMA

Unifal MG
Universidade Federal de Alfenas

PROBLEMA DE CENTRALIDADE EM GRAFOS

Determinar a importância dos nós em uma rede complexa, por meio de sua centralidade em um grafo

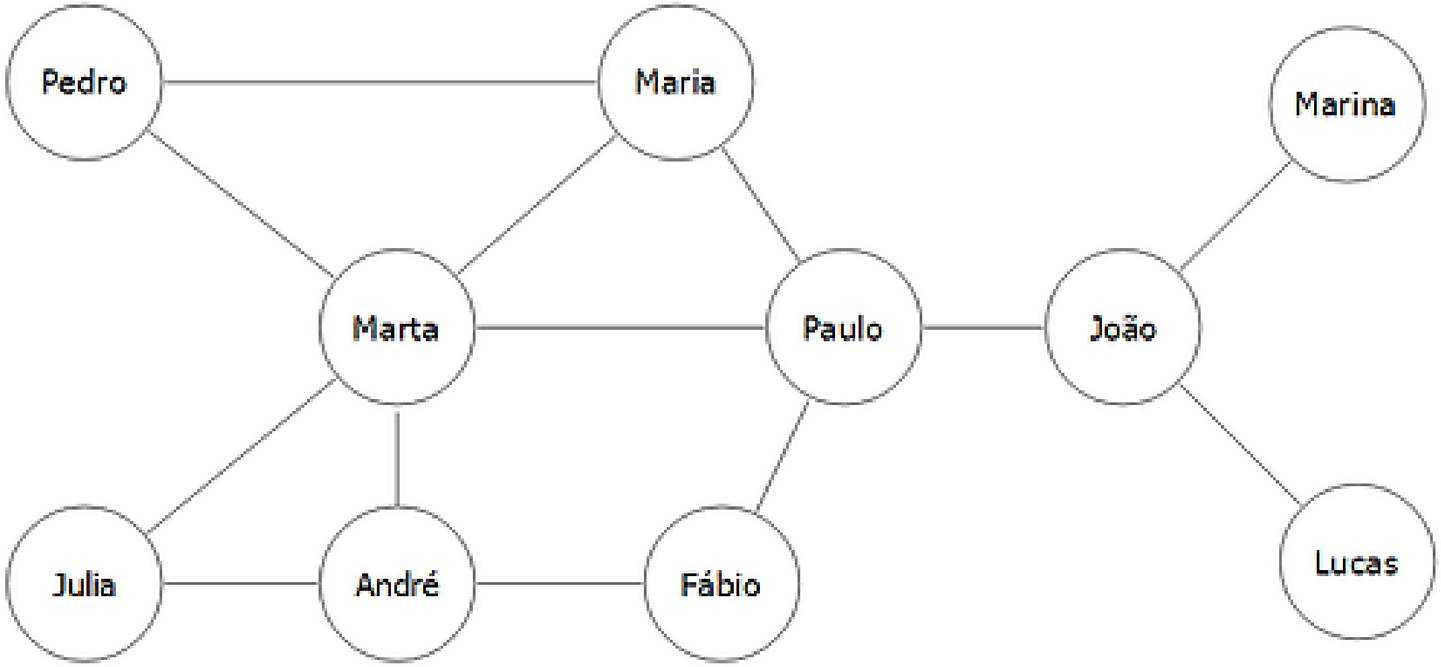


Figura 1: exemplo de grafo

CLOSENESS

Tem como objetivo quantificar os vértices mais influentes de uma rede complexa, que podem ser empregados para propagar informações mais rapidamente, de forma mais eficiente, ou barata para o restante da rede

o valor de *closeness Cs* de um vértice $s \in V$ pode ser definido como:

$V' = V - \{s\}$;
δst é o custo do caminho mínimo entre os vértices *s* e *t* em um grafo *G*;
n é o número de vértices do grafo

$$C_s = \frac{\sum_{t \in V'} \delta_{st}}{n - 1}$$

Figura 2: cálculo closeness

CLOSENESS

Como o nó “Paulo” é central no grafo, seu grau de Closeness Centrality é o maior, enquanto outros nós distantes como “Lucas” e “Marina” possuem graus menores.

Paulo	100,000
Marta	93,333
Maria	82,353
João	77,778
Fábio	73,684
André	70,000
Julia	66,667
Pedro	63,636
Marina	53,846
Lucas	53,846

Figura 3: closeness a partir do nó “Paulo”

ALGORITMOS

Calcula os caminhos mais curtos a partir do vértice de origem:

```
vector<int> Grafo::dijkstra(int src) {
    vector<int> dist(MAX_VERTICES, numeric_limits<int>::max());
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
    pq.push({0, src});
    dist[src] = 0;

    while (!pq.empty()) {
        int u = pq.top().second;
        int d_u = pq.top().first;
        pq.pop();

        if (d_u > dist[u]) continue;

        for (const auto& edge : grafo[u]) {
            int v = edge.first;
            int weight = edge.second;
            if (dist[u] + weight < dist[v]) {
                dist[v] = dist[u] + weight;
                pq.push({dist[v], v});
            }
        }
    }

    return dist;
}
```

Figura 4: primeira função (disjkstra)

Calcula a centralidade de proximidade de um vértice específico:

```
double Grafo::closeness_centrality(int vertex) {
    vector<int> shortest_paths = dijkstra(vertex);
    int total_distance = 0;

    for (int dist : shortest_paths) {
        total_distance += dist;
    }

    double closeness = (MAX_VERTICES - 1) / static_cast<double>(total_distance);
    return closeness;
}
```

Figura 5: segunda função (closeness)

ALGORITMO DE DESENHO DE GRAFOS

Algumas funções e características:

Arrasta as arestas;
Utiliza as entradas dos grafos;
Faz o cálculo dos vértices.

Figura 6: início do algoritmo de desenho de grafos

RESULTADOS

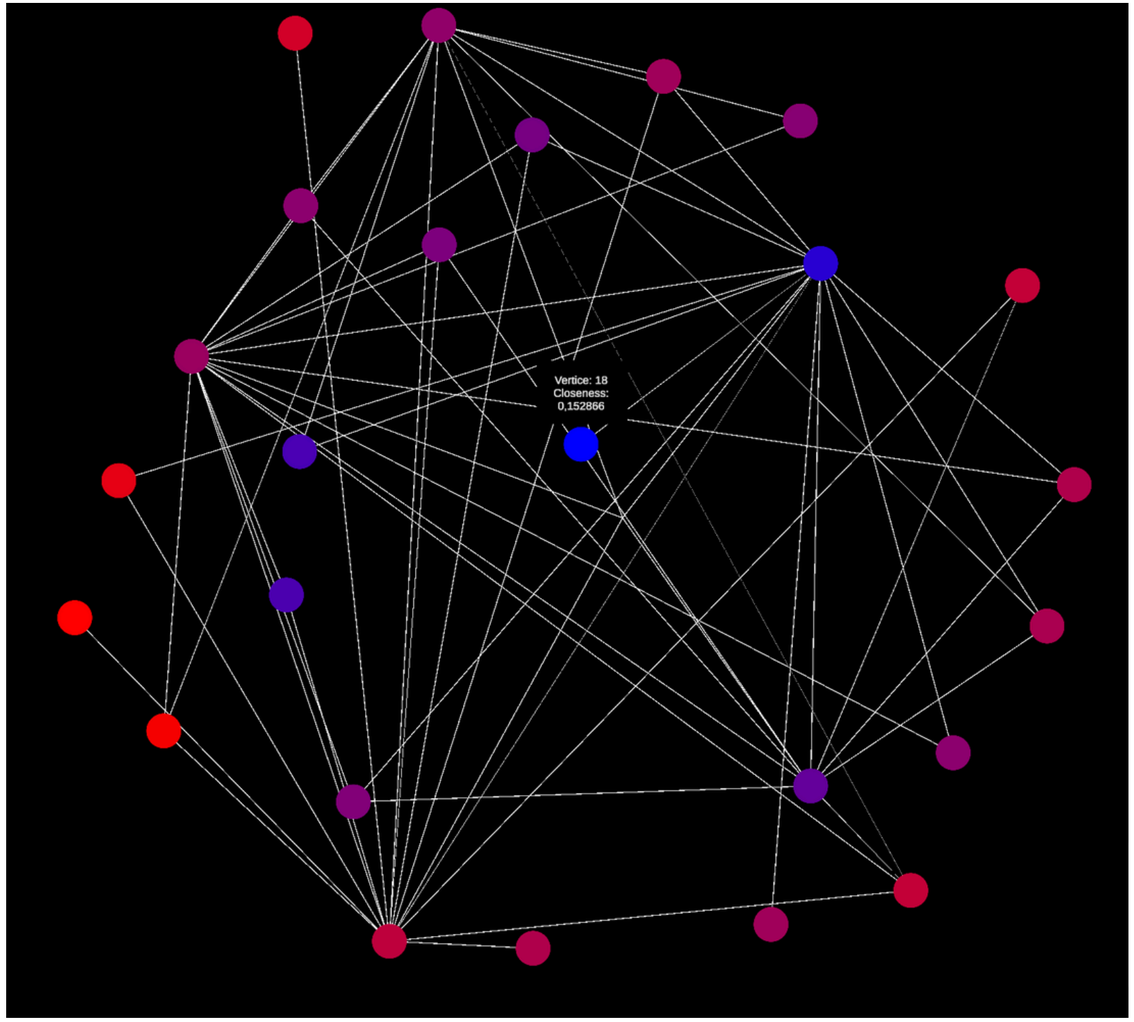


Figura 7: primeiro grafo

RESULTADOS

```
Imprimindo o grafo normalizado 8
(24, 3) (9, 1) (21, 2) (7, 3) (5, 4) (20, 4) - 0.255319
(15, 1) (11, 1) (7, 2) (13, 3) (21, 3) (12, 5) - 0.266667
(13, 3) (21, 4) (20, 1) (7, 3) (9, 5) - 0.24
(14, 5) (9, 5) (10, 3) (13, 3) (23, 5) - 0.177778
(19, 5) (20, 2) (8, 3) (7, 3) (11, 4) - 0.212389
(20, 3) (0, 4) (22, 1) (17, 1) (15, 5) (10, 3) (14, 3) - 0.244898
(15, 1) (10, 2) (17, 1) (20, 2) - 0.27907
(0, 3) (1, 2) (2, 3) (9, 5) (23, 3) (14, 1) (4, 3) (17, 4) - 0.25
(4, 3) (23, 1) (19, 1) (9, 2) (21, 5) - 0.224299
(0, 1) (2, 5) (3, 5) (7, 5) (23, 4) (13, 1) (8, 2) (18, 3) - 0.269663
(23, 5) (3, 3) (6, 2) (5, 3) (11, 1) (13, 1) (15, 1) - 0.3
(1, 1) (4, 4) (10, 1) - 0.258065
(1, 5) (23, 3) (21, 4) (19, 4) (14, 3) (15, 1) - 0.258065
(1, 3) (2, 3) (9, 1) (3, 3) (10, 1) (23, 2) (19, 3) - 0.289157
(3, 5) (7, 1) (12, 3) (24, 5) (5, 3) (20, 4) (19, 3) (23, 4) - 0.237624
(1, 1) (5, 5) (6, 1) (12, 1) (10, 1) - 0.303797
(24, 2) (20, 1) (22, 5) (17, 1) - 0.235294
(5, 1) (7, 4) (16, 1) (6, 1) (21, 4) - 0.269663
(9, 3) (23, 5) - 0.152866
(4, 5) (8, 1) (12, 4) (13, 3) (21, 1) (14, 3) - 0.228571
(2, 1) (4, 2) (5, 3) (14, 4) (16, 1) (0, 4) (6, 2) - 0.25
(0, 2) (1, 3) (2, 4) (12, 4) (17, 4) (19, 1) (8, 5) (23, 3) - 0.237624
(5, 1) (16, 5) (24, 5) - 0.198347
(7, 3) (8, 1) (9, 4) (10, 5) (12, 3) (13, 2) (21, 3) (14, 4) (18, 5) (3, 5) - 0.230769
(0, 3) (14, 5) (16, 2) (22, 5) - 0.196721
```

Figura 8: primeiro grafo normalizado

RESULTADOS

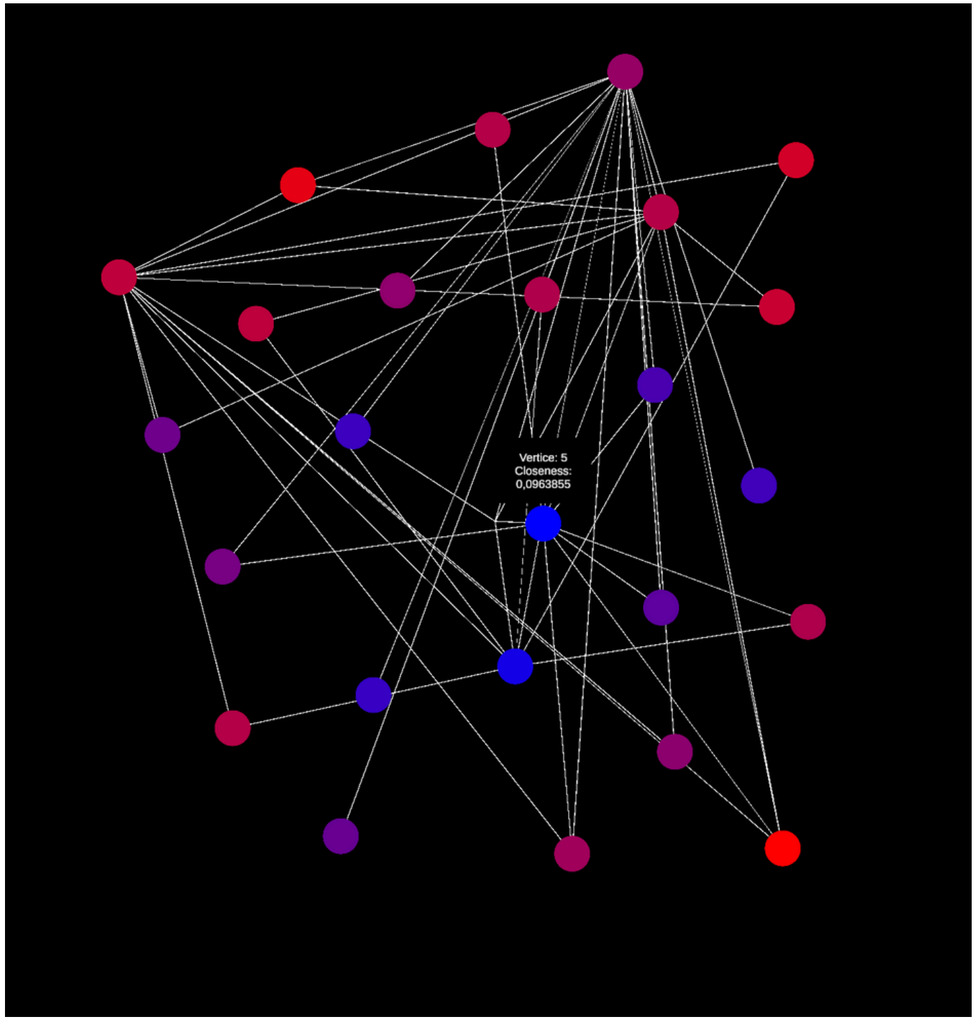


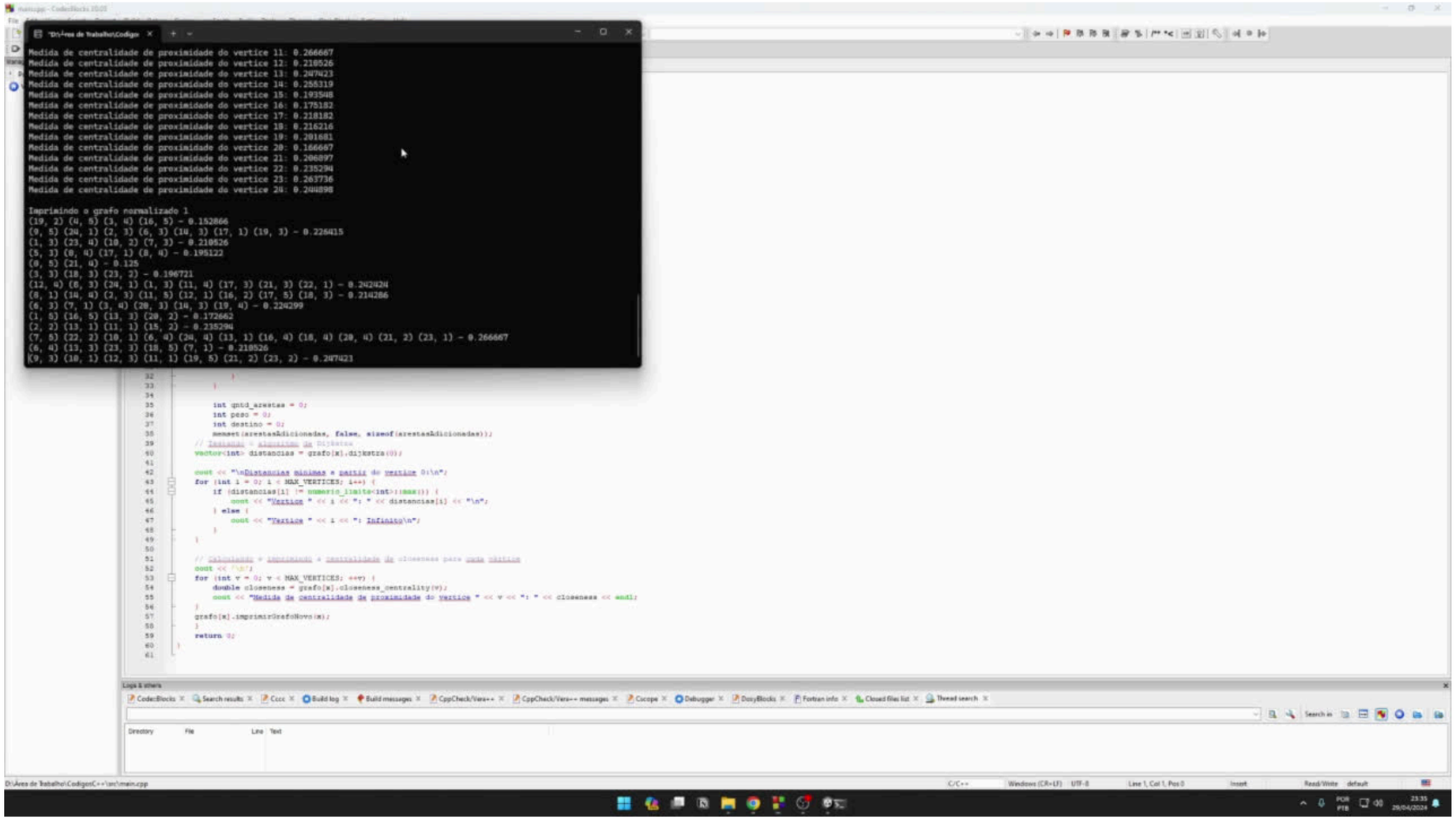
Figura 9: segundo grafo

RESULTADOS

```
Imprimindo o grafo normalizado 7
(19, 4) (2, 3) (6, 1) (21, 3) - 0.192
(22, 4) (5, 5) - 0.108597
(0, 3) (3, 1) (4, 2) (12, 4) (18, 5) (20, 2) - 0.195122
(2, 1) (21, 2) (13, 4) - 0.2
(2, 2) (10, 2) (13, 3) - 0.177778
(1, 5) (20, 5) - 0.0963855
(0, 1) (24, 2) (10, 3) (17, 1) - 0.193548
(23, 3) (8, 4) (10, 2) - 0.156863
(17, 4) (24, 2) (7, 4) - 0.152866
(17, 3) (21, 1) (15, 1) (23, 4) - 0.210526
(4, 2) (6, 3) (15, 1) (17, 3) (7, 2) - 0.198347
(13, 5) (21, 1) - 0.193548
(2, 4) (23, 4) - 0.126316
(11, 5) (21, 5) (3, 4) (4, 3) (14, 4) (16, 2) - 0.147239
(20, 5) (13, 4) (21, 4) - 0.137143
(9, 1) (10, 1) (22, 5) (21, 2) - 0.190476
(22, 4) (13, 2) - 0.129032
(8, 4) (9, 3) (10, 3) (6, 1) (19, 5) (23, 3) - 0.183206
(24, 2) (21, 1) (20, 4) (2, 5) (22, 3) - 0.220183
(0, 4) (17, 5) (24, 4) (22, 4) - 0.131868
(5, 5) (14, 5) (18, 4) (2, 2) - 0.161074
(3, 2) (9, 1) (13, 5) (18, 1) (0, 3) (11, 1) (15, 2) (14, 4) (23, 4) - 0.23301
(1, 4) (15, 5) (16, 4) (18, 3) (19, 4) (24, 3) - 0.173913
(7, 3) (9, 4) (12, 4) (17, 3) (21, 4) (24, 2) - 0.172662
(6, 2) (8, 2) (18, 2) (19, 4) (22, 3) (23, 2) - 0.205128
```

Figura 10: segundo grafo normalizado

RESULTADOS



Vídeo 1: exibição do grafo

REFERÊNCIAS

https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/independent.html

https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/v-cover.html

PERGUNTAS?

OBRIGADO!

CAIO FERNANDO DIAS
FELIPE DE GODOI CORRÊA
MATHEUS REIS DE LIMA