Problema do Clique

Vinicius José Fritzen

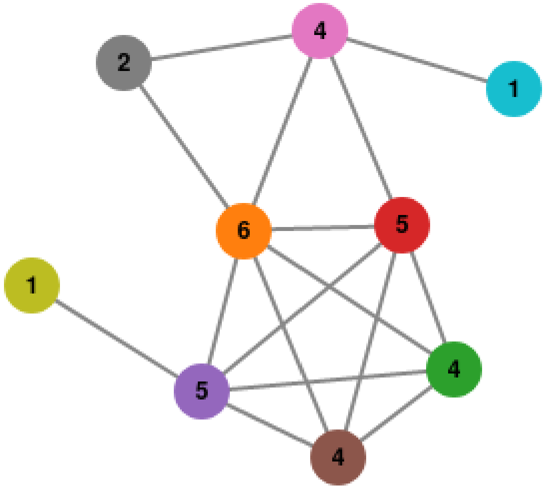
1. Introdução

1.1 O que é um clique?

Clique é um subgrafo completo de um outro grafo. Assim um clique contem apenas vertices que estão conectados a todos os outros vertices.

O tamanho de um clique é o número vértices que participam de um clique.

Um clique é **máximo** se não houver outro clique de tamanho maior neste grafo.



Problema do Clique de Otimização

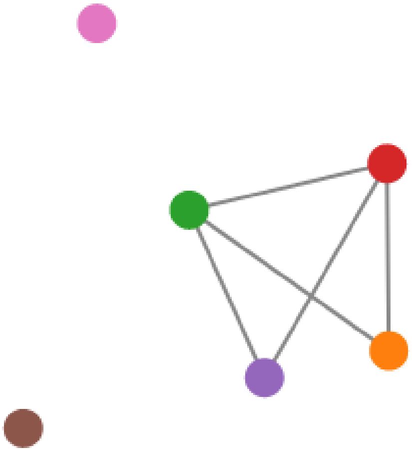
O problema original do clique é um problema de *otimização*.

As suas instâncias são (G) onde:

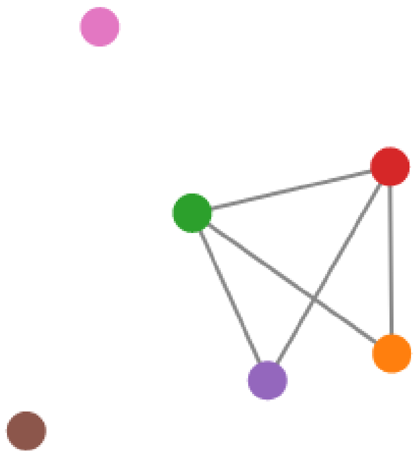
- $G = (V, A)$ é um grafo composto de vertices(V) e arrestas($A = (v_1, v_2)$).

O problema do clique é encontrar um clique máximo de um grafo.

Exemplo

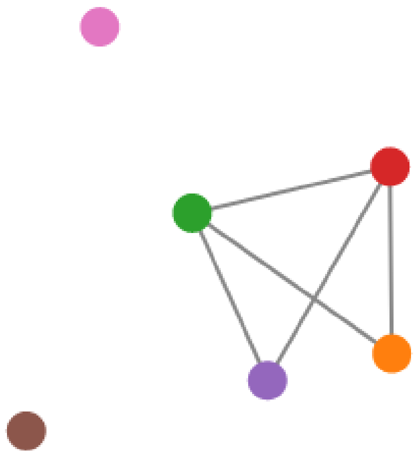


Exemplo

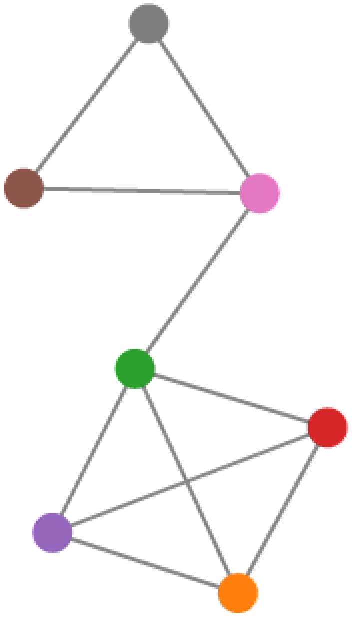


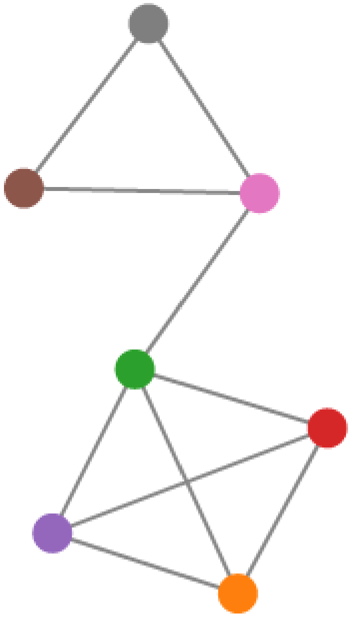
- clique máximo: {verde, vermelho, roxo} ou {verde, vermelho, laranja}

Exemplo

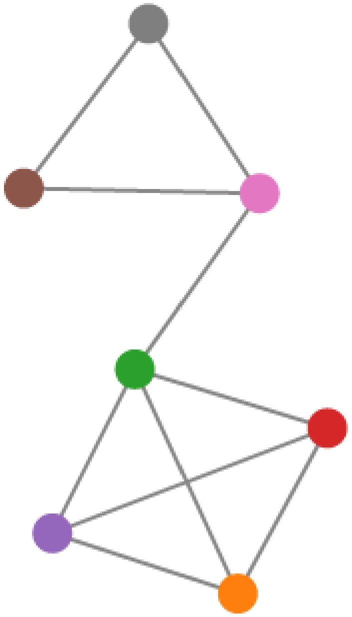


- clique máximo: {verde, vermelho, roxo} ou {verde, vermelho, laranja}
- Tamanho: 3





- clique máximo: {verde, vermelho, roxo, laranja}



- clique máximo: {verde, vermelho, roxo, laranja}
- Tamanho: 4

Problema de Decisão: K-Clique

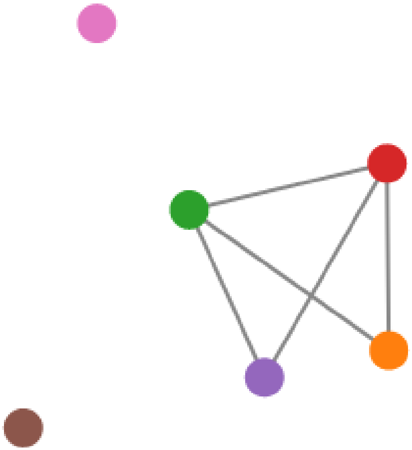
Pelo clique originamente ser um problema de otimização teremos que usar uma outra versão dele para provar sua NP-completude: o *K-Clique*.

O problema do K-Clique tem como instância (G, k) onde:

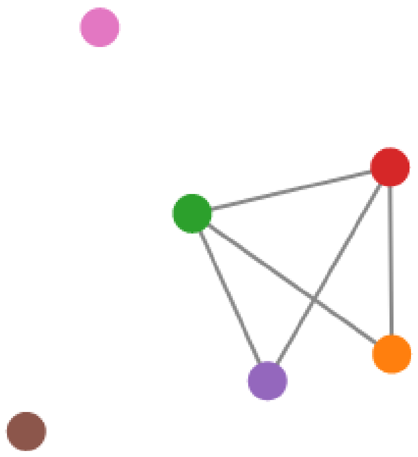
- G é um grafo da mesma forma que no clique.
- $k \in \mathbb{N}$.

O problema consiste em decidir se um grafo contém ao menos um clique de tamanho k ou maior.

Exemplo

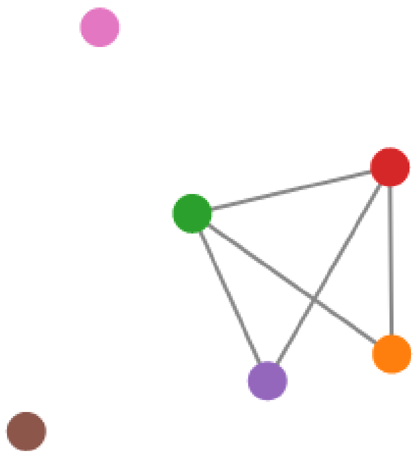


Exemplo



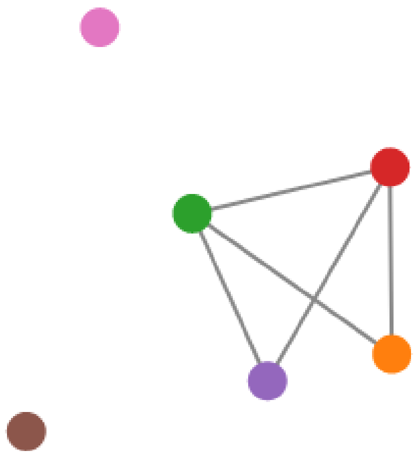
- $k = 1$? Sim

Exemplo



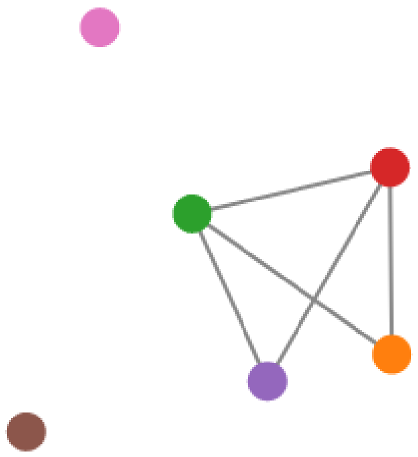
- $k = 1$? Sim
- $k = 2$? Sim

Exemplo



- $k = 1$? Sim
- $k = 2$? Sim
- $k = 3$? Sim

Exemplo



- $k = 1$? Sim
- $k = 2$? Sim
- $k = 3$? Sim
- $k = 4$? Não

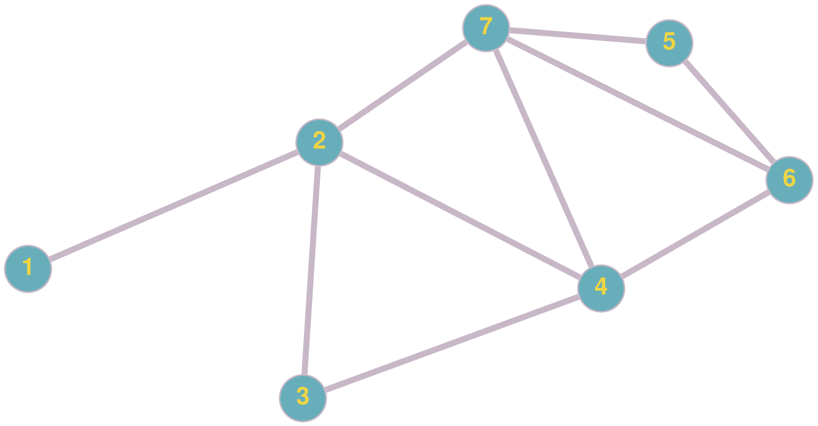
K-Clique é NP:

Para provar que K-clique é NP, mostrarei um algoritmo que válida uma solução se dado um certificado.

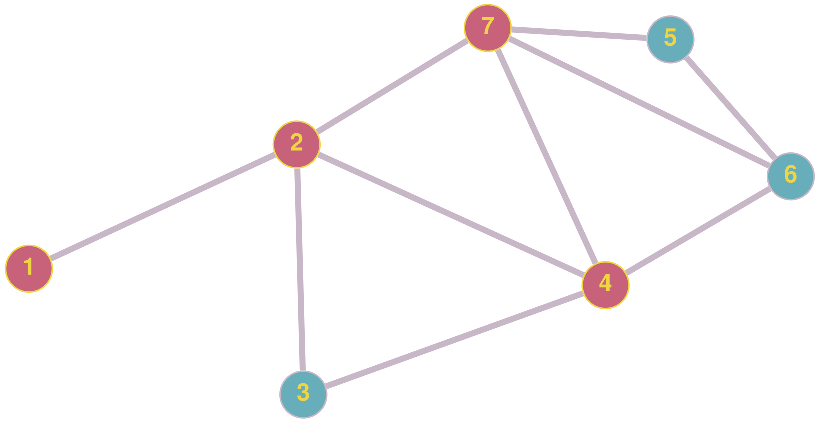
Certificado do K-Clique

O certificado C é o clique encontrado, e esse algoritmo verifica que ele realmente se encontra no grafo G e é maior que k.

k=4 G=



C=



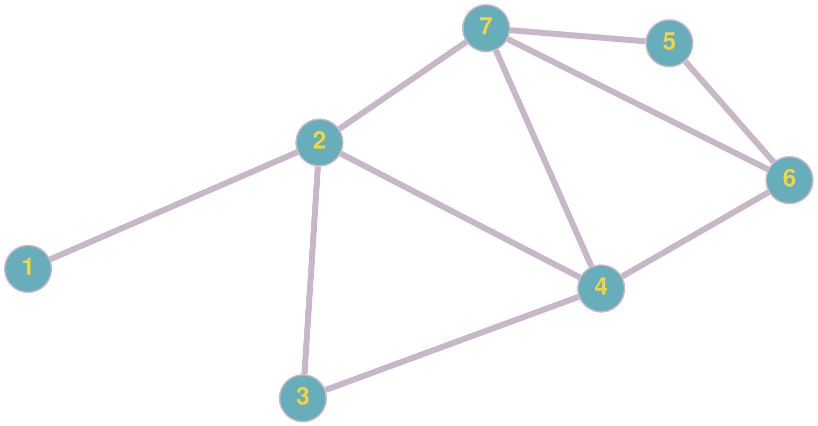
K-Clique é NP:

Para provar que K-clique é NP, mostrarei um algoritmo que válida uma solução se dado um certificado.

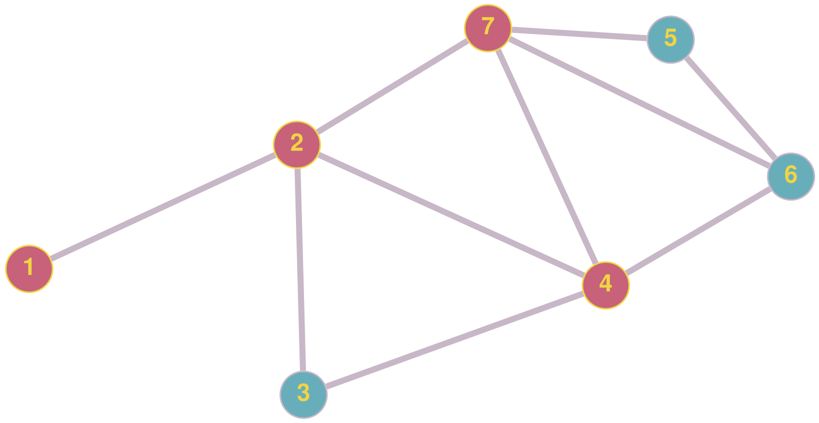
Certificado do K-Clique

O certificado C é o clique encontrado, e esse algoritmo verifica que ele realmente se encontra no grafo G e é maior que k.

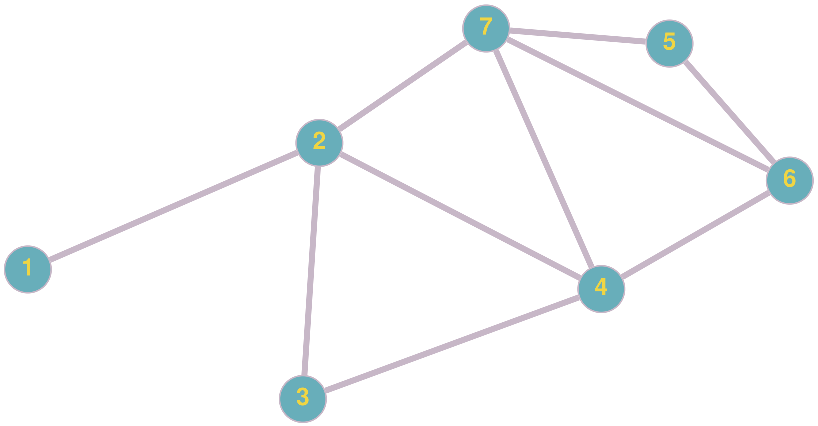
k=4 G=



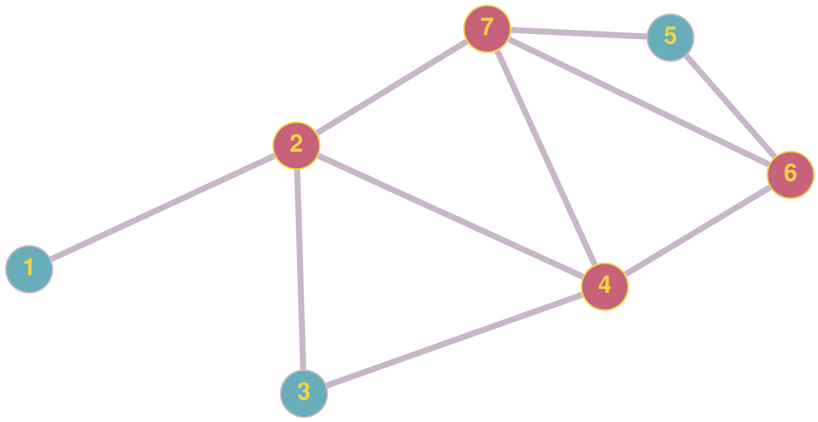
C=



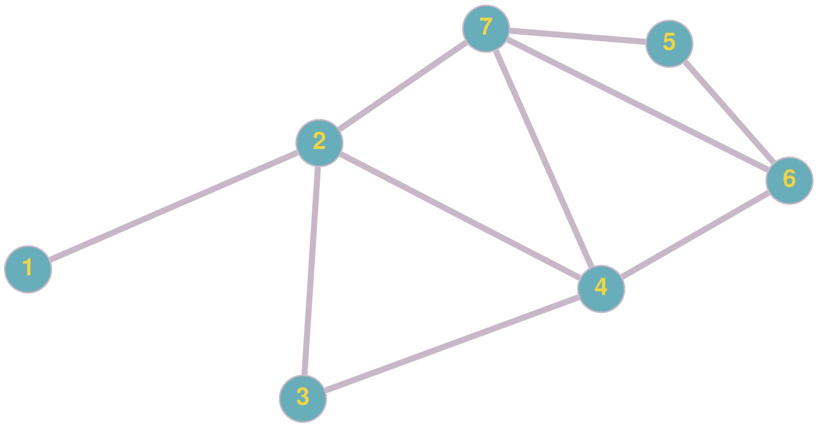
k=4 G=



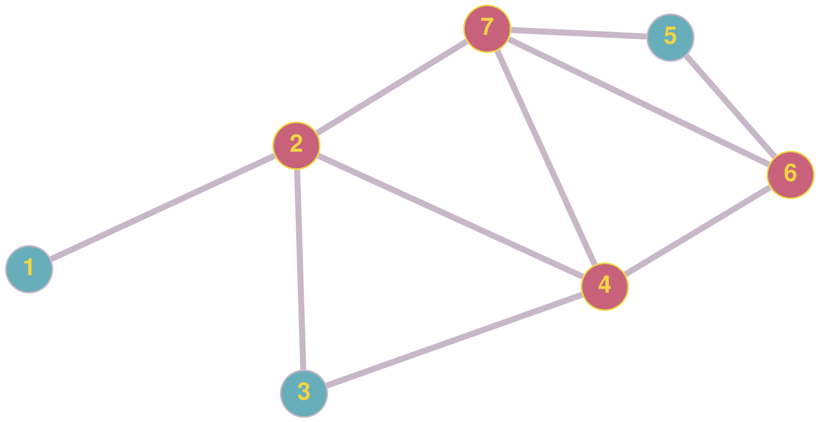
C=



k=4 G=

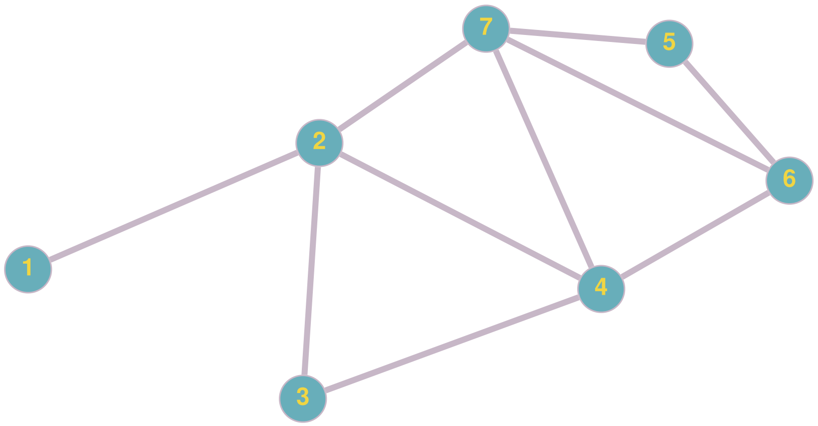


C=

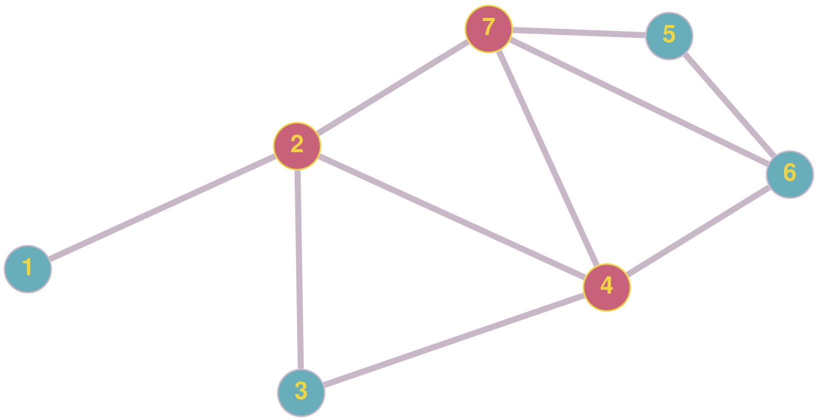


Não é clique, pois não é completo.

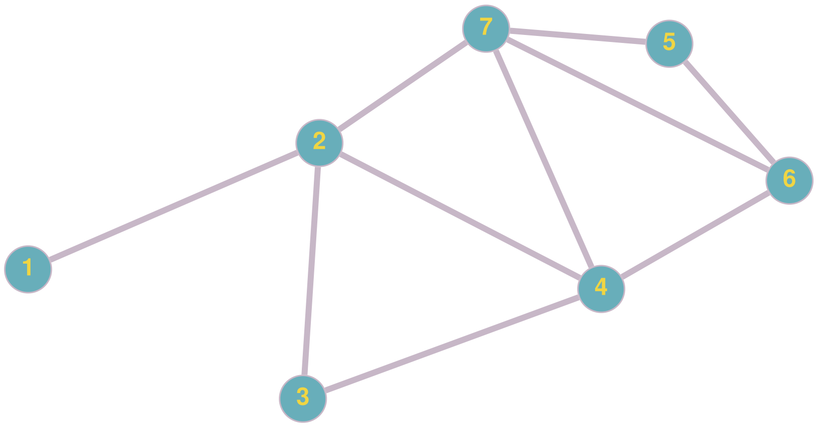
k=3 G=



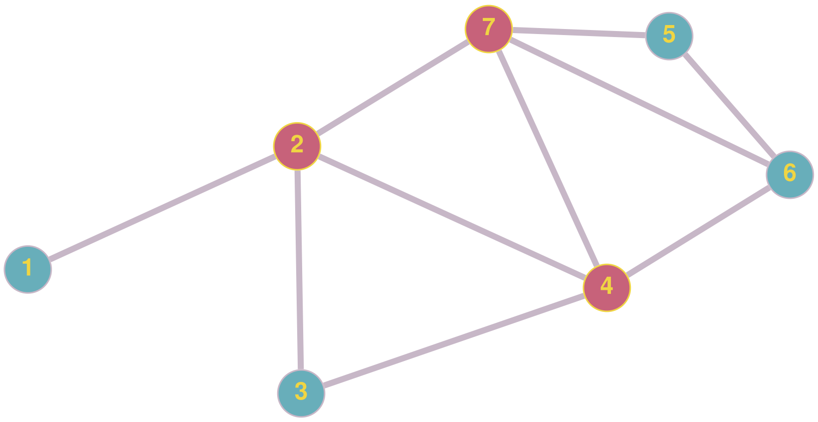
C=



k=3 G=



C=



É um clique de tamanho 3, então é válido.

Algoritmo de verificação

```
fn verifica_validade(G: Grafo, k: i32, C: Grafo) -> bool {
    if C.vert_count < k {return false}

    for (vert in C.V) {
        if (!G.V.contains(vert)) return false;

        for (other in C.V) {
            if ( other == vert) continue;
            if (!G.A.contains(vert,other)) return false;
        }
    }

    return true;
}
```

Algoritmo de verificação

```
fn verifica_validade(G: Grafo, k: i32, C: Grafo) -> bool {
    if C.vert_count < k {return false}

    for (vert in C.V) {
        if (!G.V.contains(vert)) return false;

        for (other in C.V) {
            if ( other == vert) continue;
            if (!G.A.contains(vert,other)) return false;
        }
    }

    return true;
}
```

O algoritmo verifica se o clique é maior que k, se todos os vertices estão no grafo e se todos os vertices estão conectados.

$$O(1) + \sum_{i=1}^{n}(\sum_{j=1}^{n}(O(1) + O(n))) = O(n^3)$$

K-Clique é NP-dificil

Prova que K-Clique é NP-dificil usando redução do CNF-3SAT.

O CNF-3SAT é um problema de decisão onde a instância é uma formula CNF de 3SAT.

O problema consiste em decidir se uma formula CNF de 3SAT é satisfazível ou não. Ou seja se existe uma atribuição de valores verdadeiros para as variáveis que faz a formula ser verdadeira.

Exemplo:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \tag{1}$$

Essa formula é satisfazível, pois podemos escolher $x_1 = x_2 = 1$.

Observe que ela é satisfazível, pois é possível encontrar uma variável em cada clausula que é verdadeira e de forma que nenhum literal seja negado e afirmado ao mesmo tempo.

Exemplo de formula CNF-3SAT não satisfazível

$$(x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1) \tag{2}$$

Independe de qual valor atribuirmos a x_1, a formula sempre será falsa.

Usaremos uma redução do CNF-3SAT para K-Clique para provar que K-Clique é NP-difícil.

Redução do CNF-3SAT para K-Clique

Considerando uma formula CNF de 3SAT(ϕ) de n clausulas, na qual cada clausula é uma disjunção de 3 literais, assim, cada clausula C_i é uma disjunção de 3 literais L^i_1, L^i_2, L^i_3.

$$C_i = (L_i^1 \vee L_i^2 \vee L_i^3)$$

$$\phi = \bigwedge_{i=1}^{n} C_i$$

Constuiremos um grafo $G = (V, A)$ onde:

- Para cada clausula C_r da formula CNF, teremos uma tripla de vértices v^r_1, v^r_2, v^r_3.
- Inserimos uma arresta entre dois vértices v^r_1, v^r_2 se, e somente se ambas as afirmativas forem verdadeiras:
  - v^r_i e v^s_j são vértices de clausulas diferentes. Ou seja, $r \neq s$;
  - L^r_i e L^s_j não são literais complementares. Ou seja, $L^r_i \neq \neg L^s_j$.

Seja $G = (V, A)$ o grafo construido, então:

- Se G contem um clique de tamanho n, então ϕ é satisfazível;
- Se G não contém um clique de tamanho n, então ϕ não é satisfazível.

Exemplo:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4) \tag{3}$$

```
fn sat_to_kclique(sat: Sat) {
    let grafo = Grafo();// Contem as informações de qual era a clausula original do vertice
    let clausulasDeOrigem = Mapa<Vertice, int>();
    for (clausula in sat.clausulas) {
        let v1 = grafo.add_vertice(clusula.l1);
        clausulasDeOrigem.add(v1, clausula.n);
        let v2 = grafo.add_vertice(clausula.l2);
        clausulasDeOrigem.add(v2, clausula.n);
        grafo.add_vertice(clausula.l3);
        clausulasDeOrigem.add(v3, clausula.n);
    }
    for (let vertice in grafo) {
        for (let outro_vertice in grafo) {
            if (vertice == outro_vertice) continue;
            if (
                clausulasDeOrigem.deClausulasDiferentes(vertice, outroVertice)
                && vertice != not(outro_vertice)
            ) {
                grafo.add_aresta(vertice, outro_vertice);
            }
        }
    }
    return grafo;
}
```

Análise da complexidade do Algoritmo

```
fn sat_to_kclique(sat: Sat) {
    let grafo = Grafo();// Contem as informações de qual era a clausula original do vertice
    let clausulasDeOrigem = Mapa<Vertice, int>();
    for (clausula in sat.clausulas) { // n *
        let v1 = grafo.add_vertice(clusula.l1); // O(1)
        clausulasDeOrigem.add(v1, clausula.n);// O(1)
        let v2 = grafo.add_vertice(clausula.l2);// O(1)
        clausulasDeOrigem.add(v2, clausula.n);// O(1)
        grafo.add_vertice(clausula.l3);// O(1)
        clausulasDeOrigem.add(v3, clausula.n);// O(1)
    }
    for (let vertice in grafo) { // n *
        for (let outro_vertice in grafo) { // n *
            if (vertice == outro_vertice) continue;// O(1)
            if (
                clausulasDeOrigem.deClausulasDiferentes(vertice, outroVertice) // O(1)
                && vertice != not(outro_vertice)// O(1)
            ) {
                grafo.add_aresta(vertice, outro_vertice);// O(1)
            }
        }
    }
```

Análise da complexidade do Algoritmo

\begin{align} $$ \sum_{i=1}^{n} O(1) + \sum_{i=1}^{n} \sum_{j=1}^{n} O(1) = O(n^2)

\end{align}

Conclusão

- O algoritmo de redução do CNF-3SAT para K-Clique é polinomial;
- O algoritmo de verificação de clique em um grafo é polinomial;
- Portanto, ***o K-Clique é NP-completo.***

Referências

Algorimos: Teoria e Prática, 3ª Edição, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein

Agradeço pela atenção.

Os slides estão disponíveis em:

https://vini84200.github.io/clique-slidev