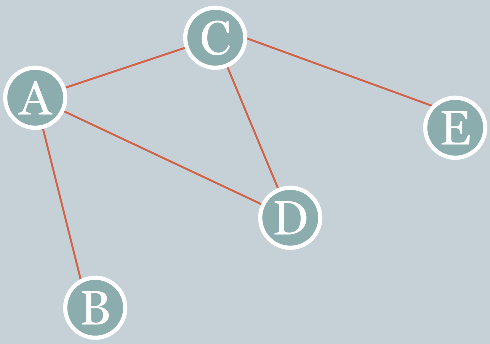
Problema de Decisão Clique

Raffael da Silva Nagel
194048

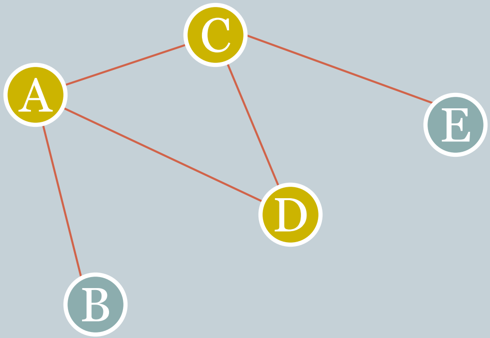
Caracterização do problema

- Grafo
  - Conjunto de pontos (*vértices*) ligados por retas (*arestas*)



Caracterização do problema

- Clique
  - Subconjunto de vértices em um grafo não orientado, tal que cada dois vértices estão conectados por uma aresta.



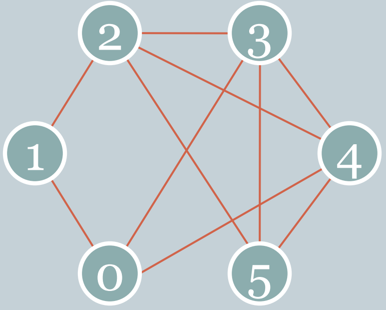
Caracterização do problema

- Problema
  - Dado um grafo G não orientado e uma constante k maior que zero, determinar se existe em G um clique de tamanho maior ou igual a k.

Caracterização do problema

- Problema
  - Dado um grafo G não orientado e uma constante k maior que zero, determinar se existe em G um clique de tamanho maior ou igual a k.

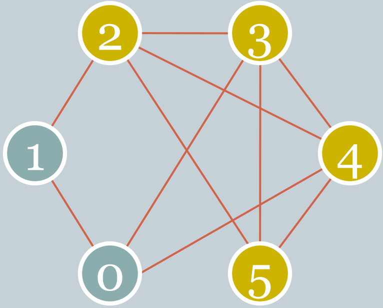
  - **Exemplo**
  - O grafo possui um clique
  de tamanho 4?



Caracterização do problema

- Problema
  - Dado um grafo G não orientado e uma constante k maior que zero, determinar se existe em G um clique de tamanho maior ou igual a k.

  - **Exemplo**
  - O grafo possui um clique de tamanho 4?

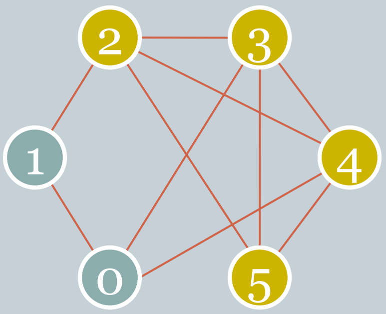


Caracterização do problema

- Problema
  - Dado um grafo G não orientado e uma constante k maior que zero, determinar se existe em G um clique de tamanho maior ou igual a k.

  - **Exemplo**
  - O grafo possui um clique de tamanho 4?

Sim!



Algoritmo de Verificação

- Certificado
  - Um conjunto de vértices Y

Entradas

Certificado

Constante K

Grafo G (contexto)

Algoritmo de Verificação

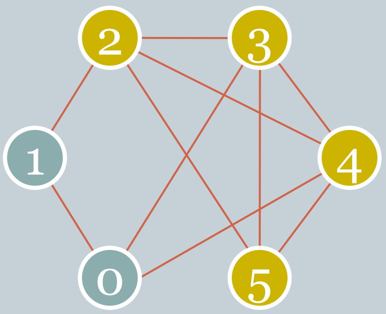
- *Verifica_Clique(G, k, Y){*

```
Se   |Y| == k   então
     Para i de 1 até k faça
          SE  Y[i] não Є G então
               retorna 0
     fim para
     Para i de 1 até k faça
          Para j = 1 até k faça
               Se i!=j   E não existe aresta entre Y[i] e Y[j]
                    retorna 0
          fim para
     fim para
     retorna 1
senão
     retorna 0
}
```

Algoritmo de Verificação

- Exemplo
  - Y = {2,3,4,5}
  - k = 4

G



Verifica_Clique(G, k, Y) == 1

Complexidade do Algoritmo de Verificação

- *Verifica_Clique(G, k, Y){*

```
        Se   |Y| == k   então
            Para i de 1 até k faça
                SE  Y[i] não € G então
                    retorna 0
            fim para
O(n²)       Para i de 1 até k faça
                Para j = 1 até k faça
                    Se i!=j   E não existe aresta entre Y[i] e Y[j]
                        retorna 0
                fim para
            fim para
            retorna 1
        senão
            retorna 0
}
```

Complexidade Polinomial >> NP

Redução

- Problema usado: *Satisfabilidade Booleana (SAT)*

- *Caracterização do problema*
  - Dada uma fórmula booleana, determinar a existência de certa valoração para as variáveis que satisfaça esta fórmula.
  - ***Exemplo***

$$\alpha = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$$

SIM ($x=1$, $y=1$, $z=0$)

Algoritmo de Redução

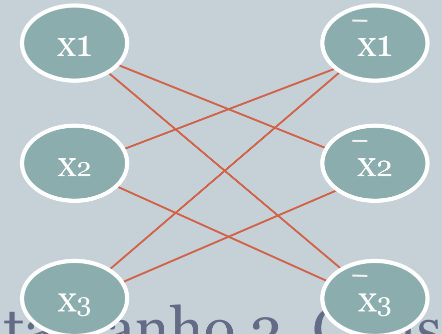
- Dada uma expressão booleana α deverá gerar um grafo *G e uma constante k, assim a expressão* α *é satisfazível sse existe em G um k-clique.*

- Lógica de Redução
  - $V = \{(x, i) \mid \text{onde } x \text{ é o literal e } i \text{ a cláusula}\}$
  - $E = \{(\,(x, i), (y, j)\,) \mid i \neq j \wedge (\,(x, i) \neq \neg(y, j)\,)\}$

  Onde *V* são os vértices de G e *E* suas arestas.

Algoritmo de Redução

- Exemplo
  - $F = (x1 \lor x2 \lor x3) \land (\neg x1 \lor \neg x2 \lor \neg x3)$



  - O gráfico contem 6 cliques de tamanho 2. Considerando o clique formado por { x1, ¬x2 } e assumindo seu valores como verdadeiros (x1 = 1, x2 = 0) satisfazemos F.

Algoritmo de Redução

***Entrada**: Expressão booleana B, composta por suas C cláusulas e seus l literais, os quais possuem informação sobre a cláusula da qual são originários.*

```
Reduction_Sat_Clique(B){
        Grafo G;
        v = l;
        para i de 0 até n faça
              para j de 0 até n faça
                    se( (l[i].C != l[j].C) e ( l[j]l != ¬l[i] )) então
                    //testa se literais fazem parte de cláusulas diferentes
                    //e se não são um a negação do outro
                          E = aresta(v[i],v[j])
              fim para
         fim para
        G = {V,E}
        retorna G
}
```

Complexidade do Algoritmo de Redução

- *Reduction_Sat_Clique(B){*

```
        Grafo G;
        v = l;
        para i de 0 até n faça
              para j de 0 até n faça
                    se( (l[i].C != l[j].C) e ( l[j]l != ¬l[i] )) então
                    //testa se literais fazem parte de cláusulas diferentes
                    //e se não são um a negação do outro
                          E = aresta(v[i],v[j])
              fim para
         fim para
        G = {V,E}
        retorna G
}
```

$O(n^2)$

Complexidade Polinomial >> NP - completo