Problema do Clique

Felipe Derkian ,Bruno Cesar

Universidade Federal do Roraima (UFRR)
Departamento de Ciência da Computação(DCC)

felipekian@yahoo.com.br , brunocclaudino@gmail.com

Abstract. The objective of this study is to perform an analysis and comparison of the performance of the exact version of the click problem and the approximate version of this problem, together with the complexity of the two algorithms.

Resumo. *O objetivo deste estudo consiste em realizar uma análise e comparação do desempenho da versão exata do problema do clique e da versão aproximada deste problema,juntamente com a complexidade dos dois algoritmos.*

1. Problema do clique

Entrada: Dado um grafo G(V, E), sendo não-direcionado e sem peso nas arestas.

Saída: retorna o maior clique encontrado no grafo. Caso exista mais de um clique com mesmo tamanho máximo, só retornará o primeiro encontrado.

2. Configuração Do Ambiente de Testes

- Notebook Acer aspire E 15 573
- 8Gb de RAM 1333mhz
- Hd 500Gb
- core i5 5200u
- Sistema Operacional Linux Ubuntu 18.04 LTS com Gnome-shell

3. Algoritmo Gerador de Grafo

```c
#include <stdio.h>
#include <stdlib.h>
void criar_Grafo();
void espera();
int main(){
    printf("\n\t\tGerador de Grafos Aleatorio\n\n");criar_Grafo();
    printf("\tGrafo gerado...\n\n");espera();return 0;}
void espera(){getchar();getchar();}
void criar_Grafo(){
    FILE * f = fopen("ArquivoGrafo.txt","w");
    if(f == NULL){
        printf("Nao foi possivel gravar no arquivo....\n\n");espera();
        exit(1);    }
    int vertices , arestas;
    printf("Informe o numero de Vertices, (0 ate numero): ");
    scanf("%d",&vertices);
    printf("\n\n\tgerando....\n\n");
    fprintf(f,"%d\n",vertices);
    int i, j;
    for(i=0 ; i < vertices+1 ; i++){
        int k = rand() % (vertices+1);
        for(j=0 ; j<k ; j++){
            int h;
            do{
                h = rand() % (vertices+1);}while(i==h);
            fprintf(f , "%d %d\n",i,h);}
        }
    fclose(f);}
```

4. Versão Exata do Problema do Clique

Para a implementação da versão exata do problema do clique utilizamos a técnica de forca bruta e testamos todas as possibilidades, inclusive refazendo os cliques que já foram encontrados anteriormente com tais vértices. Segue abaixo as funções utilizadas no algoritmo.

4.1 Função Main

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#define TRUE  1
#define FALSE 0
#define TAM 30001
struct grafo{
    int tamVertices;
    int vetVisitado[TAM];
    int Matriz[TAM][TAM];
}Grafo;
struct cliqueMax{
    int Principal;
    int tamClique;
    int vetClique[TAM];
}CliqueMax;
int main(){
    apresentacao();lerArquivo();
    if(Grafo.tamVertices < 21)
            imprimeMatriz();
    else
        printf("\nNao vai imprimir tabela.. total de vertices > 20\n\n");
    clock_t ci , cf;ci = clock();
    verificarCliqueMaximo();cf = clock();
    double tempoTotal = (double) (cf - ci) / (double) CLOCKS_PER_SEC;
    imprimeCliqueMax();
    printf("\nTempo total Clique: %lf seg\n\n",tempoTotal);
    printf("Press enter..");espera();
    return 0;
}
```

4.2 Função Verifica Clique Maximo

```
89  void verificarCliqueMaximo(){
90      limpaClique();
91      int  vertice , i , j , tam_adj;tam_adj = 0;int vet_adj[TAM]={0};
92      printf("\t\t...Procurando Clique no Grafo...\n\n");
93      for(vertice=0 ; vertice <= Grafo.tamVertices ; vertice++){
94          printf("Verificando vertice %d.\n",vertice);
95      tam_adj=0;
96          for(i=0 ; i <= Grafo.tamVertices ; i++){
97              if(Grafo.Matriz[ vertice ][ i ] == 1){vet_adj[ tam_adj] = i; tam_adj++;}
98          }
99          for(i=0 ; i<tam_adj-1 ; i++){
100             if(vet_adj[i] == -1) continue;
101             for(j = i+1 ; j < tam_adj ; j++){
102                 if(vet_adj[j] != -1 && Grafo.Matriz[ vet_adj[i] ][ vet_adj[j] ] != 1){
103                     vet_adj[j] = -1;}
104             }
105         }int tot = 1;
106         for(i = 0 ; i < tam_adj ; i++){
107             if( vet_adj[i] != -1){
108                 tot++;}
109         }
110         if( tot > CliqueMax.tamClique){
111             CliqueMax.tamClique = tot;
112             CliqueMax.Principal = vertice;int k=0;
113             for(i=0 ; i<tam_adj ; i++){
114                 if(vet_adj[i] != -1){
115                     CliqueMax.vetClique[k++] = vet_adj[i];}
116             }
117         }
118     }
119 }
```

4.3 Função Ler Arquivo Grafo

```
83  void lerArquivo(){
84
85
86
87      limparGrafo();
88
89      FILE *f = fopen("ArquivoGrafo.txt","r");
90
91      if(f == NULL){
92          system("clear");
93          printf("\n\n\tNao foi possivel acessar o arquivo...\n\n");
94          espera();
95          exit(1);
96      }
97
98      printf("\n\n\t...Lendo Arquivo com Grafo...\n\n");
99
100     int total;
101
102     fscanf(f , "%d\n",&total);
103
104     Grafo.tamVertices = total;
105     while(!feof(f)){
106         int v,a;
107         fscanf(f,"%d %d\n",&v,&a);
108         Grafo.Matriz[v][a] = 1;
109         Grafo.Matriz[a][v] = 1;
110     }
111     fclose(f);
112 }
113
```

4.4 Função Imprime Clique Max

```
85  void apresentacao(){
86
87      printf("\n\t\t...Problema do Clique Maximo...\n\n\tComponentes\n\t\tFelipe Derkian,\n\t\tBruno Cesar.\n\n");
88
89  }
90
91  void imprimeCliqueMax(){
92
93      int i , j;
94
95      printf("\n\n\t\t...Impressao das Arestas...\n\n");
96
97      if(CliqueMax.tamClique > 1)
98          printf("\tClique maximo no Grafo = %d.\n\n",CliqueMax.tamClique);
99      else
100         printf("\tClique maximo no Grafo = %d.\n\n",CliqueMax.tamClique-1);
101
102
103     for(i=0 ; i<CliqueMax.tamClique -1 ; i++){
104         printf("\t\t%d <----> %d\n",CliqueMax.Principal,CliqueMax.vetClique[i]);
105     }
106
107     for(i=0 ; i<CliqueMax.tamClique -2 ; i++){
108         for(j=i+1 ; j<CliqueMax.tamClique-1 ; j++){
109             printf("\t\t%d <----> %d\n",CliqueMax.vetClique[i], CliqueMax.vetClique[j]);
110         }
111     }
112
113 }
114
```

4.5 Função Imprime Matriz de Adjacência

```
29  void imprimeMatriz(){
30
31      printf("\n\n\t...Matriz de Adjacencia...\n\n");
32
33      int i , j;
34
35      printf("\tTotal de Vertices = %d\n\n",Grafo.tamVertices);
36
37      printf("\t");
38      for(i=0 ; i<=Grafo.tamVertices ; i++){
39          printf("%d\t",i);
40      }
41      pulaLinha();pulaLinha();
42      for(i=0 ; i<=Grafo.tamVertices ; i++){
43          printf("%d\t",i);
44          for(j=0 ; j<=Grafo.tamVertices ; j++){
45              printf("%d\t",Grafo.Matriz[i][j]);
46          }
47          pulaLinha();
48      }
49      pulaLinha();
50
51      printf("\nPress para sair..");
52      espera();
53      pulaLinha();
54      pulaLinha();
55
56
57  }
```

4.6 Outras Funções utilizadas

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#define TRUE  1
#define FALSE 0
#define TAM 30001
struct grafo{
    int tamVertices;
    int vetVisitado[TAM];
    int Matriz[TAM][TAM];}Grafo;
struct cliqueMax{
    int Principal;
    int tamClique;
    int vetClique[TAM];}CliqueMax;
void espera(){getchar();}
void pulaLinha(){printf("\n");}
void limparGrafo(){int i,j;
    for(i=0;i<TAM;i++){
        Grafo.vetVisitado[i]=0;
        Grafo.tamVertices=0;
        for(j=0;j<TAM;j++){
            Grafo.Matriz[i][j]=0;}
    }
}
void limpaClique(){
    int i;CliqueMax.Principal=-1;CliqueMax.tamClique=0;
    for(i=0;i<TAM;i++){
        CliqueMax.vetClique[i]=-1;}
}
void apresentacao(){printf("\n\t\t...Problema do Clique Maximo...\n\n\tComponentes\n\t\tFelipe Derkian,\n\t\tBruno Cesar.\n\n");}
```

5. Versão Aproximada do Problema do Clique

Para a implementação da versão aproximada, adicionamos um vetor para marcar os vértices que já pertencem a algum clique. Com isso, não precisamos reprocessar cliques já encontrados usando a técnica de Backtracking que é uma técnica de algoritmo que representa um refinamento da busca por força bruta.

5.1 Função Main

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#define TRUE  1
#define FALSE 0
#define PRETO 10
#define TAM 30001
struct grafo{
    int tamVertices;
    int vetVisitado[TAM];
    int Matriz[TAM][TAM];
}Grafo;
struct cliqueMax{
    int Principal;
    int tamClique;
    int vetClique[TAM];
}CliqueMax;
int main(){
    apresentacao();lerArquivo();
    if(Grafo.tamVertices < 21)
            imprimeMatriz();
    else
        printf("\nNao vai imprimir tabela.. total de vertices > 20\n\n");
    clock_t ci , cf;ci = clock();
    verificarCliqueMaximo();cf = clock();
    double tempoTotal = (double) (cf - ci) / (double) CLOCKS_PER_SEC;
    imprimeCliqueMax();
    printf("\nTempo total Clique: %lf seg\n\n",tempoTotal);
    printf("Press enter..");espera();return 0;
}
```

5.2 Função Verifica Clique Maximo

```
void verificarCliqueMaximo(){
    limpaClique();
    int  vertice , i , j , tam_adj;tam_adj = 0;int vet_adj[TAM]={0};
    printf("\t\t...Procurando Clique no Grafo...\n\n");
    for(vertice=0 ; vertice <= Grafo.tamVertices ; vertice++){
        if(Grafo.vetVisitado[vertice] == PRETO) continue;
        printf("Verificando vertice %d.\n",vertice);tam_adj=0;
        for(i=0 ; i <= Grafo.tamVertices ; i++){
            if(Grafo.Matriz[ vertice ][ i ] == 1){vet_adj[ tam_adj] = i; tam_adj++;}
        }
        for(i=0 ; i<tam_adj-1 ; i++){if(vet_adj[i] == -1) continue;
            for(j = i+1 ; j < tam_adj ; j++){
                if(vet_adj[j] != -1 && Grafo.Matriz[ vet_adj[i] ][ vet_adj[j] ] != 1){vet_adj[j] = -1;}
            }
        }int tot = 1;
        for(i = 0 ; i < tam_adj ; i++){
            if( vet_adj[i] != -1){
                tot++;}
        }
        if( tot > CliqueMax.tamClique){CliqueMax.tamClique = tot;CliqueMax.Principal = vertice;
            Grafo.vetVisitado[vertice] = PRETO;int k=0;
            for(i=0 ; i<tam_adj ; i++){
                if(vet_adj[i] != -1){CliqueMax.vetClique[k++] = vet_adj[i];
                    Grafo.vetVisitado[ vet_adj[i] ] = PRETO;}
            }
        }else{Grafo.vetVisitado[vertice] = PRETO;int y;
            for(y=0 ; y<tam_adj ; y++){
                if(vet_adj[y] != -1)
                    Grafo.vetVisitado[ vet_adj[y] ] = PRETO;}
        }
    }
}
```

5.3 Função Ler Arquivo Grafo

```
void lerArquivo(){
    limparGrafo();
    FILE *f = fopen("ArquivoGrafo.txt","r");
    if(f == NULL){
        system("clear");
        printf("\n\n\tNao foi possivel acessar o arquivo...\n\n");
        espera();
        exit(1);
    }
    printf("\n\n\t...Lendo Arquivo com Grafo...\n\n");
    int total;
    fscanf(f , "%d\n",&total);
    Grafo.tamVertices = total;
    while(!feof(f)){
        int v,a;
        fscanf(f,"%d %d\n",&v,&a);
        Grafo.Matriz[v][a] = 1;
        Grafo.Matriz[a][v] = 1;
    }
    fclose(f);
}
```

5.4 Função Imprime Clique Maximo

```
151 void imprimeCliqueMax(){
152 
153     int i , j;
154 
155     printf("\n\n\t\t...Impressao das Arestas...\n\n");
156 
157     if(CliqueMax.tamClique > 1)
158         printf("\tClique maximo no Grafo = %d.\n\n",CliqueMax.tamClique);
159     else
160         printf("\tClique maximo no Grafo = %d.\n\n",CliqueMax.tamClique - 1);
161 
162     for(i=0 ; i<CliqueMax.tamClique -1 ; i++){
163         printf("\t\t%d <----> %d\n",CliqueMax.Principal,CliqueMax.vetClique[i]);
164     }
165 
166     for(i=0 ; i<CliqueMax.tamClique -2 ; i++){
167         for(j=i+1 ; j<CliqueMax.tamClique-1 ; j++){
168             printf("\t\t%d <----> %d\n",CliqueMax.vetClique[i], CliqueMax.vetClique[j]);
169         }
170     }
171 
172 }
173 
```

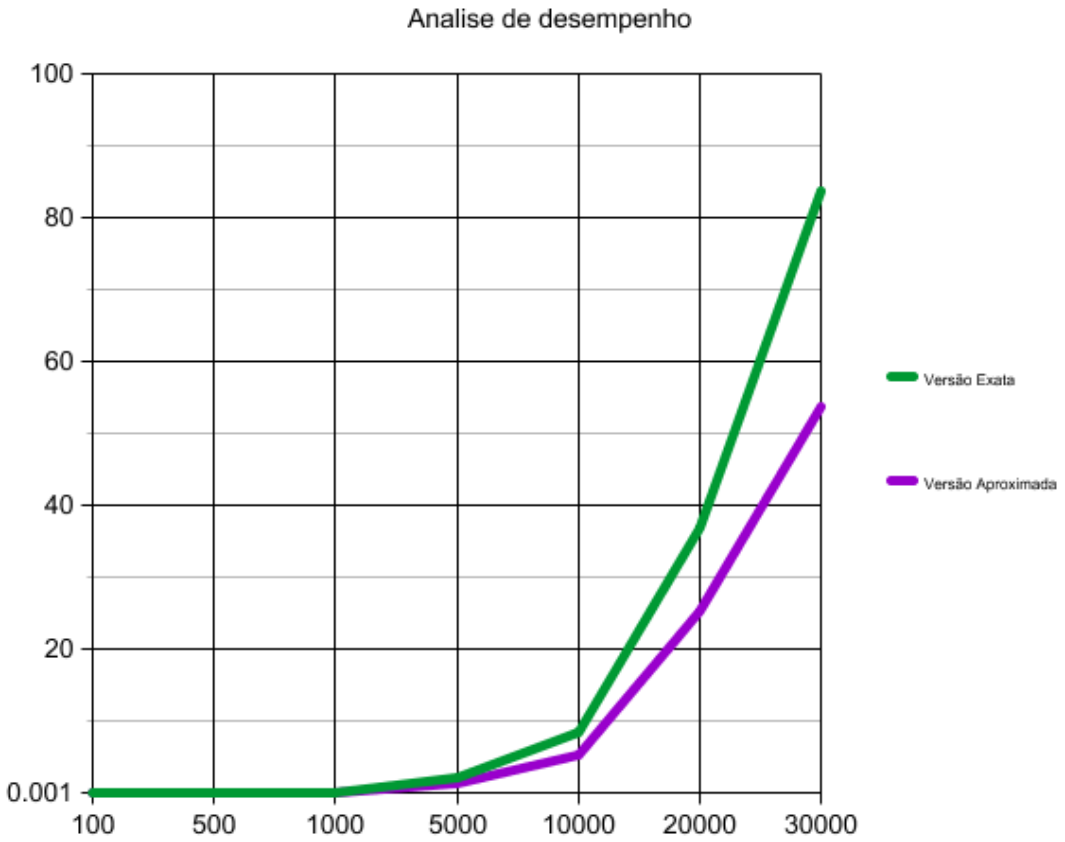
5.5 Função Imprime Matriz

```
24  void imprimeMatriz(){
25  
26      printf("\n\n\t...Matriz de Adjacencia...\n\n");
27  
28      int i , j;
29  
30      printf("\tTotal de Vertices = %d\n\n",Grafo.tamVertices);
31  
32      printf("\t");
33      for(i=0 ; i<=Grafo.tamVertices ; i++){
34          printf("%d\t",i);
35      }
36      pulaLinha();pulaLinha();
37      for(i=0 ; i<=Grafo.tamVertices ; i++){
38          printf("%d\t",i);
39          for(j=0 ; j<=Grafo.tamVertices ; j++){
40              printf("%d\t",Grafo.Matriz[i][j]);
41          }
42          pulaLinha();
43      }
44      pulaLinha();
45  
46      printf("\nPress para sair..");
47      espera();
48      pulaLinha();
49      pulaLinha();
50  
51  
52  }
```

5.6 Outras Funções Utilizadas

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#define TRUE  1
#define FALSE 0
#define PRETO 10
#define TAM 30001
struct grafo{
    int tamVertices;int vetVisitado[TAM];int Matriz[TAM][TAM];}Grafo;
struct cliqueMax{
    int Principal;int tamClique;int vetClique[TAM];}CliqueMax;
void espera(){getchar();}
void pulaLinha(){printf("\n");}
void limparGrafo(){
    int i,j;
    for(i=0;i<TAM;i++){Grafo.vetVisitado[i]=0;Grafo.tamVertices=0;
        for(j=0;j<TAM;j++){Grafo.Matriz[i][j]=0;}
    }
}
void limpaClique(){
    int i;CliqueMax.Principal=-1;CliqueMax.tamClique=0;
    for(i=0;i<TAM;i++){
        CliqueMax.vetClique[i]=-1;}
}
void apresentacao(){
    printf("\n\t\t...Problema do Clique Maximo...\n\n\tComponentes\n\t\tFelipe Derkian,\n\t\tBruno Cesar.\n\n");}
void zeraVisitado(){
    int i;for(i=0 ; i< Grafo.tamVertices+10 ; i++){Grafo.vetVisitado[i] = 0;}
}
```

6. Gráfico dos Testes



7. Dados dos Testes

Nº Vertices 0 ate N	T. FB em segundos	Tam_clique FB	T. Ver. aprox. em segundos	Tam_clique VA
30000	83.565222	43	53.689522	43
20000	36.766159	42	23.150493	42
10000	8.500564	37	5.285930	37
5000	2.095569	33	1.270586	32
1000	0.072731	25	0.042489	25
500	0.016765	21	0.009429	21
100	0.001039	15	0.000678	14

8. Complexidade Versão Aproximada

$$\sum_{v=0}^{n}\left(\sum_{i=0}^{n}+\sum_{i=0}^{m-1}\left(\sum_{j=i+1}^{m}\right)+\sum_{i=0}^{m}\right)$$

$$\sum_{v=0}^{n}\left(n+\sum_{i=0}^{m-1}\left(m-(i+1)-1\right)+m\right)$$

$$\sum_{v=0}^{n}\left(\sum_{i=0}^{m-1}\left(m-i\right)+m\right)$$

$$\sum_{v=0}^{n}\left(n+\left(m\sum_{i=1}^{m}-\sum_{i=1}^{m}i\right)+m\right)$$

$$\sum_{v=0}^{n}\left(n+m^{2}-\frac{m(m+1)}{2}+m\right)$$

$$\sum_{v=0}^{n}\left(n+m^{2}-\frac{m^{2}+m}{2}+m\right)$$

$$n\sum_{v=0}^{n}+m^{2}\sum_{v=0}^{n}-\frac{m^{2}+m}{2}\sum_{v=0}^{n}+m\sum_{v=0}^{n}$$

$$n^{2}+m^{2}n-\frac{m^{2}n+mn}{2}+mn$$

$$Complexidade=O\left(m^{2}n\right)$$

9. Complexidade Versão Exata

$$\sum_{v=0}^{n}\left(\sum_{i=0}^{n}+\sum_{i=0}^{m-1}\left(\sum_{j=i+1}^{m}\right)+\sum_{i=0}^{m}+\sum_{i=0}^{m}\right)$$

$$\sum_{v=0}^{n}\left(n+\sum_{i=0}^{m-1}\left(m-(i+1)-1\right)+2m\right)$$

$$\sum_{v=0}^{n}\left(\sum_{i=0}^{m-1}\left(m-i\right)+2m\right)$$

$$\sum_{v=0}^{n}\left(n+\left(m\sum_{i=1}^{m}-\sum_{i=1}^{m}i\right)+2m\right)$$

$$\sum_{v=0}^{n}\left(n+m^{2}-\frac{m(m+1)}{2}+2m\right)$$

$$n\sum_{v=0}^{n}+m^{2}\sum_{v=0}^{n}-\frac{m^{2}+m}{2}\sum_{v=0}^{n}+2m\sum_{v=0}^{n}$$

$$n^{2}+m^{2}n-\frac{m^{2}n+mn}{2}+2mn$$

$$Complexidade=O\left(m^{2}n\right)$$

10. Referencia

Teoria da Computação. Clique de um Grafo. Artigo de Alexandre Renato Rodrigues de Souza. Programa de Pós Graduação em Computação. ubiq.inf.ufpel.edu.br/arrsouza/lib/exe/fetch.php?media=clique_de_um_grafo.pdf

Algoritmos/ Thomas H. Cormen… [et al]; [tradução Arlete Simille Marques]. - Rio de Janeiro: Elsevier. il. Tradução de: Introduction to algorithms, 3rd ed.