1 Algoritmo de Coloração Aproximada

1.1 Descrição do Problema

Implementar o algoritmo de coloração aproximada visto em aula. A entrada deste programa, com os nós e as arestas, deve ser lida de arquivo. A saída deve ser impressa na tela do terminal, contendo o número aproximado de cores necessárias, conforme o algoritmo. Ao final da atividade, anexar apenas o código fonte desenvolvido no Google Classroom.

1.2 Arquivo de Entrada

As informações para a montagem do grafo, obrigatoriamente, devem ser lidas a partir de um arquivo do disco, em formato texto, com a seguinte formatação:

- na primeira linha deve-se informar o número de nós;
- na segunda linha o número de arestas;
- a partir da terceira linha deve-se informar as arestas na forma: nó de origem (um caractere alfanumérico), um espaço, e nó de destino (um caractere alfanumérico). Considere que o grafo não é orientado, isto é, para toda aresta lida do arquivo também há uma aresta em sentido contrario. Para desenvolver este algoritmo deverá utilizar a representação de grafos por lista de adjacências conforme o fonte anexado e exibido em aula (anexo a esta atividade)

```
5
8
A B
A C
A D
B C
B E
C D
C E
D E
```

Este arquivo irá produzir o grafo dado pela Figura 1.

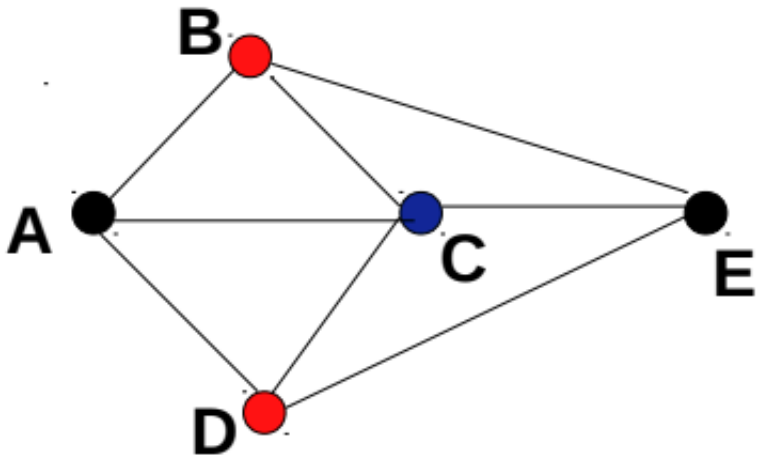

Figura 1: Exemplo de grafo. Destaque para a solução já informada com 3 cores

2 Dica

2.1 Manipulação de arquivos em C

2.1.1 Abertura de um arquivo (fopen)

Como fazer que um arquivo em disco, que tem o seu nome externo ao programa, seja associado a algum objeto ou variável que é interna ou conhecida do nosso programa? Antes que possa ser lido ou gravado um arquivo deve ser aberto pela função de biblioteca `fopen`. A função `fopen` associa o nome de uma arquivo externo ao programa, pro exemplo `entrada.txt`, a uma variável do programa. Isto é feito da seguinte maneira:

```
FILE *arq_entrada;

...
arq_entrada = fopen("entrada.txt","r");
...
```

O primeiro parâmetro da função fopen é uma cadeia de caracteres (string, seja lá o que for isto) contendo o nome do arquivo externo. Já o segundo parâmetro, que também é uma cadeia de caracteres, indica o modo como se pretende usar o arquivo: leitura ("r"), gravação ("w"), anexação ("a"). Por exemplo o comando

```
FILE *arq_entrada;

...
arq_entrada = fopen("entrada.txt","r");
if (arq_entrada == NULL)
  {
    printf("Um erro ocorreu ao tentar abrir o arquivo 'entrada.txt'.\n");
  }
...
```

associa a variável `arq_entrada` do nosso programa o arquivo em disco de nome `entrada.txt`. O arquivo `entrada.txt` deve estar na mesma pasta (diretório, folder) que o nosso programa, digamos, o `ep3.exe`. O arquivo é aberto para leitura ("r"de read). Se ocorrer um erro, como tentar associar `arq_entrada` a um arquivo que não existe, `fopen` retorna o valor representado pela constante `NULL`. Se um arquivo que não existe no disco for aberto para escrita, ele é criado. Por exemplo, o seguinte trecho de código cria um arquivo de nome `saida.txt`:

```
FILE *arq_saida;

...
arq_saida = fopen("saida.txt","w");
if (arq_saida == NULL)
  {
    printf("Um erro ocorreu ao tentar criar o arquivo 'saida.txt'\n");
  }
...
```

O arquivo `saida.txt` é criado na mesma pasta/diretório que o arquivo que contém o nosso programa, suponha que seja o `ep3.exe`. Se o arquivo já existe ele é destruído e um novo arquivo com o mesmo nome é criado. O arquivo é aberto para escrita ("w"de write). Sempre que um erro ocorre durante a abertura de um arquivo o valor devolvido pela função `fopen` é `NULL`.

2.1.2 Leitura de um arquivo (fscanf)

O próximo passo é saber como ler os dados de uma arquivo. Considere o arquivo `entrada.txt` que foi associado à variável `arq_entrada` do nosso programa. O comando:

```
FILE *arq_entrada;

...
arq_entrada = fopen("entrada.txt","r");
...

fscanf(arq_entrada,"%d", &num);
...
```

lê um número inteiro do arquivo `entrada.txt` da mesma maneira que

```
. . .
scanf("%d", &num);
. . .
```

lê um número digitado através do teclado. Na verdade, o comando

```
scanf("%d",&num);
```

é uma abreviatura do comando

```
fscanf(stdin,"%d", &num);
```

onde `stdin` é a entrada padrão, que no nosso caso, é o teclado.

2.1.3 Escrita em arquivo (fprintf)

Para escrever-se em um arquivo de nome `saida.txt` pode-se usar os seguintes comandos

```
FILE *arq_saida;

. . .
arq_saida = fopen("saida.txt","w");
. . .

fprintf(arq_saida,"Copia\n");
fprintf(arq_saida,"%d %d\n", n, m);
. . .
```

O primeiro fprintf escreve o texto "Copia\n"no arquivo saida.txt e o segundo fprintf escreve o conteúdo da variável `n`, um espaço (`" "`), o conteúdo da variável `m` e um salto de linha (`"\n"`) para mudar de linha.

Veja um exemplo um pouco mais elaborado:

```
FILE *arq_saida;
int n;
int m;

. . .
arq_saida = fopen("saida.txt","w");
if (arq_saida == NULL)
  {
    printf("Um erro ocorreu ao tentar criar o arquivo 'saida.txt'.\n");
  }
. . .
fprintf(arq_saida,"Copia\n");
fprintf(arq_saida,"%d %d\n", n, m);
fprintf(arq_saida,"n*m=%d\n", n*m);
. . .
```

De maneira semelhante ao que ocorre com o `scanf`, o comando

```
printf("%d\n", valmax);
```

é uma abreviatura do comando

```
fprintf(stdout,"%d\n", valmax);
```

onde `stdout` é a saída padrão, que no nosso caso, é a tela do monitor.

2.1.4 Fechamento de um arquivo (fclose)

A função `fclose` encerra a associação estabelecida pelo programa entre uma variável e o nome externo do arquivo. O `fclose` também libera os recursos do sistema que controlam a manipulação do arquivo em disco. A chamada abaixo fecha o arquivo associado à variável `arq_saida`.

```
fclose(arq_saida);
```

Exemplo

A seguir está um exemplo de programa que copia números de um arquivo de nome original.dat para um arquivo de nome copia.dat. O primeiro número do arquivo original.dat indica o número n de números no arquivo.

```c
/*
 * Programa que que faz uma copia de um arquivo
 * de nome original.dat para um arquivo de nome copia.dat. O
 * arquivo copia.dat ficara no mesmo diretorio do arquivo
 * original.dat.
 *
 */
#include <stdio.h>

int main()
{
  FILE *arq_o; /* associado ao arquivo original */
  FILE *arq_c; /* associado ao arquivo copia */
  int n; /* numero de elementos do arquivo original */
  float x;
  int i;

  /* 1. abra arquivo original.dat para leitura */
  arq_o = fopen("original.dat","r");
  if (arq_o == NULL)
    {
      printf("Erro na abertura do arquivo original.dat.\n");
      system("pause"); /* para WINDOWNS */
      exit(-1);  /* abandona a execucao do programa */
    }

  /* 2. abra arquivo copia.dat para escrita */
  arq_c = fopen("copia.dat","w");
  if (arq_c == NULL)
    {
      printf("Erro na abertura do arquivo copia.dat.\n");
      system("pause"); /* para WINDOWNS */
      exit(-1); /* abandona a execucao do programa */
    }

  /* 3. leia o tamanho da sequencia no arquivo original.dat */
  fscanf(arq_o,"%d", &n);

  /* 4. escreva o numero de elementos no arquivo copia.dat */
  fprintf(arq_c,"%d\n", n);

  /* 5. leia os numeros do arquivo original.dat e escreva em copia.dat*/
  for (i = 0; i < n; i++)
    {
      fscanf(arq_o,"%f", &x);
      fprintf(arq_c,"%f ", x);
      if (i == 5)
        { /* apenas cinco numeros por linha de arq_c */
          fprintf(arq_c,"\n");
        }
    }

 /* 6. feche o arquivo original.dat */
  fclose(arq_o);

  /* 7. feche o arquivo copia.dat */
  fclose(arq_c);
  return 0;
}
```