Introdução à Teoria dos Grafos
Emparelhamentos Máximos em Grafos Bipartidos

Bacharelado em Ciência da Computação, DCT–UFMS, 6/6/2005

Entrega em 04/07/2005

Resumo

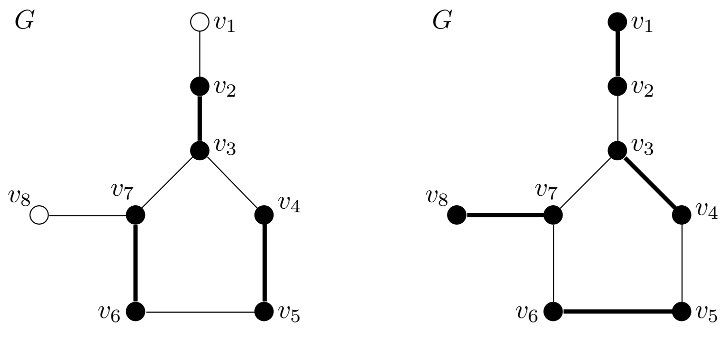
Quando estudamos emparalhementos e fatorações em grafos, mencionamos a necessidade e a existência de algoritmos para encontrar emparelhamentos de cardinalidade máxima ou de valor máximo em grafos bipartidos ou em grafos gerais. Neste trabalho você deve implementar um algoritmo que recebe como entrada um grafo bipartido e encontra um emparelhamento de cardinalidade máxima nesse grafo. Sua implementação deve ser feita na linguagem C padrão (*ANSI C*).

1 Descrição do Problema

O problema que queremos solucionar tem a seguinte descrição:

> **Problema** EMGB(G): *dado um grafo bipartido G, com partição V_1 e V_2 de V_G, encontrar um emparelhamento M em G de cardinalidade máxima.*

Seja M um emparelhamento em um grafo G e suponha que P é um caminho aumentador com respeito a M. Denote por M' as arestas de P que pertencem a M e seja $M'' = E_P \setminus M'$. Faça $M_1 = (M \setminus M') \cup M''$ e observe que M_1 é um emparelhamento em G com cardinalidade $|M| + 1$. Dizemos que M_1 é obtido **aumentando** M **sobre** P. Observe que todo vértice que não é emparelhado com respeito a M_1 também não é emparelhado com respeito a M. Veja a figura 1 para um exemplo.



(a) (b)

Figura 1: Emparelhamentos em um grafo G, onde arestas e vértices grifados são emparelhados. **(a)** Um emparelhamento M. **(b)** Um emparelhamento M_1 obtido pelo aumento de M sobre um caminho aumentador P: $v_1, v_2, v_2, v_4, v_5, v_6, v_7, v_8$.

O resultado a seguir fornece a base para algoritmos que encontram emparelhamentos de cardinalidade máxima em grafos bipartidos e em grafos gerais.

TEOREMA 1.1 *Seja M um emparelhamento em um grafo G que não é máximo e seja v um vértice não emparelhado com respeito a M. Denote por M_1 o emparelhamento obtido aumentando M sobre algum caminho aumentador. Se G contém um caminho aumentador com respeito a M_1 que tem v como vértice inicial, então G contém um caminho aumentador com respeito a M que tem v como vértice inicial.*

PROVA. Suponha o contrário, isto é, que G contém um caminho aumentador com respeito a M_1 com início no vértice v, mas G não tem um caminho aumentador com respeito a M que tem início no vértice v. Seja $P\colon v = u_1, u_2, \ldots, u_n$ um caminho aumentador com respeito a M_1. Então, por suposição, P não é um caminho aumentador com respeito a M. Assim, P contém uma aresta que pertence a M_1 mas não a M. Seja i o menor índice tal que $u_{2i}u_{2i+1} \in M_1 \setminus M$. Então u_{2i} é emparelhado com respeito a M; caso contrário, $P'\colon v = u_1, u_2, \ldots, u_{2i}$ é um caminho aumentador com respeito a M com início em v.

Suponha que M_1 é obtido aumentando M sobre um caminho aumentador $Q\colon v_1, v_2, \ldots, v_k$ com respeito a M. Então, $u_{2i}u_{2i+1} \in E_Q \setminus M$ e u_{2i} é incidente com uma aresta e de Q que pertence a M. Suponha que $u_{2i} = v_j$ com $1 < j < k$. Então $e = v_{j-1}v_j$ ou $e = v_jv_{j+1}$.

Considere inicialmente que $e = v_{j-1}v_j$. O caminho $Q'\colon v_1, v_2, \ldots, v_{j-1}, v_j$ é um caminho alternante com respeito a M que contém exatamente um vértice não emparelhado, o vértice v_1. As arestas não emparelhadas de Q' com respeito a M tornam-se arestas emparelhadas com respeito a M_1 quando aumentamos M sobre Q. Segue que Q' e P' têm somente o vértice $v_j = u_{2i}$ em comum. Mas $v = u_1, u_2, \ldots, u_{2i}, v_{j-1}, v_{j-2}, \ldots, v_1$ é um caminho aumentador com respeito a M, o que contradiz a hipótese.

No caso em que $e = v_jv_{j+1}$ podemos mostrar de forma similar que $v = u_1, u_2, \ldots, u_{2i}, v_{j+1}, \ldots, v_k$ é um caminho aumentador com respeito a M. Isto novamente produz uma contradição. Assim, M_1 não contém um caminho aumentador que tem início em v.

❑

Uma conseqüência imediata do teorema anterior é a seguinte.

COROLÁRIO 1.2 *Seja M um emparelhamento em um grafo G. Suponha que $M = M_1, M_2, \ldots, M_k$ é uma seqüência de emparelhamentos em G tal que M_i é obtido aumentando M_{i-1} sobre algum caminho aumentador, para $2 \leqslant i \leqslant k$. Suponha que o vértice v não é emparelhado com respeito a M e não existe um caminho aumentador com respeito a M com início em v. Então, G não contém um caminho aumentador com respeito a M_i que tem início em v, para $2 \leqslant i \leqslant k$.*

❑

Um emparelhamento máximo em um grafo G pode ser obtido pela construção de uma seqüência $M_1, M_2, \ldots, M_k$ de emparelhamentos em G, onde M_1 é um emparelhamento inicial em G, M_i é obtido de M_{i-1}, para $2 \leqslant i \leqslant k$, aumentando M_{i-1} sobre algum caminho aumentador e M_k é o maior emparelhamento possível, isto é, M_k é o emparelhamento de cardinalidade máxima.

2 Algoritmo

A idéia do algoritmo pode ser descrita da seguinte maneira. Seja M um emparelhamento qualquer em um grafo G. Selecione um vértice v que não é emparelhado com respeito a M e determine se existe um caminho aumentador que tem v como início. Se existe um caminho como esse, então aumente M sobre esse caminho obtendo um novo emparelhamento M' com cardinalidade $|M'| = |M| + 1$. Então, v é um vértice emparelhado com respeito a M'. Suponha, entretanto, que não existe um caminho aumentador com respeito a M que tem início em v. Então, pelo corolário 1.2, não é necessário procurar por caminhos aumentadores iniciando em v nos passos subseqüentes.

Seja G é um grafo bipartido com emparelhamento M e suponha que v é um vértice não emparelhado com respeito a M. Usando a busca em largura, uma árvore com raiz v é construída de tal forma que todos os caminhos da raiz v às folhas dessa árvore são caminhos alternantes com respeito a M. Essa árvore é chamada uma **árvore alternante**. Além disso, se existe um caminho aumentador que tem início em v, então essa informação é obtida da construção da árvore alternante.

Para verificar a existência de um caminho aumentador, o seguinte processo realizado. Se existe um vértice u não emparelhado adjacente a v, então um caminho aumentador v, u foi encontrado. Então, M é aumentado sobre este caminho aumentador para obter um emparelhamento de cardinalidade $|M|+1$. Caso contrário, todo vértice adjacente a v é emparelhado. Neste caso, uma árvore alternante com raiz v é construída, como na figura 2 a seguir. A raiz dessa árvore é o vértice v, posicionado no nível 0, e todos os vértices adjacentes a v em G, digamos $u_1, u_2, \ldots, u_k$, são posicionados no nível 1 e conectados a v. Agora, seja $u_i v_i \in M$, com $1 \leqslant i \leqslant k$, arestas do emparelhamento M adjacentes a $u_1, u_2, \ldots, u_k$. Os vértices $v_1, v_2, \ldots, v_k$ são posicionados então no nível 2 e conectados a u_i, para $1 \leqslant i \leqslant k$. Suponha que todos os níveis da árvore alternante foram construídos até o nível m, onde m é par, e que nenhum caminho aumentador com início em v tenha sido encontrado. A construção da árvore alternante continua da seguinte forma. Para todo vértice x no nível m da árvore alternante, examine todo vértice y adjacente a x. Se y já pertence à árvore alternante então não há nada a fazer. Caso contrário, y não pertence à árvore alternante e então y é um vértice emparelhado ou não. Se y é um vértice emparelhado, então $yz \in M$ para algum z e z não pertence à árvore alternante. Então, y e z são adicionados à árvore alternante nos níveis $m+1$ e $m+2$, respectivamente, e conectados. A figura 2(a) ilustra essa situação. Entretanto, se y é um vértice não emparelhado, então um (v, y)-caminho aumentador foi encontrado. A figura 2(b) ilustra esse caso.

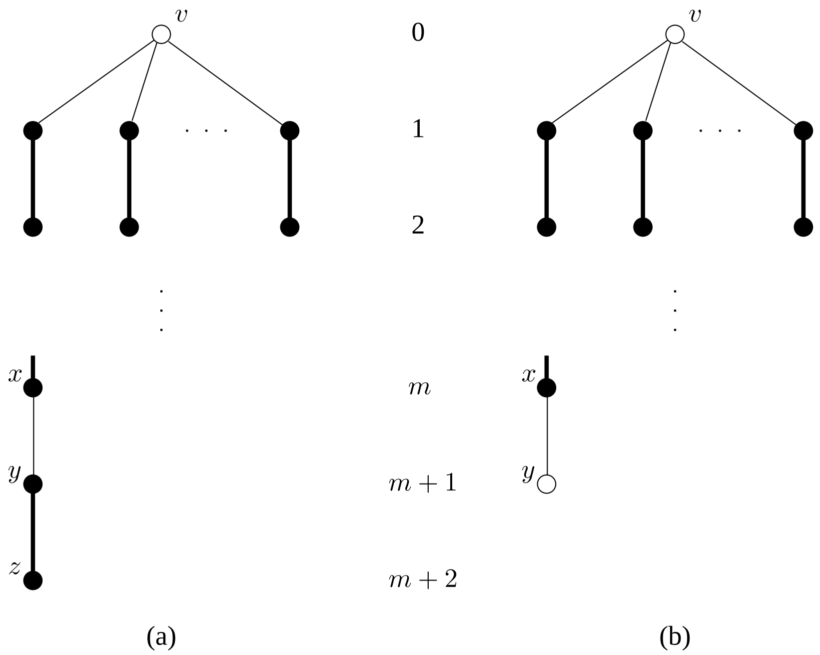


Figura 2: Construção de uma árvore alternante.

A construção da árvore alternante termina quando um caminho aumentador é encontrado ou quando não há mais como acrescentar novos níveis a essa árvore. No primeiro caso, o emparelhamento M é aumentado sobre o caminho aumentador encontrado, produzindo um emparelhamento de cardinalidade $|M|+1$.

Suponha que exista um vértice não emparelhado com relação a este novo emparelhamento que não foi considerado anteriormente como raiz de uma árvore alternante. Uma árvore alternante com raiz nesse vértice não emparelhado é construída. Se não existem mais vértices com essa característica, um emparelhamento de cardinalidade máxima foi encontrado. No segundo caso, não existe um caminho aumentador com respeito a M que tem início em v. Se G ainda possui um vértice não emparelhado que não foi previamente considerado como raiz de uma árvore alternante, então tal vértice é escolhido para ser raiz de uma nova árvore alternante. Se tal vértice não existe, um emparelhamento de cardinalidade máxima foi obtido.

A descrição do algoritmo acima é apresentada em pseudocódigo a seguir. Cada vértice $u \in V_G$ tem um atributo `emparelhado`, que contém um nó $v \in V_G$ se a aresta uv pertence ao emparelhamento corrente, um atributo `raiz` que indica logicamente se o vértice u já foi raiz de alguma árvore alternante, um atributo `árvore`, indicando logicamente se o vértice u pertence à árvore alternante, um atributo `pai`, indicando o vértice pai de u na árvore alternante, além do atributo `Adj` que contém a lista dos vértices adjacentes a u em G.

ALGORITMO EMPARELHAMENTO-MÁXIMO-B(G, M): recebe um grafo bipartido G e um emparelhamento M em G e devolve um emparelhamento de cardinalidade máxima M_1 em G.

1: $M_1 \leftarrow M$
2: **para** cada $u \in V_G$ **faça**
3:    raiz$[u] \leftarrow$ **falso**
4: **para** cada $u \in V_G$ **faça**
5:    **se** emparelhado$[u] = \lambda$ **e** raiz$[u] =$ **falso então**
6:       raiz$[u] \leftarrow$ **verdadeiro**
7:       árvore$[u] \leftarrow$ **verdadeiro**
8:       **para** cada $v \in V_G$ **faça**
9:          **se** $v \neq u$ **então**
10:             árvore$[v] \leftarrow$ **falso**
11:       $Q \leftarrow \emptyset$ $\triangleright$ Fila
12:       ENTRA-NA-FILA(Q, u)
13:       **enquanto** $Q \neq \emptyset$ **faça**
14:          $v \leftarrow$ SAI-DA-FILA(Q)
15:          **para** cada $w \in$ Adj$[v]$ **faça**
16:             **se** árvore$[w] \neq$ **verdadeiro então**
17:                **se** emparelhado$[w] \neq \lambda$ **então**
18:                   $x \leftarrow$ emparelhado$[w]$
19:                   árvore$[w] \leftarrow$ **verdadeiro**
20:                   árvore$[x] \leftarrow$ **verdadeiro**
21:                   pai$[w] \leftarrow v$
22:                   pai$[x] \leftarrow w$
23:                   ENTRA-NA-FILA(Q, x)
24:                **senão**
25:                   com o apontador 'pai', determine o (u, v)-caminho P' na árvore alternante;
26:                   seja P o caminho alternante obtido da concatenação de P' e do (v, w)-caminho;
27:                   aumente M_1 através do caminho aumentador P, obtendo M';
28:                   faça $M_1 \leftarrow M'$ e saia dos laços das linhas 15 e 13;
29: **devolva** M_1

Um exemplo de execução do algoritmo EMPARELHAMENTO-MÁXIMO-B é ilustrado na figura 3. O grafo bipartido G é apresentado com o emparelhamento inicial M destacado. O tempo de execução desse algortimo é $O(pq)$, se o grafo G tem ordem p e tamanho q.

3 Implementação

Programas que não seguirem estas recomendações não serão corrigidos e terão nota 0.

Linguagem de programação Use a linguagem C padrão (*ANSI C*).

Compilador *Bloodshed Dev-C++* é um ambiente integrado de desenvolvimento para as linguagens C e C++ que usa a implementação Mingw do GCC (*GNU Compiler Collection*) como seu compilador. O *Dev-C++* é desenvolvido por Colin Laplace, Mike Berg e Hongli Lai e é um software livre (sob a

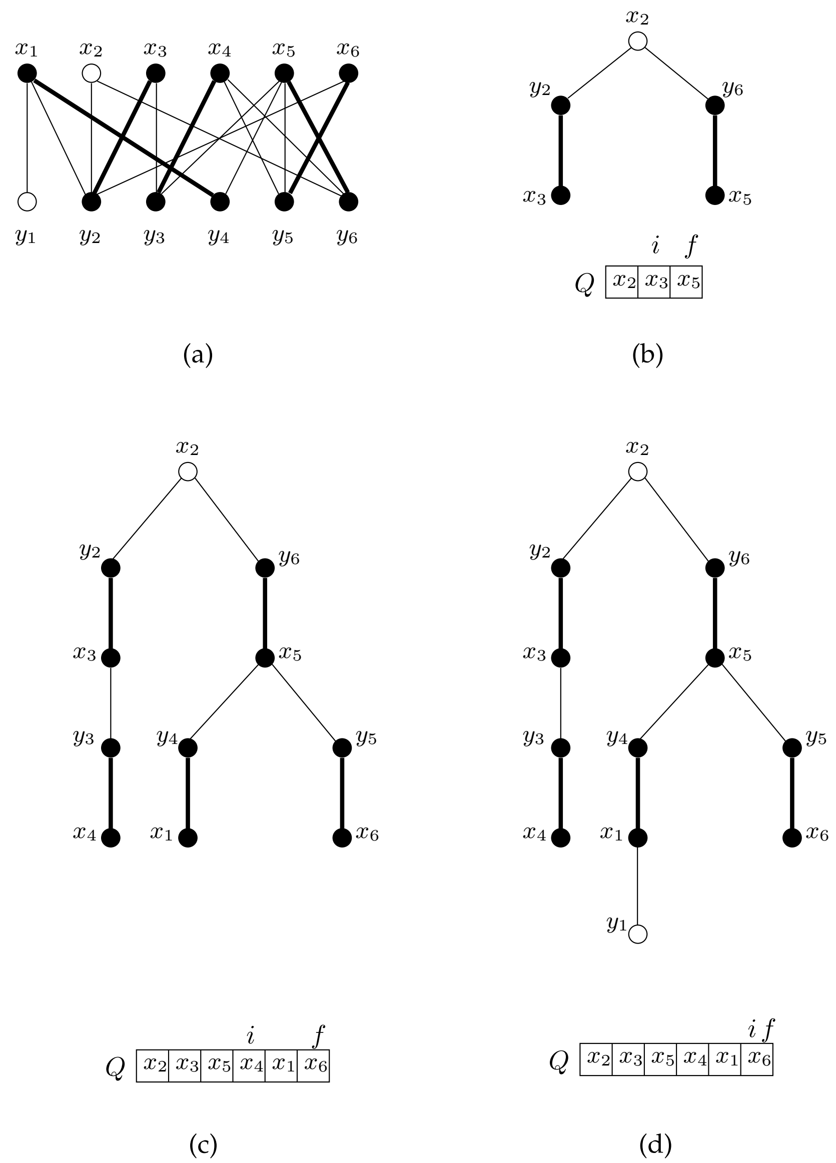


Figura 3: Execução do algoritmo EMPARELHAMENTO-MÁXIMO-B tendo o grafo em (a) como entrada e o emparelhamento M destacado. Em (b), (c) e (d) temos a construção de uma árvore alternante. Observe que um caminho aumentador é encontrado em (d).

GNU General Public License). Isso significa, entre outras coisas, que pode ser distribuído e copiado à vontade.

Você pode utilizar outro compilador, mas deve ter certeza que seu programa pode ser compilado pelo *Dev-C++* antes de entregá-lo.

Na página que contém a descrição desse trabalho[1], há um *link* com instruções para baixar e instalar esse compilador.

Entrada do programa Seu programa deve receber como entrada um grafo bipartido e um emparelhamento inicial e deve devolver um emparelhamento de cardinalidade máxima sobre esse grafo. Seu programa deve receber a entrada através de um arquivo texto contendo as informações sobre o grafo e o emparelhamento inicial. Por exemplo, suponha que o grafo e o emparelhamento da figura 3(a) sejam a entrada. Então, um arquivo de nome `entrada.txt` conterá as seguintes informações:

```
12 16
x1,x2,x3,x4,x5,x6,y1,y2,y3,y4,y5,y6
(x1,y1),(x1,y2),(x1,y4),(x2,y2),(x2,y6),(x3,y2),(x4,y3),(x4,y5),(x4,y6),(x5,y3),(x5,y4),(x5,y5),(x5,y6),(x6,y1),(x6,y5)
(x1,y4),(x3,y2),(x4,y3),(x5,y6),(x6,y5)
```

A primeira linha do arquivo contém a informação da ordem e do tamanho do grafo de entrada. A segunda linha contém os rótulos dos seus vértices. A terceira contém as arestas do grafo e a última as arestas que compõem o emparelhamento inicial.

Saída do programa A saída deve ser um arquivo texto `saida.txt` contendo as arestas de um emparelhamento de cardinalidade máxima. No exemplo da figura 3, a saída deve ser a seguinte:

```
(x1,y1),(x2,y6),(x3,y2),(x4,y3),(x5,y4),(x6,y5)
```

Linha de comando Seu programa deve ser executado em uma linha de comando da seguinte forma:
`> programa [entrada.txt] [saida.txt]`

Grupos Esse trabalho pode ser desenvolvido em grupo. Cada grupo pode conter até 2 alunos.

Entrega Na data de entrega especificada, entregue na secretaria do DCT um disquete identificado com o(s) nome(s) do(s) integrante(s) do grupo contendo o programa fonte do trabalho.

Referências

[1] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, The Macmillan Press LTD, 1977.

[2] G. Chartrand and O. R. Oellermann, *Applied and Algorithmic Graph Theory*, McGraw-Hill, Inc., 1993.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, 2nd Edition, 2001.

[1] `http://www.dct.ufms.br/~fhvm/disciplinas/2005/grafos/tarefas.html`