MAC 328 — Algoritmos em Grafos

Lista 1 — 31/3/2014

Antes de mais nada, confira as páginas do Prof. Paulo Feofiloff para o mesmo curso. As várias seções têm nomes praticamente idênticos aos tópicos mencionados no curso.

Para cada seção, faça {

1. Leia, compare as definições, teoremas e algoritmos com o visto em aula, e comente no fórum sobre as diferenças.
2. Faça o máximo possível dos exercícios. Se achar algum especialmente difícil, peça dica no fórum. Se alguém pedir dica no fórum, pode dar.
3. Entregue um ou outro escrito para avaliação pelo monitor. Não vai valer nota, mas é um retorno dizendo se você está escrevendo bem, e se não, como melhorar.

}

1. Mostre que numa busca em profundidade num grafo, para toda aresta $\{u, v\}$ um dos arcos é de retorno; o outro ou é da árvore ou é descendente. Consequentemente, não há arcos cruzados.

2. Se X é um subconjunto de vértices do grafo G, o grafo $G\backslash X$ tem como vértices $VG - X$, e como arcos todos os arcos de G que **não** têm uma ponta em X. Escreva uma função que recebe um grafo dado por listas de adjacência e um subconjunto de vértices X (dado como você quiser, explique), devolve uma representação de $G\backslash X$ por listas de adjacência. Como deve ocorrer uma renumeração dos vértices, deve devolver também um mapeamento do nome dos vértices no grafo de saída para o de entrada.

3. O *algoritmo de Tarjan* para encontrar componentes fortementes conexas faz isso com uma única DFS. Ele se baseia no seguintes fatos, mencionados e em parte provados em aula - é parte do exercício escrever demonstrações desses fatos:

   (a) Se S é uma componente forte do digrafo G e T é uma arborescência de DFS, então S induz uma sub-árvore de T, enraizada no primeiro vértice de S descoberto durante a busca (isto é, no vértice `v` tal que `pre[t]` é mínimo entre os vértices de S).

(b) Considere o vetor `low[]`, definido e calculado exatamente como no algoritmo `all_bridges` (veja a aula sobre pontes). Então, um vértice é a raiz (para aquela busca) de uma componente forte se e só se `low[t] == pre[t]`.

O algoritmo procede da seguinte maneira:

(a) Faça uma DFS em G, seguindo o esquema do `all_bridges`, calculando `pre` e `low`. Junto com isso, utilize uma pilha P, onde vértices serão inseridos assim que descobertos - P tem tamanho limitado pelo número de vértices, assim pode ser implementada com um simples vetor.

(b) Quando, ao término do processamento de um vértice `v`, se descobre que ele é raíz de uma componente forte, desempilhe P até desempilhar `v`. Os vértices desempilhados formam uma componente forte

As componentes fortes podem ser devolvidas de várias formas; uma que é consistente com o código visto para o algoritmo de Kosaraju, é através de um vetor `sc` que marca, para cada vértice,em que componente está.

Escreva o algoritmo acima em código C legível.

4. Em `http://cs.anu.edu.au/~bdm/data/formats.txt` estão descritos dois formatos de arquivos para grafos, `graph6` e `sparse6`. Escreva funções que, dado o nome de um arquivo desses devolve um apontador para uma representação do grafo correspondente. Você pode escolher se representa por matriz ou vetor de listas. Podem ser várias funções, ou uma função com comportamento ditado por um ou mais parâmetros. Documente claramente sua função.