Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Projeto e Análise de Algoritmos
Trabalho Prático 2

The Bow-Tie Structure of the Web

Alberto Hideki Ueda

Matrícula: 2014765817

Belo Horizonte
15 de outubro de 2014

1 Introdução

Dado um grafo direcionado, com vértices representando páginas da *Web* e arestas os *links* entre elas, podemos classificar cada vértice como membro de uma das estruturas definidas por Broder [1] - maior *SCC, in, out, tendrils* ou vértices desconectados. A implementação entregue realiza esta classificação.

2 Modelagem

No caso desta tarefa, a modelagem é trivial. O problema já é apresentado como um problema de grafos, onde as arestas são *links* entre diversas páginas, representadas pelos vértices. É interessante notar aqui que o problema pode ser reduzido a encontrar tais estruturas em um grafo menor, onde os vértices são os próprios componentes fortemente conexos do grafo original. Tal grafo não possui SCC's de tamanho maiores que 1, pela própria definição. Esta ideia é base para a compressão realizada em uma das etapas da solução.

3 Breve Descrição da Solução

Dado um grafo G direcionado, executamos o algoritmo aprendido em aula (atribuído a Kosaraju) para encontrar todos os componentes fortemente conexos. O SCC com maior número de vértices é o que contém os vértices do primeiro grupo. Fazemos então uma **compressão** de G, reduzindo-o a um grafo G' com $|N|$ vértices, sendo $|N| <= |V|, |N|$ igual ao número de componentes fortementes conexos no grafo. Por meio de uma DFS a partir do vértice que representa o maior SCC, descobrimos os componentes conexos do grupo OUT. Utilizando a transposta de G em uma nova DFS a partir do maior componente fortemente conexo, descobrimos então os SCC's do grupo *IN*.

Os *tendrils* foram definidos nesta especificação como os vértices que saem de *IN* ou que alcançam OUT mas que não passam pelo grande SCC, ignorando-se as direções das arestas. O algoritmo então passa a trabalhar com um novo grafo G'', que é a versão não-direcionada de G'. Por meio de uma DFS a partir de cada SCC do grupo *IN*, encontramos todos os *in-tendrils*. Já considerando os SCC's do grupo OUT como origem, encontramos todos os componentes *out-tendrils* utilizando DFS's. Os SCC's que estão tanto no grupo *in-tendrils* quanto em *out-tendrils*, são os que fazem parte do grupo de *tube-tendrils*. Todos os demais componentes, não encontrados nos passos anteriores, são desconectados em relação ao maior SCC.

Ao final, realizamos a descompressão dos componentes, chegando finalmente à classificação de cada vértice v de G, baseando-se no componente ao qual v pertencia.

4 Etapas Relevantes da Solução

1. **Leitura da entrada padrão**
   Devido à simplicidade do formato dos arquivos de entrada, a leitura de dados é trivial. Via arquivos de entrada e de saída, o grafo é construído por meio de listas de adjacência. Mais especificamente, *vectors* de listas encadeadas. Em termos de memória, utiliza espaço $O(V+E)$. A inserção é em tempo constante embora haja uma perda de desempenho em relação ao acesso à cada aresta, $O(|E|/V)$,

2. **DFS**
   DFS's são executadas ao longo do algoritmo, basicamente para determinar as categorias de cada vértice na rede *bow-tie*. Como o grafo é comprimido, o espaço de busca também se reduz ao número de componentes fortementes conexos. Cada DFS tem complexidade de tempo $O(V+E)$[2]. O uso de memória é $O(V)$, devido à utilização de listas auxiliares.

3. **Detecção de Componentes Fortementes Conexos**
   Utiliza-se o algoritmo de Kosaraju. Complexidade de tempo $O(V+E)$[2].

4. **Compressão do grafo**
   A compressão é bastante simples e exige apenas uma visita a cada vértice e aresta do grafo original. Em termos de tempo e memória, a complexidade é $O(V+E)$.

Portanto, o custo total do algoritmo é $O(V(V+E))$ em termos de tempo e de memória utilizando listas ligadas. Para todas as instâncias fornecidas no fórum e da especificação, o programa executou em menos de 0.1 segundo na máquina de teste. A execução para as bases de Stanford executam em tempo da ordem de 10 minutos, porém após refatorações e correções de erros, o programa passou a ser interrompido por uso excessivo de memória. Embora apresente resultados corretos para instâncias menores, deve-se buscar soluções ou contornos para tais problemas.

5 Dificuldades Encontradas

- Entradas de maior tamanho: após algumas refatorações, houve problemas em executar o algoritmo para entradas muito grandes. Alguns pontos foram melhorados, principalmente com a utilização de ponteiros ao invés de cópias de dados. Porém, alguns pontos de melhoria ainda podem ser explorados, pois em algumas instâncias ainda é lançado erro de alocação de memória;

- Dúvidas na especificação: as mensagens no fórum em relação à definição dos grupos *TUBE-TENDRIL* mudaram certos trechos de código em tempo de implementação. Além disso a definição do intervalo dos vértices (ao final, 1 a $|V|$) causou algumas dúvidas a princípio;

- Utilização de *blacklists*: ao invés de remover os vértices e arestas desnecessárias nos algoritmos de grafos, utilizei listas de bloqueio, o que causou impacto na memória e também na qualidade do código;

- Utilização de diversos mapas: o uso de diversos mapeamentos ao longo do algoritmo tornou o código um pouco confuso e de difícil leitura.

6 Conclusão

A categorização dos vértices por meio de componentes fortementes conexos mostrou-se mais trabalhosa que parecera de início. Embora seja de fácil visualização, a solução exigiu muito cuidado a cada passo do algoritmo. O problema tornou-se maior ao avaliar grafos com mais de 10^6 arestas, demandando diversas refatorações e diminuição do uso de memória no algoritmo.

O algoritmo ainda precisa ser melhorado para alguns cenários disponíveis no website de Stanford, embora apresente resultados satisfatórios para instâncias menores, como as disponibilizadas na especificação e no Moodle.

Um ponto interessante a ressaltar neste trabalho é a quebra de expectiva em relação ao tamanho médio dos grupos *IN* e *OUT* comparados ao tamanho do maior SCC. A primeira intuição era de que tais grupos fossem consideravelmente menores que o maior componente fortemente conexo, porém ao ler os artigos da área de pesquisa é perceptível que o tamanho destes conjuntos de entrada e saída não são desprezíveis.

Referências

[1] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Comput. Netw.*, 33(1-6):309–320, June 2000.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.