Trabalho prático 2 – Projeto e Analise de Algoritmos

Mateus V. Souza

Departamento de Ciências da Computação – Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos, 6627 – Prédio do ICEx – Pampulha, Belo Horizonte, Minas Gerais, Brasil CEP: 31270-90

mateus.vilela@dcc.ufmg.br

1. Descrição do problema

Neste trabalho é dado um grafo do qual os nós são os sites da internet e os arcos as conexões entre estes sites. Os arcos são direcionados e indicam que o site X contém um link para o site Y. Com isso a entrada do problema são os arcos que são os links do site X para o site Y. Isto pode ser melhor visualizado na Figura.1.

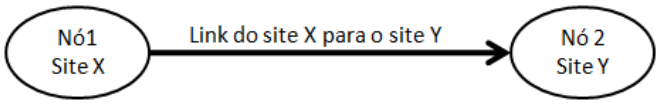

Figura 1. - Ilustração do grafo do problema

O problema consiste em dado um grafo de sites como nós e os arcos como os links entre eles, se separar 7 tipos de nós dentre os grupos citados abaixo:

- O grupo de nós SCC. O maior componente fortemente conectado do grafo.
- O grupo de nós IN. Nós que possuem um caminho até um nó do SCC, mas que os nós do SCC não possuem um caminho até eles.
- O grupo de nós OUT. Nós dos quais os nós do SCC possui um caminho até eles, mas eles não possuem um caminho até um nó do SCC.
- O grupo de nós TendrilA. Nós que são alcançados por um nó do grupo IN, caso você desconsiderar a direção de suas arestas.
- O grupo de nós TendrilB. Nós que são alcançados por um nó do grupo OUT, caso você desconsiderar a direção de suas arestas.
- O grupo de nós TendrilC. Nós que são alcançados tanto por um nó do grupo OUT e um pelo grupo IN, caso você desconsiderar a direção de suas arestas.
- O grupo de nós Disconnected. Nós que não possuem um caminho, ignorando ou não a direção dos arcos, a nenhum nó dos grupos SCC, IN, OUT, TendrilA, TendrilB e Tendrilc.

2. Estratégia de resolução do problema

Primeiramente se lê os dados e se os guardou em uma lista de adjacência que será citada mais a frente neste documento.

A primeira tarefa foi descobrir os conjuntos fortemente conectados do grafo. Para isto se realizou varias buscas em profundidade nos nós do grafo como a orientação inicial dos arcos até que todos os nós fossem visitados. Após está etapa se guardou os tempos de

finalização dos nós nestas buscas em profundidade. Então se realiza novas buscas em profundidade, mas com os arcos com seus sentidos invertidos e a ordem de realização das buscas em profundidade sendo realizadas escolhendo sempre o nó não visitado ainda com o maior tempo de finalização até aquele momento. Este procedimento nos fornecerá os grupos fortemente conectados do grafo.

Iremos selecionar o grupo de nós fortemente conectados com o maior número de elementos. Se ira pegar um nó deste grupo e realizar uma busca em profundidade a partir dele sem levar em conta a direção dos arcos. Os nós não alcançados serão o grupo dos nós Disconnected. Os outros nós serão marcados como possíveis nós SCC, IN, OUT ou de uma dos três tendrils.

A partir do nó do maior grupo fortemente conectado se realiza uma nova busca em profundidade só que com os arcos com suas orientações corretas. Os nós visitados serão marcados como OUT. Após isto se realiza outra busca em profundidade partindo do mesmo nó da busca realizada anteriormente, mas a direção dos arcos será considerada como a oposta a original. Os nós visitados e que já tiverem sidos marcados como OUT, serão remarcados como SCC. Os nós visitados e não marcados como OUT serão marcados como IN.

No final desta etapa já discriminamos os nós do grafo SCC, IN, OUT e Disconnected. Falta marcar os nós de acordo com o tipo de Tendril que eles pertencem.

Para se identificar os tipos de tendril eu faço uma busca em profundidade sobre um nó que é do tipo tendril e descubro se ele consegue alcançar um nó do tipo OUT ou IN apenas passando por nós que também são do tipo tendril. Para isso não levo em consideração a direção dos arcos na busca em profundidade. Caso ele consiga alcançar um nó do tipo IN e um do tipo OUT, realizo a mesma busca em profundidade novamente remarcando os nós tipo tendril que ele alcança como TendrilC. Caso ele só alcance nós tipo OUT, realizo a mesma busca em profundidade novamente remarcando os nós tipo tendril que ele alcança como TendrilB. Caso ele só alcance nós tipo IN, realizo a mesma busca em profundidade novamente remarcando os nós tipo tendril que ele alcança como TendrilA. Farei isto para todos os nós que podem ser um tendril e ainda não foram marcados como TendrialA, tendrilB ou TendrilC.

Ao final disto, tenho os sete grupos de nós, SCC, IN, OUT, TendrilA, TendrilB, TendrilC e Disconnected, marcados. Percorro a lista de nós e os guardo em listas os nós de cada tipo. Depois escrevo cada lista do tipo de nós em um arquivo TXT para o seu respectivo tipo.

3. Abordagens criadas

Foram criadas três abordagens para o problema. O tempo de execução de cada abordagem será apresentado posteriormente neste documento. Abaixo a descrição da diferença de cada abordagem.

Abordagem 1

A primeira abordagem do problema foi realizar uma lista de nós e cada nó com uma lista de ponteiros para os nós que são adjacentes a ele. Está abordagem se demonstrou frustrante, pois a função de ler os dados e alimentar está estrutura se demonstrou muito lenta e tendo uma função de complexidade de O(E x ln(V)) (onde V é o número de

vértices e o E é o número de arestas). Esta complexidade foi obtida por analise probabilística e que não entrarei em detalhes neste documento como a encontrei, pois o documento se tornaria muito longo. Tal complexidade supera a da busca em profundidade (O(V + E)) e se torna a complexidade do problema. Os tempos de execução dessa abordagem deixam clara a desvantagem desta abordagem.

Abordagem 2

A segunda abordagem foi tendo em vista a propriedade de que os nós vão de 1 a |V|. Com isto a estrutura de armazenamento de dados usada passou a ser um Vector. Está estrutura permite um rápido e fácil acesso ao nó que se quer acessar pelo identificador dele. O tempo de leitura caiu muito e a complexidade da função de leitura dos dados e a alimentação dos dados na estrutura do problema passou a ter uma complexidade O(E), sendo inferior a da busca em profundidade O(V + E). Contudo, está abordagem realizava a ordenação dos nós pelo tempo de finalização deles que poderia no pior caso ter complexidade de $O(V^2)$, pois foi usado o Insertion Sort.

Abordagem 3

A terceira abordagem dispensa a ordenação, pois ao longo da primeira busca em profundidade realizada ela guarda o valor do tempo de finalização e do nó que possui este tempo de finalização em ordem decrescente do tempo de finalização. Este fato dispensa a ordenação dos nós de acordo com o tempo de finalização.

3. Programa criado

Irei analisar nesta sessão apenas a abordagem 3 que é a que será entregue como versão final para este trabalho.

As estruturas de dados criadas para armazenamento de dados mais relevantes (que variam com o tamanho da entrada do problema) são:

- Estrutura NO para armazenar os dados de um nó. Ela possui os campos de identificação do tipo de nó que ela é, o grupo fortemente conexo que ela pertence, variáveis que contem o estado para buscas em profundidade, e três listas que conterão os nós se chega a partir deste nó, os nós que se consegue chegar a este nó, e a ultima que engloba os nós das duas primeiras listas.
- Uma lista de estruturas de um nó citado acima para armazenar todos os nós do grafo.
- Listas para se armazenar os nós de cada tipo de nó do grafo (SCC, IN, OUT, TendrilA, TendrilB, TendrilC e Disconnected)
- Estrutura TempoNoFinal para se armazenar o nó e seu tempo de finalização.
- Lista para armazenar as estruturas TempoNoFinal de cada nó do grafo.
- Lista para armazenar os nós de um grupo fortemente conectado encontrado.
- Lista para armazenar as listas dos grupos fortemente conectados encontrados.
- Lista para armazenar os nós do maior grupo fortemente conectado.

O pseudo-código do programa criado está abaixo:

```
Algoritmo(input)
    Ledados(input);
    /* leu dados de um arquivo "input" */
    DFS1SCC();
    DFS2SCC();
   /* descobre os grupos fortemente conectados */
    ColocaListaDeGrupos();
    SelecionaMaiorGrupo();
   /* seleciona o maior grupo fortemente conectado */
    RetornaEstatosDeBuscaEmProfundidade();
    VerificaTendril();
   /* separa os nós Disconnected dos outros nós */
    RetornaEstatosDeBuscaEmProfundidade();
    VerificaOUT();
    VerificaINeSCC();
    /* Identifica os grupos SCC, IN e OUT */
    IdentificaTipoTendriu();
    /* Identifica os grupos TendrilA, TendrilB e TendrilC */
    ColocaNosEmListaDeTipos();
    /* Coloca os nós de acordo com seus tipos em suas respectivas listas */
    EscreveArquivosDeSaida();
    /* Escreve os arquivos TXT com os nós de cada tipo */
Fim
```

4. Analises de complexidade

A complexidade dos dados lidos é da maior estrutura criada que irá dominar as outras estruturas em nosso caso. Neste caso é a lista de nós do grafo que tem uma complexidade de O(V + E) que é a complexidade de memória do programa. Ela é esta pelo fato de ter V nós armazenados e os nós armazenarem listas com os nós vizinhos a eles, estes nós vizinhos a eles causam a soma de mais E na complexidade da estrutura. Lembrando que V é o número de nós e E o número de arcos.

A complexidade do programa (que é a da abordagem 3) por fim é O(V + E). Pois a complexidade da primeira busca em profundidade é O(V + E) que irá se sobrepor, ou ser igual as complexidades das outras operações do programa como a complexidade de leitura dos dados é O(E), de colocar os nós nas listas de grupos fortemente conexos que é O(V), escrever os dados dos grupos em arquivos TXT que será O(V), entre outras.

5. Implementação e Testes realizados

O programa foi implementado em C++ e o código está junto com está documentação. O código das abordagens 1 e 2 estão em anexo também na pasta “Abordagens anteriores”.

O programa foi testado e teve suas saídas conferidas para o exemplo dado na documentação deste trabalho. Está entrada é chamada como Dados1.txt e está na pasta em anexo chamada “Instancias usadas”.

Os testes foram realizados para as entradas contidas no site HTTP://snap.standford.edu/data/index.html#web para os dados de Stanford e Notredame. Os dados de Notredame foram corrigidos para os identificadores dos nós fiquem entre 1 e |V|. Os arquivos contendo estas entradas são chamados de “web-Stanford.txt” e “NotreDameCorreto.txt” e estão na pasta em anexo com este documento “Instancias usadas”. As soluções obtidas por essas entradas foram analisadas apenas pelo número de nós no maior componente fortemente conectado que se verificou ser igual aos valores contidos no site e por isto foram consideradas corretas.

Foram realizados testes comparativos apenas para estas instâncias para que se tivesse um comparativo entre as abordagens 1, 2 e 3 realizadas no experimento. Sendo que testes com a abordagem 1 se demonstraram impraticáveis pelo tempo que este levava com as instancias do Google e de BerkStan. Foi realizado dois testes com a instância de BerkStan com a abordagem 3 e se verificou um tempo de execução de 57 minutos e outro de 63 minutos.

Os testes foram realizados na maquina com Sistema Operacional Ubunto 14,04 LTS 64-bit, que possui 3,6 RAM, Intel Core i7-3517U CPU 1,90 GHz X 4. O compilador usado foi g++ (Ubunto 4.8.2-19ubunto1) 4.8.2.

Os tempos de execução em segundos para as instâncias estão na Tabela 1.

Tabela 1. Tempos de execução em segundos

Instâncias	Abordagem 1	Abordagem 2	Abordagem 3
Standford	21308,86	2224,24	2208,53
Notredame	23897,91	476,53	470,10

6. Otimizações realizadas

Para se otimizar o código foi retirado variáveis de controle utilizadas para se verificar o que o programa estava realizando ao longo da construção deste. Mas as medidas para se otimizar o código mais bem sucedidas foram mudar a estrutura de armazenamento dos dados e seu modo de leitura, e a retirada de rotinas de ordenação no código que se viram desnecessárias. Isto é claramente visto ao se analisar os tempos de execução de cada abordagem.

7. Observações importantes

A comprovação de certas complexidades não foram comprovadas a fundo, pois podem ser encontradas no livro (Cormen, Leiserson, Rivest, & Stein, 2012) e na internet. Também pelo fato para esta documentação não se tornar muito extensa.

O programa e os arquivos Shell scripts foram testados em um computador contendo o Sistema Operacional Ubunto 14,04 LTS 64-bit.

Os arquivos compile.sh e execute.sh estão juntos com está documentação.

A primeira abordagem foi utilizado largamente a estrutura List neste trabalho. Pois ainda não a tinha utilizado antes e gostaria de me familiarizar com sua utilização.

Caso os identificadores dos nós não esteja entre a faixa de 1 a |V| o programa ira gerar soluções erradas.

A mudança das definições do que seriam os Tendrils A, B e C foi prejudicial para a confecção deste trabalho. Pois tinha imprimido o documento que contem a descrição do trabalho e só fui verifica que se tinha uma alteração neste na terça-feira (7 de outubro) quando já tinha feito boa parte do trabalho e tive que refazer certas partes do mesmo e mudar a estratégia de resolução. Além das mudanças terem causado uma certa confusão sobre o tipo Tendril C.

Caso receba está documentação é pelo fato de não ter conseguido implementar corretamente o programa sem recursões utilizando pilhas, que seria a abordagem 4. Até o momento estou me defrontando com um problema para armazenar o tempo que os nós são finalizados pela primeira busca em profundidade.

O período de incerteza sobre a identificação dos nós de 1 a |V| também foi prejudicial. Pois gerou incerteza e tive que parar de realizar o trabalho esperando uma posição final dos monitores e do professor.

Caso a pessoa que corrigiu está documentação tenha alguma recomendação sobre como faze-lá melhor, ou o programa, ou um feedback sobre o trabalho, fique a vontade para me enviar um e-mail para mateusvilelasouza@gmail.com com as recomendações. Isto se da pelo fato que sou engenheiro de produção e não estou acostumado a fazer documentação e escrever programas.

8. Conclusão

Com este trabalho vi como a leitura de dados e a estrutura de dados utilizada pode alterar o tempo de execução do problema e influenciar na complexidade do programa. Apesar da abordagem 1 ser mais robusta, permite a entrada de um grafo que os identificadores não fique presa entre 1 e |V|, ele se demonstrou muito ineficiente no quesito de tempo de execução em relação ao tempo de execução das outras abordagens.

Por fim, as mudanças e incertezas ao longo do problema a ser tratado geraram retrabalho e possivelmente a impossibilidade da implementação de estratégias mais eficientes. Espero que leve isto em consideração caso o meu programa estoure o limite de tempo estipulado para a execução dos testes.

Bibliografia

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2012). *Algoritmos - teoria e Prática.* Rio de Janeiro: Campus.