Internet Connectivity

Algoritmia e Desempenho em Redes de Computadores

Grupo 9

Diogo Alves (86980)
Luís Crespo (87057)
Pedro Martin (87094)

Prof. João Sobrinho

1 Objectivos

A *Internet* pode ser vista como uma colecção de interconexões entre vários *Autonomous Systems*, que, por sua vez, definem entre si uma relação comercial. Neste trabalho, recorrendo à teoria dos grafos, desenvolvem-se quatro algoritmos que pretendem analisar algumas das possíveis propriedades destas redes: *i)* se é conexa, *ii)* se é *link-biconnected*, *iii)* se é comercialmente acíclica e *iv)* se é comercialmente conexa.

2 Estrutura de dados

A estrutura de dados escolhida para representar o grafo consiste num vetor de listas. O índice do vetor representa o vértice do grafo e os elementos da sua lista os vértices seus vizinhos, estando o tipo de ligação representado em cada nó da lista através de uma variável. O tamanho do vetor corresponde ao *ID* máximo da rede. Escolheu-se este tipo de implementação face a uma matriz de adjacências devido à complexidade de memória da última ($O(|V|^2)$) que com redes com elevado número de nós facilmente atinge a ordem dos GB. O vetor de listas apresenta uma complexidade igual apenas em redes completas e, visto que, as redes normalmente são esparsas, este tipo de solução apresenta bons resultados. A complexidade de adicionar uma aresta nesta estrutura é de *O(1)* e, para grafos esparsos, a complexidade de aceder a uma aresta também pode ser considerada *O(1)*.

3 Algoritmo I

3.1 Explicação

Com este algoritmo pretende-se determinar se um grafo é conexo. Para tal, é feita uma BFS a partir de qualquer vértice (neste caso o primeiro com *ID* válido), considerando todos os tipos de ligações. No final, verifica-se se todos os nós do grafo foram visitados. Se tal acontecer, o grafo é conexo, caso contrário não é.

3.2 Prova de Correção

Um grafo é conexo se e só se a partir de cada vértice for possível chegar a qualquer outro. Isto é equivalente a dizer que um grafo é conexo se e só se, a partir de um nó s arbitrário for possível chegar a qualquer outro.

Condição suficiente: Se a partir de s for possível chegar a todos os nós, então quaisquer nós u e v terão garantidamente pelo menos um caminho entre si - que passa por s.

Condição necessária: Evidentemente, se existir um nó t tal que não exista um caminho entre s e t, então o grafo não é conexo.

3.3 Complexidade

A complexidade deste algoritmo é igual à complexidade da BFS ($O(|V| + |E|)$), somada à complexidade de verificar se todos os nós se encontram descobertos ($O(|V|)$), o que resulta numa complexidade em pior caso de $O(|V| + |E|)$.

3.4 Discussão

A partir da definição de grafo conexo (a partir de cada vértice tem que ser possível chegar a qualquer outro), poderia imaginar-se que um algoritmo para resolver este problema teria que percorrer cada vértice um número de vezes proporcional ao número de vértices, mas de facto, recorrendo ao argumento lógico em 3.2, provou-se que apenas é necessário visitar cada vértice uma vez.

4 Algoritmo II

4.1 Explicação

Com este algoritmo pretende-se verificar se um grafo é *link-biconnected*. Para tal, percorre-se uma DFS a partir de qualquer vértice (começando no primeiro com *ID* válido) e verifica-se a existência de pontes. Caso

existam pontes, o grafo não é *link-biconnected*. A existência de uma ponte é verificada se não existir nenhuma *back edge* na DFS da árvore de um nó v para um antepassado do seu antecessor u, sendo a ligação uv uma ponte.

4.2 Prova de Correção

Um grafo é *link-biconnected* se e só se, ao remover uma aresta, o grafo continua conexo. Isto é equivalente a dizer que grafo é *link-biconnected* se e só se não tiver pontes. Dito isto, o algoritmo implementado parte do partido que uma ligação uv é uma ponte se e só se a sub árvore de DFS do vértice v não tiver uma *back edge* para um antepassado do vértice u.

Condição suficiente: Se para chegar a um nó t sucessor de v a partir de um nó u só é possível fazê-lo percorrendo a ligação uv então é porque uv é uma ponte.

Condição necessária: Se a ligação uv for uma ponte e houver um nó t sucessor de v que tenha ligação para um antecessor x (*back edge*) de u, então, é porque há dois caminhos possíveis entre o nó t e x e a ligação uv não é uma ponte.

4.3 Complexidade

A complexidade deste algoritmo é exatamente igual à complexidade da DFS ($O(|V| + |E|)$) pois consiste em verificar cada vértice e as suas ligações.

4.4 Discussão

A partir da definição de grafo *link-biconnected*, poderia imaginar-se que um algoritmo para resolver este problema seria remover uma a uma todas as arestas e verificar se alguma destas remoções torna o grafo desconexo. Este tipo de implementação resultaria numa complexidade $O(|E| \times (|V| + |E|))$. Ao adaptar o algoritmo discutido nas aulas teóricas referente a pontos de articulação de forma a que ligações entre nós que não tenham *back edges* sejam pontes construiu-se um algoritmo em que é apenas necessário visitar cada vértice e as suas ligações apenas uma vez. A grande diferença face ao algoritmo de pontos de articulação é referente à condição de identificação de ponto de articulação, $l[v] >= d[u]$. No caso do algoritmo desenvolvido é utilizada a condição, $l[v] > d[u]$ porque se $l[v] = d[u]$ a *back edge* do vértice v é o u, ou seja, a ligação uv é uma ponte.

Para o grafo ser *link-biconnected* é primeiro necessário verificar se este é conexo porque podem existir sub-grafos *link-biconnected* num grafo que não é conexo.

5 Algoritmo III

5.1 Explicação

De forma a determinar se um grafo é comercialmente acíclico, recorre-se a uma DFS (começando no primeiro nó com *ID* válido) utilizando apenas ligações *provider-customer*. Quando a DFS aponta para um nó que já foi visitado mas não é considerado acabado, então, é porque foi encontrado um ciclo. Um nó é considerado acabado quando todos os seus sucessores foram visitados. Caso esta condição não se verifique e a DFS tenha terminado, o algoritmo verifica se há nós por visitar e corre de novo uma DFS no primeiro nó por visitar com *ID* válido. Caso não sejam encontrados ciclos e todos os nós tenham sido visitados, então é porque o grafo é comercialmente acíclico.

5.2 Prova de Correção

Um grafo é comercialmente acíclico se e só se não existirem ciclos *provider-customer*, ou seja, se e só se não existir um circuito em que cada nó é *provider* (ou *customer*) do próximo nó, ou seja, um grafo é acíclico se e só se não existir um nó s que seja *provider* de um seu antecessor u, existindo apenas ligações *provider-customer* de s a u.

Condição suficiente: Se existir um nó s que seja *provider* de um nó u, seu antecessor, então, é possível percorrer um ciclo que contenha o nó u, as ligações de u a s, o nó s e a ligação su, percorrendo sempre ligações *provider-customer*.

Condição necessária: Considerando um grafo comercialmente acíclico, existindo um nó s que é sucessor de um nó u, tal que, o caminho de u a s é constituído unicamente por ligações *provider-customer*, então s não pode ser *provider* de u e o grafo permanecer comercialmente acíclico.

5.3 Complexidade

A complexidade deste algoritmo corresponde à soma das complexidades das DFS utilizadas ($O(|V| + |E|)$) com a complexidade de verificar se há nós por visitar ($O(|V| + |E|)$), o que, no pior caso, corresponde a uma complexidade igual a ($O(|V| + |E|)$).

5.4 Discussão

De forma a imprimir um ciclo descoberto basta, no momento em que se encontre um antecessor, u, do nó actual, s, que seja seu *customer*, imprimir o nó actual, s, e recuar na DFS até verificar que esta aponta para o *customer*, u, do último nó, s, visitado pela DFS. À medida que se recua na DFS, caso existam mais nós entre os dois nós mencionados, imprime-se um a um os nós apontados até se verificar a condição anterior. Desta forma, obtém-se a impressão de todos os nós do ciclo descoberto.

6 Algoritmo IV

6.1 Explicação

Para o problema de determinar se um grafo é comercialmente conexo, para grafos acíclicos, foi desenvolvido o seguinte algoritmo:

(1) Verificar se todos os nós de Tier 1 (nós que não têm providers) estão todos ligados uns aos outros por ligações de *peer*. Se não estiverem, o grafo não é comercialmente conexo.

(2) Fazer uma DFS tomando os nós de Tier 1 como pontos de partida e utilizando apenas as ligações *provider-customer*. Se todos os nós forem visitados, o grafo é comercialmente conexo, caso contrário não é.

6.2 Prova de Correção

Para um grafo acíclico, é condição necessária e suficiente para que este seja comercialmente conexo, que os nós de Tier 1 sejam todos *peers* e os restantes nós sejam descendentes destes através de ligações *provider-customer*.

Condição suficiente: Se todos os nós de Tier 1 forem *peers* e todos os restantes nós forem *customers* descendentes destes, então todos os nós conseguem comunicar com todos os outro da mesma "árvore" (enraizada nos nós de Tier 1) usando uma sucessão de ligações P (*customer to provider*) seguido de uma sucessão de ligações C (*provider to customer*). Nós de "árvores" diferentes conseguem comunicar entre si subindo até ao seu respetivo nó de Tier 1 com ligações P, seguida de uma ligação do tipo R (*peer to peer*) até ao nó de Tier 1 correspondente ao nó de destino e terminando com uma sucessão de ligações C até ao nó de destino.

Condição necessária: Assumindo que há um nó u de Tier 1 que não é *peer* de outro nó v de Tier 1. Como nós de Tier 1 não têm providers, então o caminho entre u e v não poderá acabar numa ligação C nem começar numa P, pelo que o único caminho possível é através de uma única ligação R. No entanto, como os dois nós não são *peers*, não existe um caminho entre eles.

Assumindo que há um nó s de Tier superior a 1 que não é descendente de um nó de Tier 1. Ora, como não é de Tier 1, s tem que ter um *provider*. Subindo na "árvore" de providers a partir de s, como assumimos que o grafo é comercialmente acíclico, chegamos inevitavelmente a um nó de Tier 1.

6.3 Complexidade

O pior dos casos que este algoritmo pode encontrar é aquele em que todos os nós são de Tier 1 e são todos *peers* uns dos outros. Neste caso a complexidade do algoritmo é $O(|V|^2)$. No entanto, num caso normal, o número de nós de Tier 1 será bastante reduzido, pelo que a complexidade do algoritmo será igual à complexidade da DFS ($O(|V| + |E|)$).

6.4 Discussão

Apesar de assumirmos que o grafo tem que ser comercialmente acíclico, se este algoritmo for utilizado num grafo com ciclos comerciais e chegar à conclusão que o grafo é comercialmente conexo, então o grafo é mesmo comercialmente conexo. Isto porque o algoritmo só chega a esta conclusão se for possível chegar a todos os nós a partir de nós de Tier 1 apenas com ligações *customer to provider*. No entanto, se o grafo for cíclico e o algoritmo disser que ele não é comercialmente conexo, não podemos confiar nesta decisão. Para resolver este problema, implementámos também outro algoritmo que funciona para qualquer grafo, mas cuja complexidade é muito superior ($O(|V| \times (|V| + |E|))$), pelo que não dá resultados em tempo útil. Não achámos relevante explicar este algoritmo em detalhe neste relatório, uma vez que o seu funcionamento é trivial: consiste em pegar em cada nó e fazer uma DFS a partir desse nó por caminhos comerciais, verificando se a partir de cada nó é possível chegar a qualquer outro utilizando apenas caminhos comerciais.