Algoritmos em Grafos

Cláudia Linhares Sales

Julho 2020

Algoritmo de Busca em Largura - ABL

Essas notas de aula seguem livremente, e portanto não pretendem ser tradução, o livro "Introduction to Algorithms", Third Edition, de Cormen, Leiserson, Rivest e Stein, MIT Press, 2009. Elas pretendem ser o que eu escreveria no quadro em nossas aulas presenciais.

Para a compreensão deste tópico, relembre os conceitos de adjacência; vizinhos de um vértice v, onde o conjunto deles é denotado por $N(v)$; grau de um vértice v, denotado por $grau(v)$; caminho e caminho mínimo entre dois vértices; e distância entre dois vértices u e v, denotada por $\delta(u, v)$. Adicionalmente, diremos que um vértice u alcança um vértice v, se há um caminho de u a v.

Recorde também que ao final da execução dos algoritmos de busca aplicados a um grafo $G = (V, E)$, podemos definir o grafo G_π da seguinte forma:

- $V(G_\pi) = \{u \in V(G) | cor[u] = preta\}$
- $E(G_\pi) = \{(u, v) \in E(G) | pai[v] = u\}$

Como apresentado anteriormente, a ordem de visita dos vértices da *Busca em Largura* pode ser assim resumida: tendo descoberto um vértice, visite, um a um, os seus vizinhos não visitados (ou seja, brancos), antes de visitar os vizinhos de seus vizinhos. Em seguida, visite, um a um, os vizinhos dos vizinhos, aplicando o mesmo princípio de busca.

Pela descrição da busca, caso fóssemos colocar o vértices em ordem a partir de um vértice u, a ordem seria:
u
$A = \{v | v \in N(u), cor[v] = branca\}$ – vizinhos não visitados de u
$B = \{w | w \in N(v), v \in A, cor[w] = branca\}$ – vizinhos não visitados de vizinhos de u
etc.

Para garantir que nenhum vértice do conjunto B seja visitado antes dos vértices do conjunto A, pode-se usar uma fila Q, onde os vértices de A antecedem os vértices de B em Q, e assim sucessivamente.

Observe que a ordem de visita do vértices de A (ou de B) é irrelevante, pois não causa conflito com o princípio geral da Busca em Largura.

Segue abaixo o Algoritmo de Busca em Largura (ABL). Para focar na essência do algoritmo, não criaremos as estruturas de dados utilizadas (vetores d, cor e pai, a fila Q e a lista de adjacências Adj que armazena o grafo). A estrutura de dados que armazena G é uma lista de adjacências indexada por cada um dos vértices de G, de forma que $Adj[u]$ é a lista de todos os vizinhos de u.

Com essas observações, veja o Algoritmo:

Algorithm 1: Algoritmo de Busca em Largura – ABL

Input : Um grafo (ou digrafo) $G = (V, E)$ e um vértice $s \in V(G)$
Output: Vetores d e pai

```
1  for todo v ∈ V(G) do
2  |   cor[v] ← branca;
3  |   pai[v] ← ⊤;
4  |   d[v] ← ∞
5  end
6  Q ← ∅;
7  d[s] ← 0;
8  cor[s] ← cinza
9  Enfila(Q, s);
10 while Q ≠ ∅ do
11 |   u ← Desenfila(Q);
12 |   for todo v ∈ Adj[u] do
13 |   |   if cor[v] = branca then
14 |   |   |   pai[v] ← u;
15 |   |   |   d[v] ← d[u] + 1;
16 |   |   |   cor[v] ← cinza;
17 |   |   |   Enfila(Q, v);
18 |   end
19 |   cor[u] ← preta;
20 end
```

A complexidade do ABL

Podemos identificar 2 partes no algoritmo:

1. As Linhas 1 a 5 do algoritmo respondem pela inicialização dos vetores d, cor e pai, controladas pelo comando **for** e tem **custo** $O(n)$, uma vez que o grafo tem n vértices;

2. As Linhas 10 a 20 do algoritmo respondem pelo processo de busca propriamente dito, controlado pelo comando **while**, cuja repetição depende

da fila Q. Enquanto a fila Q não está vazia, esse laço é executado. Para entrar na fila, um vértice branco torna-se cinza. Como não há comandos de atribuição de cor branca nesse laço, ao perder a cor branca, um vértice jamais tornará-se branco novamente. Logo, cada vértice só pode entrar uma única vez em Q, portanto o **while** itera no máximo n vezes, e as Linhas 11 e 19 do algoritmo são executadas no máximo n vezes. Dentro do bloco **while**, existe um comando **for** (Linhas 12 a 18), que é executado tantas vezes quando forem os vizinhos de cada vértice. O número de vizinhos de cada vértice v é o grau de v ($grau(v)$. A soma dos graus de todos os vértices de um grafo G é exatamente $2m$, onde m é o número de arestas de G. Logo, o total de execuções das Linhas 13 a 17 neste bloco **for** é $O(m)$.

Concluímos então que a soma dos custos dessas partes leva à complexidade $O(n + m)$.

As propriedades do ABL

As propriedades importantes do ABL são:

1. Ao final da execução do ABL todos vértices tem cor branca ou preta, a cor preta indicando que ele foi visitado.

2. Ao final da execução do ABL, todos os vértices que s alcança (ou seja que tem um caminho de s até eles) são visitados e tem a cor preta;

3. Para todo $v \in V(G)$, $d[v] = \delta(s, v)$, ou seja, $d[v]$ possui o tamanho de um caminho mínimo entre s e v, que é a distância entre s e v.

4. o grafo G_π tal qual foi definido anteriormente é uma árvore enraizada em s, contendo todos os vértices que s alcança e caminhos mínimos entre s e eles.

Temos que provar cada uma dessas propriedades.

Proposição 1. *Ao final da execução do ABL, todos os vértices de G tem cor branca ou preta, a cor preta indicando que ele foi visitado.*

Demonstração. Observe que até a Linha 7 do algoritmo, todos os vértices de G possuem a cor branca. Só há dois comandos que mudam a cor branca para cinza: as Linhas 8 e 16, que são seguidas pela inclusão do vértice agora cinza na fila Q. Como o algoritmo só termina quando a fila Q está vazia, cada vértice cinza retirado de Q na Linha 11 será tornado preto na Linha 19. Logo, ao final da execução, não há vértices cinza e todos os vértices pretos passaram pela fila Q, ou seja, foram visitados. □

Para provar as proposições seguintes, precisamos provar dois lemas clássicos de teoria de grafos.

Lema 2. *Seja $G = (V, E)$ um grafo e (u, v) uma aresta de G. Logo, $\delta(s, v) \leq \delta(s, u) + 1$.*

Demonstração. Suponha primeiro que s não alcance u, logo $\delta(s, u) = \infty$ e portanto a desigualdade vale para qualquer valor de $\delta(s, v)$. Agora suponha que s alcance u. Como existe a aresta (u, v), s alcanca v. Portanto, um limite superior para o caminho mínimo de s a v será dado pelo tamanho do caminho mínimo de s a u (dado por $\delta(s, u)$) mais uma unidade que seria referente à aresta (u, v). E isso encerra a prova. □

Lema 3. *Seja $G = (V, E)$ um grafo e $P =< v_0, \ldots, v_k >$ um caminho mínimo de G entre v_0 e v_k. Então qualquer trecho $< v_i, \ldots, v_j >$ de P, $0 \leq i < j \leq k$, é um caminho mínimo entre v_i e v_j.*

Demonstração. Suponha por absurdo que o Lema é falso e considere qualquer trecho $< v_i, \ldots, v_j >$ de P. Se ele não é um caminho mínimo entre v_i e v_j, existe um caminho Q entre v_i e v_j de tamanho inferior a $j - i$. Portanto, o caminho $P' =< v_0, \ldots, v_{i-1}, Q, v_{j+1}, \ldots, v_k >$ é mais curto do que P, contradizendo a hipótese de P ser um caminho mínimo entre v_0 e v_k. □

Proposição 4. *Ao final da execução do ABL, para todo vértice $v \in V(G)$ tal que existe um caminho de s até v (ou seja que s alcança) é visitado (e portanto tem a cor preta).*

Demonstração. Observe que se existe um caminho de s a um vértice v, então existe um caminho mínimo entre s e v e logo $\delta(s, v) \neq \infty$. Podemos provar, por indução, na distância k entre s e os vértices v, que s alcança que v é visitado, torna-se cinza e é enfilado em Q e posteriormente desenfilado. Para $k = 0$, temos que $\delta(s, s) = 0$, s é tornado cinza e enfilado em Q nas Linhas 8 e 9, e desenfilado na Linha 11. Portanto a base da indução está provada. Agora, considere um vértice v tal que $\delta(s, v) = k$. Então, existe um caminho mínimo $P =< s = v_0, \ldots, v_k = v >$ de tamanho k. Pelo Lema 3, o trecho $< s = v_0, \ldots, v_{k-1} >$ é um caminho mínimo entre s e v_{k-1}. Logo, $\delta(s, v_{k-1}) = k - 1$. Portanto, por hipótese de indução, v_{k-1} foi visitado, tornou-se cinza e foi enfilado em Q e será posteriormente desenfilado. Quando v_{k-1} for desenfilado na Linha 11, todos os seus vizinhos serão visitados nas Linhas 12 a 18. Como $v_k = v$ é vizinho de v_{k-1}, se já não o tiver sido antes, ele será tornado cinza, enfilado e desenfilado posteriormente, pois o ABL só termina quando a fila Q estiver vazia. Como todo vértice desenfilado torna-se preto na Linha 19, isso encerra a prova desta proposição. □

Queremos provar agora que para todo $v \in V(G)$, $d[v] = \delta(s, v)$, ou seja, $d[v]$ possui o tamanho de um caminho mínimo entre s e v, ou seja, a distância entre s e v, e que o grafo G_π tal qual foi definido anteriormente, é uma árvore enraizada em s, contendo todos os vértices que s alcança e caminhos mínimos entre s e eles.

Faremos isso em três passos:

1. Primeiro, provaremos que depois da etapa de inicialização (Linhas 1 a 5 do ABL), para todo $v \in V(G)$, $d[v] \geq \delta(s, v)$;

2. Depois, estudaremos o comportamento da fila Q com respeito a esse atributo, provando que cada vértice v colocado na fila tem $d[v]$ superior ou igual ao seu predecessor na fila, e que $d[v]$ difere de no máximo uma unidade de $d[u]$, se u for o primeiro vértice da fila;

3. Finalmente, usando as proposições anteriores, vamos provar que $d[v] = \delta(s, v)$, para todo vértice v e que G_π satisfaz às propriedades reinvindicadas.

Proposição 5. *Após a etapa de inicialização do ABL (Linhas 1 a 5) e ao término de execução do mesmo, para todo $v \in V(G)$, $d[v] \geq \delta(s, v)$. .*

Demonstração. Essa prova pode ser feita por indução no número k de operações de *Enfila*. Para o caso base ($k = 1$), pode-se ver na Linhas 7 e 8, que $0 = d[s] \geq \delta(s, s) = 0$. Suponha agora que v é descoberto através da aresta (u, v). Temos que u, já tendo sido enfilado antes, satisfaz à hipótese de indução e portanto $d[u] \geq \delta(s, u)$. Observe que $d[v] = d[u] + 1$ (Linha 15 do ABL). Aplicando o Lema 2 e a hipótese de indução, temos que $\delta(s, v) \leq \delta(s, u) + 1 \leq d[u] + 1 = d[v] - 1 + 1 = d[v]$, provando a proposição. □

Proposição 6. *Em qualquer ponto de execução do ABL, se $Q = v_1, \ldots, v_r$, então $d[v_1] \leq d[v_2] \ldots \leq d[v_r]$. Além disso, $d[v_r] \leq d[v_1] + 1$.*

Demonstração. Essa prova pode ser feita por indução no número k de operações na fila Q. Para $k = 1$, a primeira operação é feita na Linha 9 e a proposição é válida. Suponha agora que ao final da $k-1$-ésima operação, $Q = v_1, \ldots, v_r$ e a proposição seja válida, ou seja, $d[v_1] \leq d[v_2] \ldots \leq d[v_r]$ e $d[v_r] \leq d[v_1] + 1$. Agora, vamos examinar a k-ésima operação em Q. Há duas opções:

1. a operação foi de *Desenfila*. Nesse caso, v_1 será removido da fila e a propriedade continua satisfeita pois a remoção de v_1 não altera o fato de que $d[v_2] \leq d[v_3] \ldots \leq d[v_r]$. Como $d[v_1] \leq d[v_2]$ e $d[v_r] \leq d[v_1] + 1$, temos que $d[v_r] \leq d[v_2] + 1$ e isso encerra este caso.

2. a operação foi de *Enfila*. Neste caso, essa operação foi executada na Linha 17 do ABL, antecedida pela remoção de um vértice da fila (que vamos chamar de v_0) do qual um vértice (que vamos chamar de v_{r+1}) era um vizinho branco. Observe que $d[v_{r+1}] = d[v_0] + 1$ (Linha 15). Por hipótese de indução, com v_0 estava antes de v_1 na fila, $d[v_0] \leq d[v_1]$ e $d[v_r] \leq d[v_0] + 1$. Logo, $d[v_{r+1}] = d[v_0] + 1 \leq d[v_1] + 1$ e portanto ao final da operação de *Enfila*, a diferença do atributo d é de máximo uma unidade entre o primeiro e o último vértice de Q. Por outro lado, o fato de que $d[v_r] \leq d[v_0] + 1 = d[v_{r+1}]$ encerra a prova.

□

Teorema 7. *Ao final da execução do ABL que teve como entrada um grafo $G = (V, E)$ e um vértice s, G_π é uma árvore que contém todos os vértices v que s alcança e um caminho mínimo de G entre s e v. Além disso, para todo $v \in V(G)$, $d[v] = \delta(s, v)$.*

Demonstração. Pelas proposições e lemas anteriores, todos os vértices que s alcança são visitados, e portanto ao final da execução do ABL tem cor preta e, pela definição de G_π, pertencem a G_π. Suponha agora por absurdo que existe $v \in G$, tal que $d[v] \neq \delta(s, v)$, e entre todos os vértices nessa situação considere aquele que minimiza $\delta(s, v)$. Sabemos que $v \neq s$, uma vez que $d[s] = 0$ (Linha 7 do ABL) e $\delta(s, s) = 0$. Também sabemos que $d[v] > \delta(s, v)$ (pela Proposição 3). Suponha que s não alcança v. Portanto, $\delta(s, v) = \infty$. Porém, como $d[v] = \infty$ após a inicialização (Linhas 1 a 5 do ABL), isso contrariaria a escolha de v. Logo s alcança v. Seja u o vértice que antecede v em um caminho mínimo de s a v. Pelo Lema 3, $\delta(s, v) = \delta(s, u) + 1$ e portanto pela escolha de v, $d[u] = \delta(s, u)$. Pela Proposição 6, u foi descoberto antes de v. Então, temos duas opções a examinar:

1. v foi descoberto através de u (pela aresta (u, v)). Nesse caso, $pai[v] = u$, e $d[v] = d[u] + 1 = \delta(s, u) + 1 = \delta(s, v)$, contrariando a escolha de v;

2. se v foi descoberto por outro vértice w. Quando isso ocorreu, fez-se $d[v] = d[w]+1$ (Linha 15) e v é enfilado (Linha 17). Como $d[u] < d[v]$, temos que, pela Proposição 6, u já está em Q. Por outro lado, como a diferença no atributo d entre o primeiro e o último da fila é no máximo de uma unidade, e $d[v] \leq \delta(s, v)$ (Lema 2), temos que $d[v] \leq d[u]+1 = \delta(s, u)+1 = \delta(s, v)$, contrariando novamente a escolha de v.

Falta apenas provar duas propriedades sobre G_π: que é uma árvore e contém caminhos mínimos. Primeiro, vamos arguir que G_π não tem ciclos, caso contrário existiria pelo menos um vértice do ciclo com dois pais, mas isso não é possível. Por outro lado, observe que apenas s não possui pai. Logo, todos os outros vértices de G_π estão na mesma componente que s (para perceber isso, basta retroceder buscando o pai do pai de cada vértice v de $V(G_\pi)$ e assim sucessivamente. Como esse processo não é infinito, isso levará a um vértice sem pai, que no caso é s). Portanto, G_π sendo conexo e acíclico, é uma árvore.

Resta provar que os caminhos em G_π são caminhos mínimos. Seja $v \in V(G_\pi)$ e $pai[v] = u$. Se o caminho de s a u em G_π é mínimo, como $\delta(s, v) = \delta(s, u) + 1$, tomando-se esse caminho e a aresta $(pai[v], v)$ (que é a aresta (u, v)), que pertence a G_π, temos um caminho de tamanho $\delta(s, v)$ entre s e v, que portanto é mínimo. Como esse raciocínio pode ser aplicado a todos os vértices de G_π, e o caminho entre s e s em G_π é mínimo, isso encerra a argumentação. □

Em grafos não ponderados, onde a distância é medida pelo número de arestas (ou arcos) entre os vértices, o ABL calcula a distância entre s e todos os outros vértices de G.

Desafios

Desafio 1: É possível usar o AVL para calcular a distância entre qualquer par de vértices de G, G sendo um grafo ou digrafo não ponderado? Se sim, diga como, apresentando um algoritmo e a sua complexidade. Se não, por que?
Desafio 2: É possível usar o AVL para descobrir se um grafo qualquer é conexo? Se sim, diga como, apresentando um algoritmo e a sua complexidade. Se não, por que?
Desafio 3: É possível usar o AVL para descobrir se um digrafo qualquer é conexo? Se sim, diga como, apresentando um algoritmo e a sua complexidade. Se não, por que?