AULA 03 – BUSCA EM LARGURA

Prof. Daniel Kikuti

Universidade Estadual de Maringá

23 de março de 2015

Sumário

- Breve revisão
- Introdução
- Algoritmo de busca em largura: propriedades, correção e tempo de execução
- Aplicações

Revisão de conceitos

Representação

Dúvidas quanto à representação de grafos?

Revisão de conceitos

Representação

Dúvidas quanto à representação de grafos?

Para a aula de hoje

- Um vértice v é **alcançável** a partir de um vértice s em um grafo $G = (V, E)$ se existe um caminho de s a v em G.

Revisão de conceitos

Representação

Dúvidas quanto à representação de grafos?

Para a aula de hoje

- Um vértice v é **alcançável** a partir de um vértice s em um grafo $G = (V, E)$ se existe um caminho de s a v em G.
- A **distância** de s a v é o comprimento de um **caminho mais curto** de s a v.

Revisão de conceitos

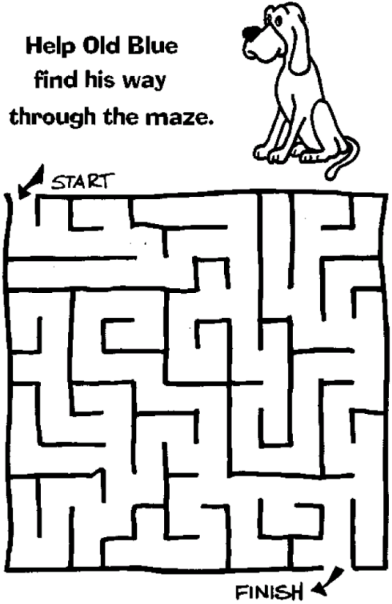
Representação

Dúvidas quanto à representação de grafos?

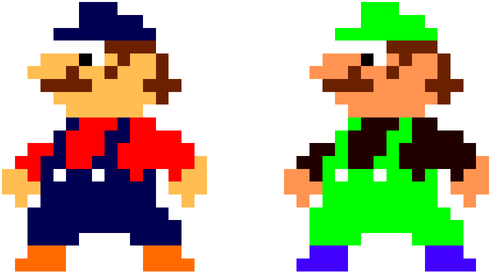
Para a aula de hoje

- Um vértice v é **alcançável** a partir de um vértice s em um grafo $G = (V, E)$ se existe um caminho de s a v em G.
- A **distância** de s a v é o comprimento de um **caminho mais curto** de s a v.
- Se v não é alcançável a partir de s, então dizemos que a distância de s a v é infinita.

Introdução - Labirinto



Introdução - Colorindo figuras



Busca em grafos

Estratégias de busca

- Busca em largura
- Busca em profundidade

Busca em grafos

Estratégias de busca

- Busca em largura
- Busca em profundidade

Busca em largura

- Entrada: $G = (V, E)$ e um vértice s (fonte).

Busca em grafos

Estratégias de busca

- Busca em largura
- Busca em profundidade

Busca em largura

- Entrada: $G = (V, E)$ e um vértice s (fonte).
- Percorre todos os vértices alcançáveis a partir de s em ordem de distância deste.

Busca em grafos

Estratégias de busca

- Busca em largura
- Busca em profundidade

Busca em largura

- Entrada: $G = (V, E)$ e um vértice s (fonte).
- Percorre todos os vértices alcançáveis a partir de s em ordem de distância deste.
- Constrói uma **Árvore de Busca em Largura** com raiz s. Cada caminho de s a um vértice v nesta árvore corresponde a um caminho mais curto de s a v.

Busca em Largura

Visão geral

- Inicialmente a árvore de busca contém apenas o vértice s.
- Para cada vizinho v de s, o vértice v e a aresta (s, v) são acrescentados à árvore.
- Processo é repetido para os vizinhos dos vizinhos de s até que todos os vértices alcançáveis por s sejam inseridos na árvore.
- Usa uma fila `Q` (FIFO).

Busca em Largura

Cores dos vértices marcam o progresso do algoritmo:

- Branco = não descoberto
- Cinza = descoberto e na fronteira (na fila `Q`)
- Preto = todos os seus vizinhos já foram descobertos

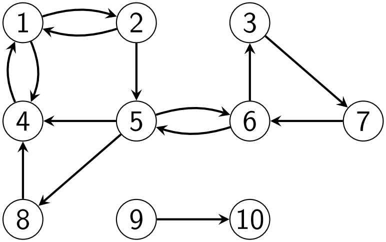
Busca em Largura

Cores dos vértices marcam o progresso do algoritmo:

- Branco = não descoberto
- Cinza = descoberto e na fronteira (na fila `Q`)
- Preto = todos os seus vizinhos já foram descobertos

Exemplos passo-a-passo

- Considere o grafo orientado a seguir:



- Veja no Cormen um exemplo com grafo não direcionado.

O algoritmo

```
Inicializa(G,s)
para cada vertice u ∈ G.V − {s} faça
    u.cor ← branco
    u.dist ← ∞
    u.pred ← NIL
s.cor ← cinza
s.dist ← 0
s.pred ← NIL
```

O algoritmo

```
Busca-em-Largura(G,s)
Inicializa(G,s)
Insere(Q,s)
enquanto Q ≠ ∅ faça
    u ← Remove(Q)
    para cada v ∈ u.Adj faça
        se v.cor = branco então
            v.cor ← cinza
            v.dist ← u.dist + 1
            v.pred ← u
            Insere(Q,v)
    u.cor ← preto
```

Análise do algoritmo

Consumo de tempo

Análise do algoritmo

Consumo de tempo

- Inicialização: $\Theta(V)$

Análise do algoritmo

Consumo de tempo

- Inicialização: $\Theta(V)$
- Cada vértice é inserido na fila `Q` apenas uma vez. Inserção e Remoção de fila consomem $\Theta(1)$. Resultando em um total de $O(V)$.

Análise do algoritmo

Consumo de tempo

- Inicialização: $\Theta(V)$
- Cada vértice é inserido na fila `Q` apenas uma vez. Inserção e Remoção de fila consomem $\Theta(1)$. Resultando em um total de $O(V)$.
- Em uma lista de adjacência cada vértice é percorrido apenas uma vez. A soma dos comprimentos das listas é $\Theta(E)$. Portanto o tempo gasto para percorrer as listas é $O(E)$.

Análise do algoritmo

Consumo de tempo

- Inicialização: $\Theta(V)$
- Cada vértice é inserido na fila `Q` apenas uma vez. Inserção e Remoção de fila consomem $\Theta(1)$. Resultando em um total de $O(V)$.
- Em uma lista de adjacência cada vértice é percorrido apenas uma vez. A soma dos comprimentos das listas é $\Theta(E)$. Portanto o tempo gasto para percorrer as listas é $O(E)$.

Conclusão

A complexidade do algoritmo `Busca-em-Largura(G,s)` é $O(V + E)$. [Análise agregada]

Análise do algoritmo

Correção

Seja $\delta(s, v)$ a distância do vértice s ao vértice v.
Precisamos mostrar que:

- v.dist $= \delta(s, v)$ para todo $v \in G.V$
- Com o predecessor de cada vértice (`v.pred`), definimos uma **árvore de busca em largura** com raiz s.

Análise do algoritmo - Correção

Lema 1

Seja G um grafo (orientado ou não) e $s \in G.V$ um vértice qualquer. Então, para qualquer aresta $(u, v) \in E$,

$$\delta(s, v) \leq \delta(s, u) + 1.$$

Análise do algoritmo - Correção

Lema 1

Seja G um grafo (orientado ou não) e $s \in G.V$ um vértice qualquer. Então, para qualquer aresta $(u, v) \in E$,

$$\delta(s, v) \leq \delta(s, u) + 1.$$

Demonstração

Se u é alcançável de s, então v também é. O menor caminho de s a v não pode ser mais longo que o menor caminho de s a u seguido da aresta (u, v), e portanto a inequação se mantém. Se u não é alcançável de s, então $\delta(s, u) = \infty$, e a inequação se mantém.

Análise do algoritmo - Correção

Lema 2

Seja G um grafo (orientado ou não) e $s \in G.V$ um vértice qualquer. Após a execução do algoritmo `Busca-em-Largura(G,s)`, observa-se a seguinte propriedade:

$$v.dist \geq \delta(s, v) \text{ para todo } v \in G.V.$$

Análise do algoritmo - Correção

Lema 2

Seja G um grafo (orientado ou não) e $s \in G.V$ um vértice qualquer. Após a execução do algoritmo `Busca-em-Largura(G,s)`, observa-se a seguinte propriedade:

$$v.dist \geq \delta(s, v) \text{ para todo } v \in G.V.$$

Ideia da demonstração

- `v.dist` é uma estimativa superior de $\delta(s, v)$.
- Prova indutiva baseada no número de operações de Inserção em `Q`.
- `v.dist` nunca muda após ser inserido na fila.

Análise do algoritmo - Correção

Demonstração

- **Base:** Depois da Inicialização $s.dist = 0 = \delta(s, s)$ e $v.dist = \infty \geq \delta(s, v)$ para $v \in G.V - \{s\}$.
- **Passo da indução:** Quando v é descoberto a partir de u, pela hipótese indutiva temos que $u.dist \geq \delta(s, u)$. Então, pela atribuição (linha 8) e Lema 1 temos:

$$\begin{aligned} v.dist &= u.dist + 1 \\ &\geq \delta(s, u) + 1 \\ &\geq \delta(s, v). \end{aligned}$$

Análise do algoritmo - Correção

Lema 3

Suponha que durante a execução do algoritmo `Busca-em-Largura(G,s)` a fila `Q` contém os seguintes vértices $\{v_1, v_2, \ldots, v_r\}$, onde v_1 é a cabeça de `Q` e v_r é a calda. Então:

- $v_r.dist \leq v_1.dist + 1$
- $v_i.dist \leq v_{i+1}.dist$ para $i = 1, 2, \ldots, r - 1$.

Análise do algoritmo - Correção

Lema 3

Suponha que durante a execução do algoritmo `Busca-em-Largura(G,s)` a fila `Q` contém os seguintes vértices $\{v_1, v_2, \ldots, v_r\}$, onde v_1 é a cabeça de `Q` e v_r é a calda. Então:

- $v_r.dist \leq v_1.dist + 1$
- $v_i.dist \leq v_{i+1}.dist$ para $i = 1, 2, \ldots, r - 1$.

Ideia da demonstração

- Prova indutiva baseada no número de operações de Inserção e Remoção em `Q` (ver demonstração no Cormen [Lema 22.3]).
- O Lema diz que os vértices são inseridos na fila em ordem crescente e, há no máximo dois valores de `v.dist` para os vértices na fila.

Análise do algoritmo - Correção

Teorema[1]

Seja G um grafo e $s \in G.V$ um vértice qualquer. Então após a execução de `Busca-em-Largura(G,s)`, $v.dist = \delta(s, v)$ para todo $v \in G.V$ e, para qualquer vértice $v \neq s$ que é alcançável de s, um caminho mínimo de s a v é o caminho mínimo de s a $v.pred$ seguido da aresta $(v.pred, v)$.

[1] http://www.ic.unicamp.br/~fkm/disciplinas/mc448/2012s1/slides/aula17

Análise do algoritmo - Correção

Teorema[1]

Seja G um grafo e $s \in G.V$ um vértice qualquer. Então após a execução de `Busca-em-Largura(G,s)`, $v.dist = \delta(s, v)$ para todo $v \in G.V$ e, para qualquer vértice $v \neq s$ que é alcançável de s, um caminho mínimo de s a v é o caminho mínimo de s a $v.pred$ seguido da aresta $(v.pred, v)$.

Ideia da demonstração

- Provar por indução que $v.dist = \delta(s, v)$. [ver demonstração alternativa no Cormen (Teorema 22.5)].
- Se $\delta(s, v) = \infty$ então $v.dist = \infty$ (Lema 2 e 3).
- Consideremos os casos em que $\delta(s, v) < \infty$.

[1] http://www.ic.unicamp.br/∼fkm/disciplinas/mc448/2012s1/slides/aula17

Análise do algoritmo - Correção

Demonstração

- **Base:** Se $v = s$ então $\delta(s, v) = s.dist = 0$.
- **Passo da indução:** Seja v um vértice com $\delta(s, v) = k$. Considere um caminho mínimo de s a v em G e chame de u o vértice que antecede v neste caminho. Suponha que $u.dist = \delta(s, u)$ para todo vértice u com $\delta(s, u) = k - 1$.

Análise do algoritmo - Correção

Considere o instante em que u foi removido da fila `Q`. O vértice $v \in u.Adj$ pode ser branco, cinza ou preto.

- **Branco:** $v.dist = u.dist + 1 = (k-1) + 1 = k$.
- **Cinza:** v já foi visitado por um vértice w ($v \in w.Adj$) e $v.dist = w.dist + 1$. Pelo Lema 3, $w.dist \leq u.dist = k - 1$ e segue que $v.dist = k$.
- **Preto:** v já passou por `Q` e pelo Lema 3, $v.dist \leq u.dist = k - 1$. Mas por outro lado, pelo Lema 2, $v.dist \geq \delta(s, v) = k$, o que é uma contradição. Este caso não ocorre.

Portanto, em todos os casos temos $v.dist = \delta(s, v)$.
Para concluir a prova do Teorema, observe que se `v.pred = u`, então $v.dist = u.dist + 1$. Portanto, podemos obter o caminho mais curto de s a v tomando o caminho mais curto de s a $v.pred$ e percorrendo a aresta $(v.pred, v)$.

Caminho mais curto

Imprime um caminho mais curto de s a v.

```
Imprime-Caminho(G,s,v)
se v = s então
   imprime s
senão
   se v.pred = NIL então
      imprime não existe caminho de s a v
   senão
      Imprime-Caminho(G,s,v.pred)
      imprime v
```

Exercícios

1. Considere os exemplos introdutórios. Esboce algumas alterações no algoritmo `Busca-em-Largura(G,s)` para resolver cada um dos problemas.
2. Existem dois tipos de lutadores profissionais: "bons sujeitos" e "maus sujeitos". Entre qualquer par de lutadores profissionais pode ou não haver uma rivalidade. Suponha que temos n lutadores profissionais e temos uma lista de r pares de lutadores para os quais existem rivalidades. Dê um algoritmo de tempo $O(n + r)$ que determine se é possível designar alguns dos lutadores como bons sujeitos e os restantes como maus sujeitos, de tal forma que a rivalidade ocorra em cada caso entre um bom sujeito e um mau sujeito. Se for possível realizar tal designação, seu algoritmo deve produzi-la. [Cormen 22.2.6]