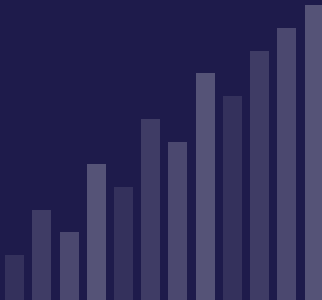

AULA 28 · BLOCO 4: GRAFOS

Grafos e busca em largura

Seções 22.1 e 22.2

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Representar grafos por listas de adjacência e por matriz de adjacência
- ▶ Executar a busca em largura e obter distâncias e a árvore de caminhos mínimos
- ▶ Provar que a BFS custa $O(V + E)$
- ▶ Medir o custo linear e o “mundo pequeno de grafos aleatórios



Teoria

Representações, BFS, distâncias, árvore de busca em largura



Prática em C

BFS em C com o grafo da Figura 22.3



Experimento

Operações = $V + 2E$; distância média em grafos aleatórios

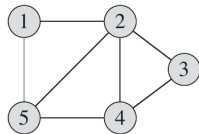


Conexões

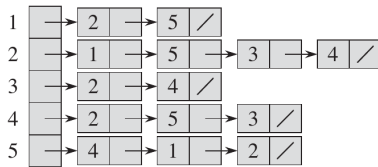
Redes sociais, web, roteamento e jogos

Duas representações

Listas: memória $\Theta(V + E)$; ótimas para grafos **esparços** ($E \ll V^2$), como quase todos os reais. **Matriz:** memória $\Theta(V^2)$; responde "existe a aresta (u, v) ?" em $O(1)$; boa para grafos **densos**.



(a)



(b)

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

Figura 22.1 de Cormen et al. (2009) — (a) grafo não direcionado; (b) listas de adjacência; (c) matriz de adjacência.

Busca em largura

- ▶ Explora em **camadas**: todos à distância k antes de qualquer um à distância $k + 1$.
- ▶ Fila FIFO (aula 12).
- ▶ Cada vértice entra na fila uma vez; cada lista é percorrida uma vez: $O(V + E)$.
- ▶ **Teorema 22.5:** $v.d = \delta(s, v)$, a distância mínima em arestas.

BFS(G, s)

```

1  para cada  $u \in G.V - \{s\}$  faça
2       $u.cor \leftarrow \text{BRANCO}; u.d \leftarrow \infty; u.\pi \leftarrow \text{NIL}$ 
3   $s.cor \leftarrow \text{CINZA}; s.d \leftarrow 0; Q \leftarrow \{s\}$ 
4  enquanto  $Q \neq \emptyset$  faça
5       $u \leftarrow \text{Dequeue}(Q)$ 
6      para cada  $v \in G.Adj[u]$  faça
7          se  $v.cor = \text{BRANCO}$  então
8               $v.cor \leftarrow \text{CINZA}; v.d \leftarrow u.d + 1; v.\pi \leftarrow u$ 
9               $\text{Enqueue}(Q, v)$ 
10      $u.cor \leftarrow \text{PRETO}$ 
    
```

BFS passo a passo

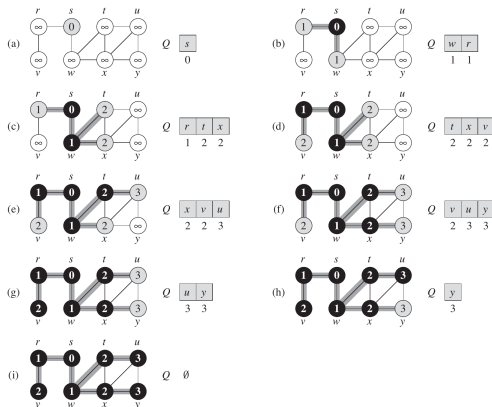


Figura 22.3 de Cormen et al. (2009) — BFS a partir de s : cinza = na fila, preto = terminado; Q mostra a fila e as



Prática em C

- ▶ Listas de adjacência com inserção na cabeça
- ▶ Fila em vetor: cada vértice entra uma vez
- ▶ Teste com o grafo da Figura 22.3 (vértices r a y)

Busca em largura em C (1/4)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /* Entrada: V E, depois E arestas não direcionadas como pares de letras (a, b, ...),
5     depois o índice da origem s (0 = primeira letra). Vértices: 'a' + i, ou as letras
6     r s t u v w x y da Figura 22.3 quando V = 8. */
7  typedef struct Aresta { int v; struct Aresta *prox; } Aresta;
8
9  int V;
10 Aresta **adj;                // adj[u]: lista de vizinhos de u
```

Busca em largura em C (2/4)

```
12 void liga(int u, int v) {                                     // insere na cabeça: O(1)
13     Aresta *a = malloc(sizeof(Aresta));
14     a->v = v;
15     a->prox = adj[u];
16     adj[u] = a;
17 }
18
19 int letra_para_vertice(char c, const char *nomes) {
20     for (int i = 0; i < V; i++) if (nomes[i] == c) return i;
21     return -1;
22 }
23
24 int main(void) {
25     int E, s;
26     if (scanf("%d %d", &V, &E) != 2) return 1;
27     const char *nomes = V == 8 ? "rstuvwxy" : "abcdefghijklmnopqrstvwxyz";
```

Busca em largura em C (3/4)

```
28 adj = calloc(V, sizeof(Aresta *));
29 for (int k = 0; k < E; k++) {
30     char a[2], b[2];
31     if (scanf("%1s %1s", a, b) != 2) return 1;
32     int u = letra_para_vertice(a[0], nomes), v = letra_para_vertice(b[0], nomes);
33     liga(u, v);
34     liga(v, u); // não direcionado: duas entradas
35 }
36 if (scanf("%d", &s) != 1) return 1;
37 int *d = malloc(V * sizeof(int)), *fila = malloc(V * sizeof(int));
38 for (int u = 0; u < V; u++) d[u] = -1; // -1 = infinito (branco, não descoberto)
39 int ini = 0, fim = 0;
40 d[s] = 0;
41 fila[fim++] = s;
42 while (ini < fim) { // BFS: fila FIFO
```

Busca em largura em C (4/4)

```
43     int u = fila[ini++];
44     for (Aresta *a = adj[u]; a; a = a->prox)
45         if (d[a->v] < 0) {                // descobre v: distância de u mais 1
46             d[a->v] = d[u] + 1;
47             fila[fim++] = a->v;
48         }
49     }
50     for (int u = 0; u < V; u++) {
51         if (d[u] < 0) printf("%c:inf ", nomes[u]);
52         else printf("%c:%d ", nomes[u], d[u]);
53     }
54     printf("\n");
55     return 0;
56 }
```



Experimento

- ▶ Operações contadas: vértices que saem da fila + entradas de adjacência
- ▶ Grafos com até 2 milhões de vértices
- ▶ Distância média com grau médio 10

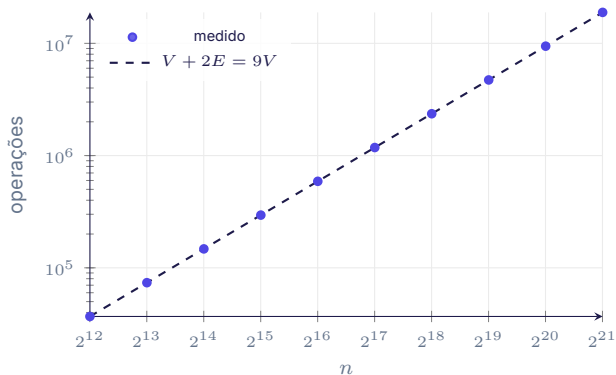
Linear em $V + E$, e o mundo é pequeno

Hipótese. Num grafo não direcionado conexo, a BFS retira cada vértice da fila uma vez e examina cada entrada de adjacência uma vez: exatamente $V + 2E$ operações. Com $E = 4V$: $9V$. Num grafo aleatório com grau médio d , as camadas crescem como d^k : a distância típica é $\approx \ln V / \ln d$, que cresce **muito devagar**.

Método

1. Listas compactas (CSR) para economizar memória
2. $V = 2^{12}$ a 2^{21} , $E = 4V$, com um caminho que garante conexidade
3. Distância: grafo aleatório com $E = 5V$ (grau médio 10)
4. Contagem exata das operações

Operações da BFS



n	medido	razão
4.096	36.864	1,000
16.384	147.456	1,000
65.536	589.824	1,000
262.144	2.359.296	1,000
1.048.576	9.437.184	1,000
2.097.152	18.874.368	1,000

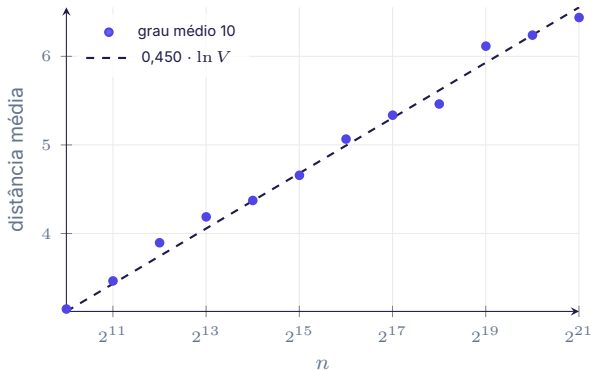
razão = medido $\div$ $V + 2E = 9V$

✓ desvio máx. 0,0%

Exatamente $9V$ em todos os tamanhos:
o custo é linear no tamanho do grafo.

🧪 EXPERIMENTO

Distância média: seis graus de separação



<i>n</i>	medido	razão
1.024	3,148	0,454
4.096	3,896	0,468
16.384	4,373	0,451
65.536	5,066	0,457
262.144	5,462	0,438
2.097.152	6,438	0,442

razão = medido $\div$ $\ln V$

✓ desvio máx. 3,1%

Dois milhões de vértices a uma distância média de 6,4. A constante ajustada fica perto de $1/\ln 10 = 0,43$.

EXPERIMENTO

O que o experimento mostrou

A BFS faz exatamente $V + 2E$ operações: **linear**. E grafos aleatórios são “pequenos”: com dois milhões de vértices e grau médio 10, a distância média é 6,4, crescendo como $\ln V$. É o fenômeno dos **seis graus de separação**, medido por BFS em redes sociais reais.

➤ Reproduza: `gcc -O2 aula28_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Redes sociais
- ▶ Web e buscadores
- ▶ Redes de computadores
- ▶ Jogos e robótica

Onde isso aparece na computação

Redes sociais

O Facebook mediu, com BFS em 721 milhões de usuários, uma distância média de 4,74; "pessoas que você talvez conheça são vértices à distância 2.

Web e buscadores

Rastreadores (*crawlers*) percorrem a web em largura a partir de páginas-semente: a BFS encontra primeiro as páginas mais próximas e relevantes.

Redes de computadores

Broadcast e descoberta de vizinhos inundam a rede camada por camada; o *traceroute* revela as camadas da BFS da internet.

Jogos e robótica

Em mapas em grade sem pesos, a BFS dá o menor caminho de um personagem ou robô; o cubo de Rubik foi resolvido ("número de Deus = 20) com buscas em largura.

Para casa

Exercícios

1. (22.1-1) Com listas de adjacência, quanto custa calcular o grau de saída e o de entrada de todos os vértices?
2. (22.1-5) Calcule o quadrado de um grafo ($u \rightarrow w$ se há caminho de 2 arestas) com listas e com matriz.
3. (22.2-1) Execute a BFS na Figura 22.2(a) a partir do vértice 3.
4. (22.2-7) Divida lutadores em "bons e "maus com rivalidades entre grupos: é testar se o grafo é bipartido. Faça em $O(V + E)$.
5. Baixe uma rede real (SNAP, Stanford) e meça a distância média com o seu código.

Leitura

Cormen et al., seções 22.1 e 22.2.

Código

josedihego.net/estruturas