Acessibilidade e Fecho Transitivo de Grafos Dirigidos

1. Fecho transitivo
2. Multiplicação de matrizes booleanas
3. Algoritmo de Warshall
4. Equivalência em termos de complexidade computacional com o produto de matrizes booleanas
5. Fecho através de busca em profundidade (grafos esparsos)

Multiplicação de matrizes

```
for (s = 0; s < V; s++)
  for (t = 0; t < V; t++)
    for (i = 0, C[s][t] = 0; i < V; i++)
      C[s][t] += A[s][i]*B[i][t];
```

Multiplicação de matrizes booleanas

```
for (s = 0; s < V; s++)
  for (t = 0; t < V; t++)
    for (i = 0, C[s][t] = 0; i < V; i++)
      if (A[s][i] && B[i][t]) C[s][t] = 1;
```

Cômputo do quadrado de matrizes booleanas

```
for (s = 0; s < V; s++)
  for (t = 0; t < V; t++)
    for (i = 0, C[s][t] = 0; i < V; i++)
      if (A[s][i] && A[i][t])
        C[s][t] = 1;
```

Conseqüência: podemos calcular o fecho transitivo em tempo $O(n^3 \log n)$!
[Calcule A^2, A^4, A^8, $A^{16}, \ldots$]

Algoritmo de Warshall para o fecho transitivo

```
for (i = 0; i < V; i++)
  for (s = 0; s < V; s++)
    for (t = 0; t < V; t++)
      if (A[s][i] && A[i][t])
        A[s][t] = 1;
```

Algoritmo de Warshall para o fecho transitivo

Propriedade 1 (Propriedade 19.7)**.** *O fecho transitivo de um grafo dirigido com n vértices pode ser computado em tempo $O(n^3)$ pelo algoritmo de Warshall.*

Prova. Asserção a ser provada por indução em i: após a i-ésima iteração do laço do i, a entrada $A[s][t]$ é 1 se e só se existe um caminho de s a t cujos vértices internos pertencem a $\{0, \ldots, i\}$. (Base: $i = -1$) $\square$

Algoritmo de Warshall para o fecho transitivo (variante)

```
for (i = 0; i < V; i++)
  for (s = 0; s < V; s++)
    if (A[s][i])
      for (t = 0; t < V; t++)
        if (A[i][t]) A[s][t] = 1;
```

Programa 19.3: Algoritmo de Warshall

```
void GRAPHtc(Graph G)
  { int i, s, t;
    G->tc = MATRIXint(G->V, G->V, 0);
    for (s = 0; s < G->V; s++)
      for (t = 0; t < G->V; t++)
        G->tc[s][t] = G->adj[s][t];
    for (s = 0; s < G->V; s++) G->tc[s][s] = 1;
    for (i = 0; i < G->V; i++)
      for (s = 0; s < G->V; s++)
        if (G->tc[s][i] == 1)
          for (t = 0; t < G->V; t++)
            if (G->tc[i][t] == 1) G->tc[s][t] = 1;
  }
int GRAPHreach(Graph G, int s, int t) { return G->tc[s][t]; }
```

Algoritmo de Warshall, complexidade

Propriedade 2 (Propriedade 19.8)**.** *Podemos implementar um teste de acessibilidade de tempo constante em grafos dirigidos com n vértices usando espaço adicional $O(n^2)$ e tempo de pre-processamento $O(n^3)$.*

Digressão: equivalência entre fecho transitivo e produto de matrizes booleanas

Propriedade 3 (Propriedade 19.9)**.** *Podemos calcular o produto AB de duas matrizes booleanas $n \times n$ em tempo $O(T_n)$, onde T_n é o tempo necessário para se calcular o fecho transitivo de um grafo dirigido com n vértices.*

Proof. A e B: duas matrizes booleanas $n \times n$. Pomos

$$M = \begin{bmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}.$$

Digressão: equivalência entre fecho transitivo e produto de matrizes booleanas (cont.)

Observamos que

$$M^2 = \begin{bmatrix} I & A & AB \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}. \qquad (1)$$

Ademais, observamos que a matriz N no lado direito de (**??**) acima é tal que $MN = N$. $\square$

Algoritmo de Floyd para distâncias (versão primitiva)

```
for (i = 0; i < V; i++)
  for (s = 0; s < V; s++)
    for (t = 0; t < V; t++)
      if (A[s][i] + A[i][t] < A[s][t])
        A[s][t] = A[s][i] + A[i][t];
```

Cômputo do fecho transitivo com BeP

```
void TCdfsR(Graph G, Edge e)
  { link t;
    G->tc[e.v][e.w] = 1;
    for (t = G->adj[e.w]; t != NULL; t = t->next)
      if (G->tc[e.v][t->v] == 0)
        TCdfsR(G, EDGE(e.v, t->v));
  }
void GRAPHtc(Graph G, Edge e)
  { int v, w;
    G->tc = MATRIXint(G->V, G->V, 0);
    for (v = 0; v < G->V; v++)
      TCdfsR(G, EDGE(v, v));
  }
int GRAPHreach(Graph G, int s, int t) { return G->tc[s][t]; }
```

Cômputo do fecho transitivo com BeP

Propriedade 4 (Propriedade 19.10). *Podemos implementar um teste de acessibilidade de tempo constante em grafos dirigidos com n vértices e m arcos usando espaço adicional $O(n^2)$ e tempo de pre-processamento $O(n(m+n))$.*

Análise empírica de desempenho

Esparso (10n arcos)					Denso (250 vértices)				
n	W	W*	A	L	m	W	W*	A	L
25	0	0	1	0	5000	289	203	177	23
50	3	1	2	1	10000	300	214	184	38
125	35	24	23	4	25000	309	226	200	97
250	275	181	178	13	50000	315	232	218	337
500	2222	1438	1481	54	100000	326	246	235	784

W Warshall W* Warshall (variante)

A BeP, matrizes de adjacência

L BeP, listas de adjacência