Programação Dinâmuca - Algoritmo de Floyd-Warshall (Cormem et al., Seção 25.2)

Programação Dinâmica - Algoritmo de Floyd-Warshall (Cormem et al., Seção 25.2)

All-Pairs Shortest-Paths problem

- **Entrada:** Grafo direcionado G com pesos $w_{i,j}$ nas arestas (podem ser negativos), sem ciclo negativo. O peso $w_{i,j} = \infty$ se não existir a aresta do vértice i para o vértice j.
- **Objetivo:** Achar menor caminho entre todo par de vértices.

Passo 1: Propriedade da Subsetrutura Ótima

- Seja G o grafo com vértices de 1 até n e imagine um menor caminho P entre dois vértices i e j em G com mais de uma aresta.
- Seja k um vértice qualquer de P. Ou seja, P vai de i a k e de k a j.
- Observe que o trecho de P de i até k é um menor caminho de i a k e o trecho de P de k até j é um menor caminho de k até j.
- Isso porque, caso contrário e houvesse um caminho menor de i até k, poderíamos obter um caminho menor de i até a j juntando o trecho de k até j. Absurdo, pois P é um caminho mínimo de i até j (não pode haver caminho menor). O mesmo argumento vale para o trecho de k até j.

Passo 2. Eq. Recorrência - Algoritmo recursivo simples

- Seja $d_k(i,j)$ o peso de um menor caminho entre vértices i e j podendo usar apenas vértices de 1 até k.
- $d_k(i,i) \;=\; 0$
- $d_0(i,j) \;=\; w_{i,j}$ (pois não pode haver intermediários)
- Para $k > 0$, testar se k participa ou não do menor caminho de i a j:

$$d_k(i,j) \;=\; \min\Big\{\, d_{k-1}(i,j),\; d_{k-1}(i,k) \;+\; d_{k-1}(k,j) \Big\}.$$

AllShortest-REC(w,i,j,k)

se $(i = j)$ **então retorne** 0
se $(k = 0)$ **então retorne** $w_{i,j}$
$a \;\leftarrow\;$ *AllShortest-REC*$(w,i,j,k-1)$
$b \;\leftarrow\;$ *AllShortest-REC*$(w,i,k,k-1)$
$c \;\leftarrow\;$ *AllShortest-REC*$(w,k,j,k-1)$
retorne $\min\{a,\; b+c\}$

Passo 2. Eq. Recorrência - Algoritmo recursivo simples

Sobreposição de Subproblemas

- **Chamada inicial**: $\mathit{AllShortest\text{-}REC}(w, i, j, n)$ para todo par $i, j \leq n$
- **Tempo**: Exponencial $\Omega(3^n)$ (para cada par i, j).
- **Indução**: $T(n) \geq 3 \cdot T(n-1) + 1$
- **Superposição**: A instância $(i, k, k-1)$ é chamada por (i, j, k) $\forall j$.

AllShortest-REC(w, i, j, k)

se $(i = j)$ **então retorne** 0
se $(k = 0)$ **então retorne** $w_{i,j}$
$a \leftarrow \mathit{AllShortest\text{-}REC}(w, i, j, k-1)$
$b \leftarrow \mathit{AllShortest\text{-}REC}(w, i, k, k-1)$
$c \leftarrow \mathit{AllShortest\text{-}REC}(w, k, j, k-1)$
retorne $\min\{a,\ b + c\}$

Passo 2b. Memoização (Alg. rec. + memória) - Top Down

AllShortest-memo(w, n, d)

para $k \leftarrow 1$ **até** n **faça**:
    para $i \leftarrow 1$ **até** n **faça**:
        para $j \leftarrow 1$ **até** n **faça**:
            $d[k, i, j] \leftarrow \infty$
para $i \leftarrow 1$ **até** n **faça**:
    para $j \leftarrow 1$ **até** n **faça**:
        $D \leftarrow \mathit{AllShortest\text{-}REC\text{-}memo}(w, i, j, n, d)$; **print** $d_{i,j} = D$

AllShortest-REC-memo(w, i, j, k, d)

se $(d[k, i, j] < \infty)$: **retorne** $d[k, i, j]$
se $(i = j)$: **retorne** 0; **se** $(k = 0)$: **retorne** $w_{i,j}$
$a \leftarrow \mathit{AllShortest\text{-}REC\text{-}memo}(w, i, j, k-1, d)$
$b \leftarrow \mathit{AllShortest\text{-}REC\text{-}memo}(w, i, k, k-1, d)$
$c \leftarrow \mathit{AllShortest\text{-}REC\text{-}memo}(w, k, j, k-1, d)$
$d[k, i, j] \leftarrow \min\{a,\ b + c\}$; **retorne** $d[k, i, j]$

Passo 3. Algoritmo p/ Valor Ótimo (Bottom-up, não rec)

$Floyd - Warshall(w, n)$

Criar matriz 3D $d[0 \ldots n, 1 \ldots n, 1 \ldots n]$
para $i \leftarrow 1$ **até** n:
    para $j \leftarrow 1$ **até** n:
        $d[0, i, j] \leftarrow w_{i,j}$
para $k \leftarrow 1$ **até** n **faça**:
    para $i \leftarrow 1$ **até** n **faça**:
        para $j \leftarrow 1$ **até** n **faça**:
            $d[k, i, j] \leftarrow d[k-1, i, j]$
            se $(d[k, i, j] > d[k-1, i, k] + d[k-1, k, j])$ **então**
                $d[k, i, j] \leftarrow d[k-1, i, k] + d[k-1, k, j]$
retorne matriz d

Tempo polinomial $\Theta(n^3)$
Espaço polinomial $\Theta(n^3)$ (memória)

Passo 3. Algoritmo p/ Valor Ótimo (Bottom-up, não rec)

$Floyd - Warshall(w, n)$

Criar matriz 2D $d[1 \dots n, 1 \dots n]$
para $i \leftarrow 1$ **até** n:
    para $j \leftarrow 1$ **até** n:
        $d[i,j] \leftarrow w_{i,j}$
para $k \leftarrow 1$ **até** n **faça**:
    para $i \leftarrow 1$ **até** n **faça**:
        para $j \leftarrow 1$ **até** n **faça**:
            $\cancel{d[i,j] \leftarrow d[i,j]}$
            se $(d[i,j] > d[i,k] + d[k,j])$ **então**
                $d[i,j] \leftarrow d[i,k] + d[k,j]$
retorne matriz d

Tempo polinomial $\Theta(n^3)$
Espaço polinomial $\Theta(n^2)$ (memória)

Passo 4. Algoritmo p/ obter Solução Ótima (Bottom-up)

$Floyd - Warshall(w, n)$

Criar matriz 2D $d[1 \ldots n, 1 \ldots n]$
para $i \leftarrow 1$ **até** n:
    para $j \leftarrow 1$ **até** n:
        $d[i,j] \leftarrow w_{i,j}$; $R[i,j] \leftarrow i$
para $k \leftarrow 1$ **até** n **faça**:
    para $i \leftarrow 1$ **até** n **faça**:
        para $j \leftarrow 1$ **até** n **faça**:
            $\cancel{d[i,j] \leftarrow d[i,j]}$
            se $(d[i,j] > d[i,k] + d[k,j])$ **então**
                $d[i,j] \leftarrow d[i,k] + d[k,j]$; $R[i,j] \leftarrow k$
retorne matrizes d e R

Tempo polinomial $\Theta(n^3)$
Espaço polinomial $\Theta(n^2)$ (memória)

Passo 4b. Algoritmo p/ escrever Soℓução Ótima (rec.)

Print-Opt(i, j, R)

print i
Print-Opt-rec(i, j, R)

Print-Opt-rec(i, j, R)

$k \leftarrow R[i, j]$
se $(k = i)$ **então** **print** j
senão
    Print-Opt(i, k, R)
    Print-Opt(k, j, R)

.
Chamada principal: *Print-Opt*(i, j, R) para qualquer par i, j.
Imprime o caminho de i a j.
Tempo $O(n)$ (pior caso: caminho hamiltoniano de tamanho n).