Problemas de caminhos em grafos e Programação Dinâmica

Claudia M. Justel

Instituto Militar de Engenharia, Rio de Janeiro, Brasil

Brazilian ICPC Summer School 2023 - Brazilian Final class

AGENDA

03/02/2023 - AULA 1: Problemas de caminhos em grafos

04/02/2023 - AULA 2: Programação Dinâmica

AULA 2 - Objetivo

Mostrar como varia a a complexidade do pior caso segundo o tipo de problema resolvido usando Programação dinâmica (PD):

- pseudo-polinomial: Coeficiente Binomial $O(nk)$, Mochila Inteiro $O(nW)$,
- polinomial: Floyd-Warshall $O(n^3)$, Árvore binária de busca de custo ótimo $O(n^3)$,
- exponencial - Caixeiro Viajante $O(2^n)$.

AULA 2

Definições:

- complexidade do pior caso polinomial - $O(f(x))$ onde x é o tamanho da entrada e f é função polinomial.
- complexidade do pior caso pseudo-polinomial - $O(f(x, w))$ onde x é o tamanho da entrada, $w = 2^{log(w)}$ é magnitude de números envolvidos na entrada e f é função polinomial.
- complexidade do pior caso exponencial - $O(f(x))$ onde x é o tamanho da entrada e f é função exponencial.

AULA 2 - PD Coeficiente Binomial

$0! = 1, \forall n > 0, n! = n(n-1)!$.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

procedimento COMB(n, k)
se $k = 0$ ou $k = n$ *então* retornar 1
senão retornar COMB($n-1, k-1$) + COMB($n-1, k$)

AULA 2 - PD Coeficiente Binomial

Para computar os coeficientes binomiais, vários valores COMB(i, j) são calculados mais de uma vez.

Se utilizarmos um tabela guardando os valores intermediários, pode-se diminuir o cálculo de valores repetidos.

procedimento COMB1(n, k)
para $i = 0, \ldots, n$ *faça* $C(i, 0) = 1$
para $j = 1, \ldots, k$ *faça* $C(0, j) = 0$
para $i = 1, \ldots, n$ *faça*
    para $j = 1, \ldots k$ *faça*
        se $i > j$ *então* $C(i, j) = C(i-1, j-1) + C(i-1, j)$
        senão se $i = j$ *então* $C(i, j) = 1$
            senão $C(i, j) = 0$
retornar $C(n, k)$

AULA 2 - PD Coeficiente Binomial

EXEMPLO: Calcular $\binom{5}{3}$ usando COMB1.
Os cálculos intermediários $C(i, j)$ durante a execução de COMB1 são mostrados na Tabela .

n/k	0	1	2	3
0	1	0	0	0
1	1	1	0	0
2	1	2	1	0
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

O valor $C(5, 3) = 10$ é a solução dada pela algoritmo para o cálculo de $\binom{5}{3}$.

AULA 2 - PD Coeficiente Binomial

Neste exemplo de cálculo de coeficiente binomial, não é necessário armazenar a tabela completa com todos os valores $C(i, j)$, $1 \leq i \leq n$ e $1 \leq j \leq k$.

Manter um vetor de dimensão k, representando a linha atual, e atualizar os valores de esquerda para direita.

Com esta forma de armazenamento, obtém-se um algoritmo com complexidade do pior caso $O(nk)$ (pseudo-polinomial). O de espaço necessário é $O(k) = O(n)$.

AULA 2 - PD Mochila inteiro

Dados n objetos e uma mochila, onde cada objeto i, $1 \leq i \leq n$ tem associado um peso w_i e um valor v_i. A mochila carrega peso máximo W. Deseja-se encher a mochila de maneira que a soma dos valores dos objetos incluídos seja máxima e seja respeitada a capacidade da mochila. Neste caso os objetos não podem ser fracionados.

O problema pode ser descrito como:

$max \sum_{i=1}^{n} x_i v_i$
sujeito a $\sum_{i=1}^{n} x_i w_i \leq W$
$x_i = 0$ ou $1, \forall 1 \leq i \leq n$.

AULA 2 - PD Mochila inteiro

Construir uma matriz $V\ (n+1) \times (W+1)$.
Se $V[i,j]$ é o máximo valor correspondente ao problema da mochila com capacidade j, considerando únicamente objetos numerados de 1 até i. Sejam

- $V[i-1,j]$: valor correspondente à solução do problema da mochila com capacidade j e objetos numerados de 1 até $i-1$, (não contém o objeto i).
- $V[i-1,j-w_i]+v_i$: valor correspondente ao problema da mochila contendo o objeto i (cujo peso é $w_i \leq j$) e escolhendo objetos numerados 1 até $i-1$ para capacidade $j-w_i \geq 0$.

Então $V[i,j] = max\{V[i-1,j]; V[i-1,j-w_i]+v_i\}$.
E a solução do problema será dada por $V[n,W]$.

AULA 2 - PD Mochila inteiro

procedimento MOCHILA INTEIRO (n, v, w, W)
para $j = 0, \ldots, W$ *faça* $V[0, j] = 0$
para $i = 1, \ldots, n$ *faça*
    $V[i, 0] = 0$
    para $j = 1, \ldots, W$ *faça*
        se $w_i \leq j$ *então*
            se $V[i-1, j-w_i] + v_i > V[i-1, j]$ *então* $V[i, j] = V[i-1, j-w_i] + v_i$
            senão $V[i, j] = V[i-1, j]$
        senão $V[i, j] = V[i-1, j]$
retornar $V[n, W]$

A complexidade do pior caso de MOCHILA INTEIRO é $O(nW)$, onde W é um dos dados que definem a instância do problema (pseudo-polinomial).

AULA 2 - PD Mochila inteiro

Para obter os objetos colocados dentro da mochila que correspondem ao valor máximo, utilizar matriz K.
Se existe objeto k tal que $K[i,j] = k$, então o objeto k foi o último objeto colocado na mochila na solução do problema $V[i,j]$.
Se não foi preenchido o valor $K[i,j]$, então o último objeto incluido do subproblema $V[i,j]$ e igual ao do subproblema $V[i-1,j]$ que se encontra no elemento $K[i-1,j]$.

procedimento MOCHILA INTEIRO1 (n, v, w, W)
para $j = 0, \ldots, W$ *faça* $V[0,j] = 0$
para $i = 1, \ldots, n$ *faça*
    $V[i,0] = 0$
    para $j = 1, \ldots, W$ *faça*
        se $w_i \leq j$ *então*
            se $V[i-1, j-w_i] + v_i > V[i-1,j]$ *então* $V[i,j] = V[i-1, j-w_i] + v_i$
                $K[i,j] = i$
            senão $V[i,j] = V[i-1,j]$
        senão $V[i,j] = V[i-1,j]$
retornar $V[n,W]$

AULA 2 - PD Mochila inteiro

EXEMPLO: Sejam $n = 5$, $W = 11$ e os valores e peso de cada objeto dados na Tabela, entrada do procedimento MOCHILA INTEIRO.

i	v_i	w_i
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

AULA 2 - PD Mochila inteiro

Os cálculos intermediários $V[i, j]$ durante a execução de MOCHILA INTEIRO são mostrados a seguir .

i/j	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1	1
2	0	1	6	7	7	7	7	7	7	7	7	7
3	0	1	6	7	7	18	19	24	25	25	25	25
4	0	1	6	7	7	18	22	24	28	29	29	40
5	0	1	6	7	7	18	22	28	29	34	35	40

O valor $V[5, 11] = 40$ é a solução dada pela algoritmo para o valor ótimo carregado na mochila, repeitando a capacidade e sem objetos fracionados, utilizando os objetos 3 e 4.

AULA 2 - PD Mochila inteiro

Os valores $K[i, j]$ obtidos durante a execução de MOCHILA INTEIRO são mostrados a seguir .

i/j	0	1	2	3	4	5	6	7	8	9	10	11
0												
1		1	1	1	1	1	1	1	1	1	1	1
2			2	2	2	2	2	2	2	2	2	2
3						3	3	3	3	3	3	3
4								4		4	4	4
5								5	5	5	5	

Partindo do valor $K[5, 11]$ podem ser identificados os objetos que correspondem ao valor mínimo obtido pelo algoritmo.

AULA 2 - PD Árvore binária de busca de custo ótimo

O problema consiste em, dado $S = \{s_1, ..., s_n\}$ conjunto de chaves, $s_1 < ... < s_n$ determinar em quais posições da árvore armazenar as chaves de maneria que a soma do número de comparações total, ou seja o número de comparações para acessar cada chave, seja o menor possível.

Seja T uma árvore binária de busca para $S = \{s_1, ..., s_n\}$ conjunto de chaves.

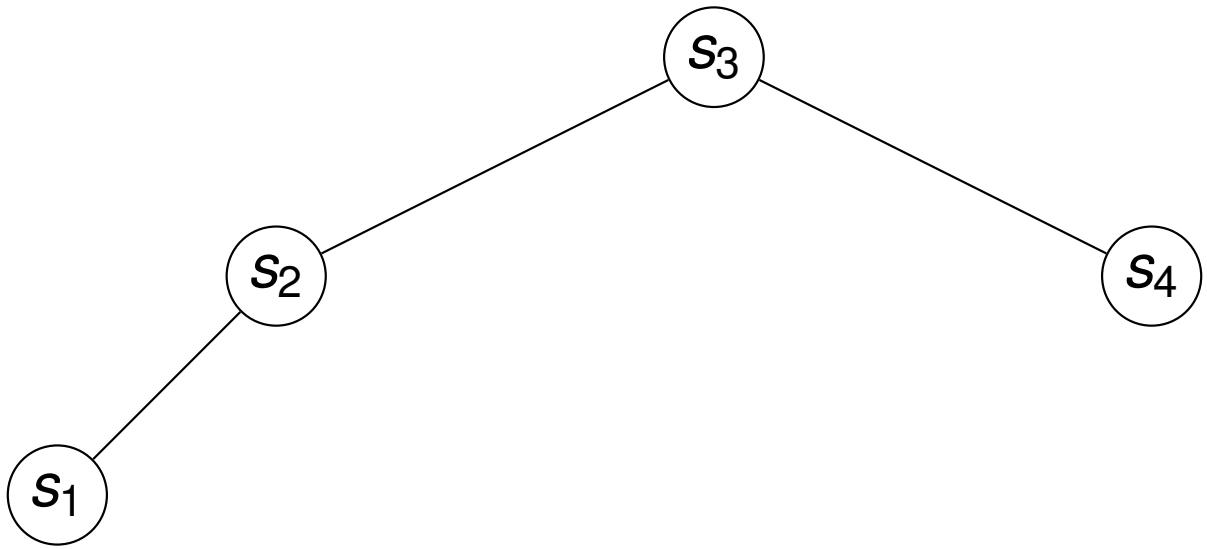
HIPÓTESE 1: frequência de acesso aos nós idêntica

i. Buscas com sucesso

Comprimento do caminho interno da árvore binária de busca T: $I(T) = \sum_{1 \leq k \leq n} l_k$, onde l_k é o nível do nó com chave s_k (número de comparações necessárias para o acesso à chave s_k).

AULA 2 - PD Árvore binária de busca de custo ótimo

OBSERVAÇÃO: nível da raiz de T é igual a 1 neste caso.



$n = 4$
$l(T) = 1 + 2 + 2 + 3 = 8$

AULA 2 - PD Árvore binária de busca de custo ótimo

ii. Buscas sem sucesso

Seja S um conjunto de chaves $S = \{s_1, s_2, ..., s_n\}$,
$s_1 < s_2 < \cdots < s_n$.
As chaves são alocadas nos nós de uma árvore binária de busca T.
Para cada chave s_k, seja l_k o nível que corresponde a s_k em T.
O conjunto de números reais pode ser dividido em:

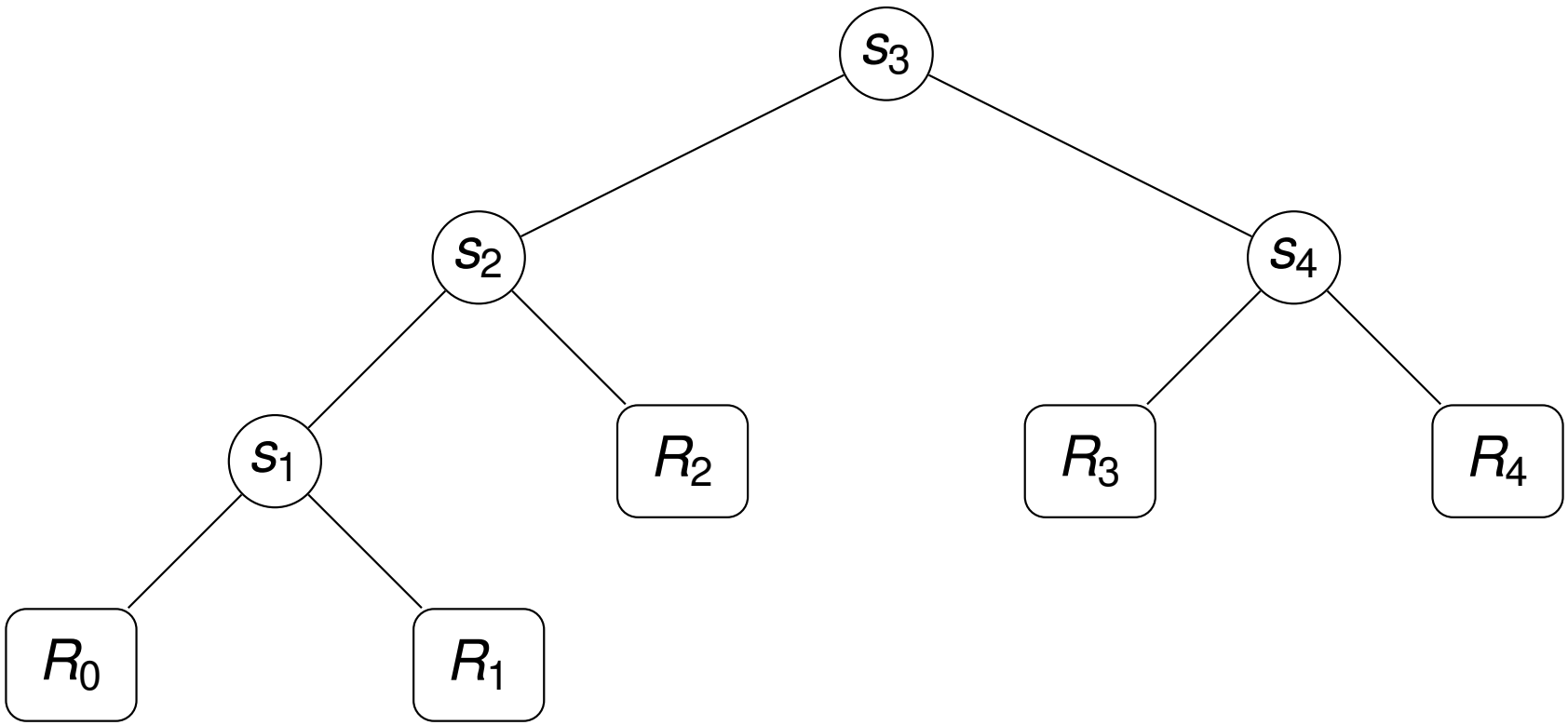
$$(-\infty, s_1), s_1, (s_1, s_2), s_2, ..., s_{n-1}, (s_{n-1}, s_n), s_n, (s_n, \infty)$$

Notamos por:
$R_0 = (-\infty, s_1)$,
$R_i = (s_i, s_{i+1}), 1 \leq i \leq n-1$,
$R_n = (s_n, \infty)$.

R_k, $0 \leq k \leq n$ são os nós externos representam as saídas possíveis das buscas sem sucesso na árvore binária de busca T.

AULA 2 - PD Árvore binária de busca de custo ótimo

Comprimento do caminho externo da árvore binária de busca T: $E(T) = \sum_{0 \leq k \leq n}(l'_k - 1)$, onde l'_k é o nível do nó externo R_k (número de comparações necessárias determinar que a chave s_k não está em T).



$E(T) = 3 + 3 + 2 + 2 + 2 = 12$

AULA 2 - PD Árvore binária de busca de custo ótimo

Consideramos agora
HIPÓTESE 2: frequência de acesso aos nós diferenciada

T árvore binária de busca.
S um conjunto de chaves $S = \{s_1, s_2, ..., s_n\}$,
$s_1 < s_2 < \cdots < s_n$ com freqüências de acesso $f_1, \ldots f_n$.

As chaves são alocadas nos nós de T.

$\forall\, 1 \leq p \leq n$, $s_p \rightarrow$ freqüência de acesso $= f_p$, nível que corresponde a s_p em $T = l_p$.

$R_0, R_1, \ldots, R_n$ são os nós externos com freqüências de acesso $f'_0, f'_1, \ldots f'_n$.

$\forall\, 0 \leq p \leq n$, $R_p \rightarrow$ com freqüência de acesso $= f'_p$, nível correspondente em $T = l'_p$.

AULA 2 - PD Árvore binária de busca de custo ótimo

Define-se

Comprimento do caminho interno ponderado da árvore binária de busca T: $\sum_{1 \leq p \leq n} l_p f_p$
(número total de comparações necessárias para todas as possibilidades de busca com sucesso em T considerando as chaves com frequëncias de acesso diferenciadas) (buscas com sucesso).

Comprimento do caminho externo ponderado da árvore binária de busca T: $\sum_{0 \leq p \leq n} (l'_p - 1) f'_p$
(número total de comparações necessárias para todas as possibilidades de busca sem sucesso em T considerando as chaves com frequëncias de acesso diferenciadas) (busca sem sucesso).

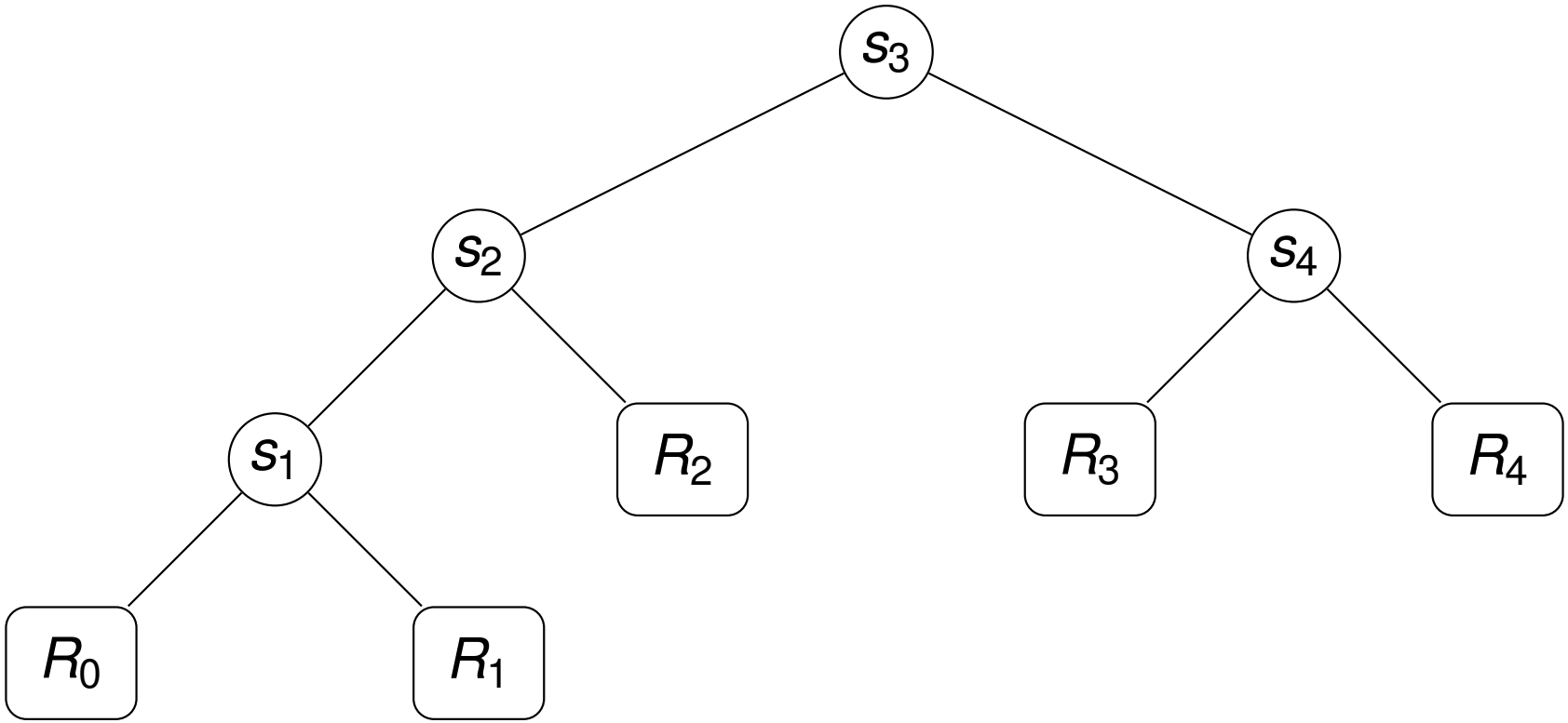
AULA 2 - PD Árvore binária de busca de custo ótimo

Dada uma árvore binária de busca T nas condiçõesda Hipótese 2, $c(T)$ é o número total de comparações necessárias para todas as possiblidades de busca, com e sem sucesso:

$$c(T) = \sum_{1 \leq p \leq n} l_p f_p + \sum_{0 \leq p \leq n} (l'_p - 1) f'_p.$$

AULA 2 - PD Árvore binária de busca de custo ótimo

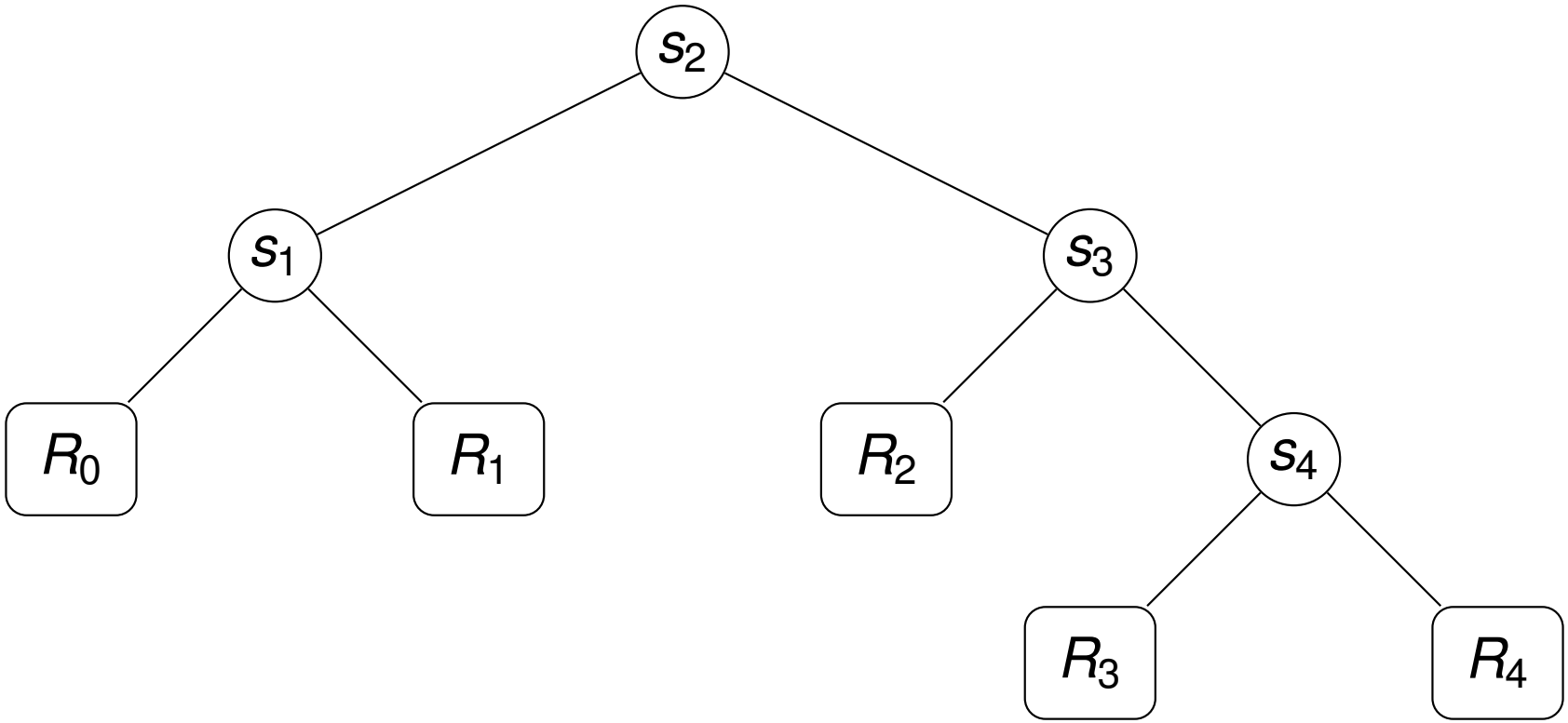
Seja $n = 4$ chaves com freqüências
$f_1 = 10, f_2 = 1, f_3 = 3, f_4 = 2, f'_0 = 2, f'_1 = 1, f'_2 = 1, f'_3 = 1, f'_4 = 1.$
T



$$C(T) = (10.3+1.2+3.1+2.2)+(2.3+1.3+1.2+1.2+1.2) = 54.$$

AULA 2 - PD Árvore binária de busca de custo ótimo

Exemplo: $n = 4$ chaves com freqüências
$f_1 = 10, f_2 = 1, f_3 = 3, f_4 = 2, f'_0 = 2, f'_1 = 1, f'_2 = 1, f'_3 = 1, f'_4 = 1.$
T'



$$C(T') = (10.2+1.1+3.2+2.3)+(2.2+1.2+1.2+13+1.3) = 47.$$

AULA 2 - PD Árvore binária de busca de custo ótimo

Dados n, as frequencias $f_p, 1 \leq p \leq n$ e $f'_p, 0 \leq p \leq n$, podemos obter uma árvore binária de busca T para as chaves $s_1 < s_2 < \ldots, s_n$ que apresente menor custo $C(T)$ possível?

Lema 1:
As subárvores de uma árvore binária de busca ótima são também ótimas.

Seja $T(i,j)$ a árvore ótima correspondente ao subconjunto de chaves $\{s_{i+1}, \ldots, s_j\}$, $0 \leq i \leq j \leq n$.
Seja $F(i,j)$ a soma de todas as freqüências correspondentes a $T(i,j)$: $F(i,j) = \sum_{i<p\leq j} f_p + \sum_{i\leq p\leq j} f'_p$.

Lema 2:
Seja $T(i,j)$ a árvore binária de busca ótima de raiz s_k correspondente as chaves $\{s_{i+1}, \ldots, s_j\}$, $0 \leq i \leq j \leq n$. Então $C(T(i,j)) = C(T(i,k-1)) + C(T(k,j)) + F(i,j)$.

AULA 2 - PD Árvore binária de busca de custo ótimo

ENTRADA: n e $f_1, \ldots, f_n, f'_0, \ldots, f'_n$.

SAÍDA: $C(0, n)$

procedimento ARVORE BINARIA OTIMA 1(n, f, f', C)
para $j = 0, \ldots, n$ *faça*
    $C[j, j] = 0$, $F[j, j] = f'_j$
para $d = 1, \ldots, n$ *faça*
    para $i = 0, \ldots, n - d$ *faça*
        $j = i + d$
        $F[i, j] = F[i, j - 1] + f_j + f'_j$
        $C[i, j] = min_{i<k\leq j}\{C[i, k - 1] + C[k, j]\} + F[i, j]$

A complexidade de ARVORE BINARIA OTIMA 1 é $O(n^3)$.

AULA 2 - PD Árvore binária de busca de custo ótimo

Para construir a árvore binária com custo ótimo deve ser armazenado o valor de k que efetiviza o mínimo no cálculo de $C[i,j]$ na entrada $K[i,j]$ da matriz K.

procedimentoARVORE BINARIA OTIMA 2(n, f, f', C, K)
para $j = 0, \ldots, n$ *faça*
    $C[j,j] = 0$, $F[j,j] = f'_j$
para $d = 1, \ldots, n$ *faça*
    para $i = 0, \ldots, n-d$ *faça*
        $j = i + d$
        $F[i,j] = F[i,j-1] + f_j + f'_j$
        $C[i,j] = min_{i<k\leq j}\{C[i,k-1] + C[k,j]\} + F[i,j]$
        $K[i,j] = k$

AULA 2 - PD Árvore binária de busca de custo ótimo

EXEMPLO: Seja $n = 4$, determinar a árvore binária de busca de custo ótimo para as freqüências:
$f_1 = 10, f_2 = 1, f_3 = 3, f_4 = 2,$
$f'_0 = 2, f'_1 = 1, f'_2 = 1, f'_3 = 1, f'_4 = 1.$

No final de ARVORE BINARIA OTIMA 2 obtemos as matrizes C F e K. O custo da árvore binária de busca ótima e $C(0,4) = 39$ e a raiz da mesma é a chave s_1 $(= K(0,4))$.

$$C = \begin{pmatrix} 0 & 13 & 18 & 29 & 39 \\ & 0 & 3 & 10 & 17 \\ & & 0 & 5 & 12 \\ & & & 0 & 4 \\ & & & & 0 \end{pmatrix} \quad F = \begin{pmatrix} 2 & 13 & 15 & 19 & 22 \\ & 1 & 3 & 7 & 10 \\ & & 1 & 5 & 8 \\ & & & 1 & 4 \\ & & & & 1 \end{pmatrix}$$

$$K = \begin{pmatrix} - & 1 & 1 & 1 & 1 \\ - & - & 2 & 3 & 3 \\ - & - & - & 3 & 3 \\ - & - & - & - & 4 \\ - & - & - & - & - \end{pmatrix}$$

AULA 2 - PD Floyd-Warshall

Seja $G = (V, E)$, $\forall\, e \in E, w(e) \in \mathbb{R}$,.

procedimento FLOYD1(L, D, n)
para $i = 1, \ldots, n$ *faça*
    para $j = 1$ ate n *faça* $D^0[i, j] = L[i, j]$
para $k = 1, \ldots, n$ *faça*
    para $i = 1, \ldots, n$ *faça*
        para $j = 1, \ldots, n$ *faça*
            se $D^{k-1}[i, k] + D^{k-1}[k, j] < D^{k-1}[i, j]$ *então*
                $D^k[i, j] = D^{k-1}[i, k] + D^{k-1}[k, j]$
            senão $D^k[i, j] = D^{k-1}[i, j]$
retornar D^n

A complexidade do algoritmo de Floyd é $O(n^3)$.

Quando existir ciclo de comprimento negativo num grafo, o algoritmo de Floyd-Warshall não determina corretamente a solução do problema de caminhos mínimos entre todos os pares de vértices.

AULA 2 - PD Caixeiro Viajante

Algoritmo de Programação Dinâmica para o Problema do Caixeiro Viajante (TSP) a seguir.

$G = (V, E)$, $n = |V|$, vértices numerados de 1 até n,
$w(i, j) \geq 0, \forall (i, j) \in E$.
C a matriz n^2, $C_{i,j} = w(i, j)$, $(i, j) \in E$.

Algoritmo usando programação dinâmica para TSP.
CUSTO-MINIMO: determina o comprimento do ciclo hamiltoniano mínimo.
MELHOR-CH permite reconstruir sequência de vértices no ciclo hamiltoniano de custo mínimo.

AULA 2 - PD Caixeiro Viajante

procedimento TSP-PD(n, C)
para $i = 2, \ldots, n$ *faça*
    CUSTO$[\{i\}, i] = C_{1,i}$; MELHOR-CAMINHO$[\{i\}, i] = (1, i)$
para $j = 2, \ldots, n-1$ *faça*
    para $S \subseteq \{2, \ldots, n\}$ tal que $|S| = j$ *faça*
        para $i \in S$ *faça*
            CUSTO$[S, i] = min_{k \in S - \{i\}} \{\text{CUSTO}[S - \{i\}, k] + C_{k,i}\}$
            seja k o vertice que atinge o minimo
            MELHOR-CAMINHO$[S, i]$ =MELHOR-CAMINHO $[S - \{i\}, k]//i$
CUSTO-MINIMO $= min_{k \neq 1} \{CUSTO[\{2, \ldots, n\}, k] + c_{k,1}\}$
Seja k o vertice que atinge o minimo
MELHOR-CH = MELHOR-CAMINHO$[\{2, \ldots, n\}, k]//1$

AULA 2 - PD Caixeiro Viajante

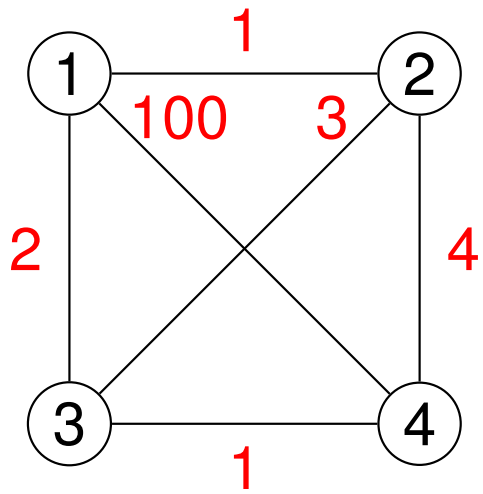
A complexidade do pior caso de TSP-PD é dada por:

$$\sum_{j=2}^{n-1} \binom{n-1}{j} c\, j^2 \leq c[2^{n-1}(n-1)^2] = O(n^2 2^n),$$

onde c é uma constante positiva.

AULA 2 - PD Caixeiro Viajante

EXEMPLO: Seja $G = K_4$



$$C = (c_{i,j}) = \begin{bmatrix} 0 & 1 & 2 & 100 \\ 1 & 0 & 3 & 4 \\ 2 & 3 & 0 & 1 \\ 100 & 4 & 1 & 0 \end{bmatrix}.$$

AULA 2 - PD Caixeiro Viajante

S	i	CUSTO[S, i]	MELHOR-CAMINHO[S, i]
$\{1\}$	1	0	$\emptyset$
$\{2\}$	2	1	1, 2
$\{3\}$	3	2	1, 3
$\{4\}$	4	100	1, 4
$\{2, 3\}$	2	5 ($k = 3$)	1, 3, 2
$\{2, 3\}$	3	4 ($k = 2$)	1, 2, 3
$\{2, 4\}$	2	104 ($k = 4$)	1, 4, 2
$\{2, 4\}$	4	5 ($k = 2$)	1, 2, 4
$\{3, 4\}$	3	101 ($k = 4$)	1, 4, 3
$\{3, 4\}$	4	3 ($k = 3$)	1, 3, 4
$\{2, 3, 4\}$	2	7 ($k = 4$)	1, 3, 4, 2
$\{2, 3, 4\}$	3	6 ($k = 4$)	1, 2, 4, 3
$\{2, 3, 4\}$	4	5 ($k = 3$)	1, 2, 3, 4

CUSTO-MINIMO $= min_{k \neq 1} \{\text{CUSTO}[\{2, 3, 4\}, k]\} + c_{k,1} = 8$

$k = 2$, CUSTO$[\{2, 3, 4\}, 2]\} + c_{2,1} = 7 + 1 = 8$

$k = 3$, CUSTO$[\{2, 3, 4\}, 3]\} + c_{3,1} = 6 + 2 = 8$

$k = 4$, CUSTO$[\{2, 3, 4\}, 4]\} + c_{4,1} = 5 + 100 = 105$

Para $k = 2$, MELHOR-CAMINHO$[\{2, 3, 4\}, 2] = 1, 3, 4, 2$. MELHOR-CH$_1 = 1, 3, 4, 2, 1$, $\sum_{(i,j) \in CH_1} c_{i,j} = 8$.

Para $k = 3$, MELHOR-CAMINHO$[\{2, 3, 4\}, 3] = 1, 2, 4, 3$. MELHOR-CH$_2 = 1, 2, 4, 3, 1$, $\sum_{(i,j) \in CH_2} c_{i,j} = 8$.

AULA 2 - Resumo Fórmulas PD usadas

Problema	Algoritmo Prog. Dinâmica	tam. entrada	complex. pior caso	fórmula
coeficiente binomial	COMB1(n, k)	n, k	$O(n.k)$	$C(n,k) = C(n-1,k-1) + C(n-1,k)$
mochila inteiro (não fraccionar)	MOCHILA INTEIRO (n, v, w, W)	n, W	$O(n.W)$	$V[i,j] =$ $max\{V[i-1,j]; V[i-1,j-w_i] + v_i\}$
caminhos mínimos (todos os pares)	FLOYD(L, D, n)	n	$O(n^3)$	$D^k[i,j] =$ $min\{D^{k-1}[i,j]; D^{k-1}[i,k] + D^{k-1}[k,j]\}$
árvore binária de busca ótima	ARVORE BINARIA OTIMA (n, f, f', C)	n	$O(n^3)$	$C(T(i,j)) =$ $C(T(i,k-1)) + C(T(k,j)) + F(i,j)$
caixeiro viajante	TSP-PD(n, C)	n	$O(n^2 2^n)$	CUSTO$[S,i] = min_{k \in S-\{i\}}$ $\{$CUSTO$[S-\{i\},k] + C_{k,i}\}$

Bibliografia

Cormen, Lieserson, Rivest. *An Introduction to Algorithms*, MIT Press (1990). ISBN 0-262-53091-0.

Brassard, Bratley. *Fundamental of Algorithms*, Prentice Hall (1996). ISBN 0-13-335068-1.

Dasgupta, Papadimitriou, Vazirani. *Algorithms*, McGraw-Hill Higher Education (2008). ISBN 978-0-07-352340-8.

Garey, Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*, Freeman (1979). ISBN 0-7167-1044-7.

Bibliografia

Kleinberg, J., Tardos, E. *Algorithm Design*, Addison Wesley (2006) ISBN-10 0-321-29535-8.

Szwactfiter. *Teoria Computacional de Grafos*, Elsevier, (2018). ISBN 978-85-352-8885-8.

Szwarcfiter, Markenzon. *Estruturas de Dados e seus Algoritmos*, LTC (2010). ISBN 85-2161-750-X.

https://www.geeksforgeeks.org/dynamic-programming/?ref=lbp