Aula 21: Freqüências de Acesso Diferenciadas

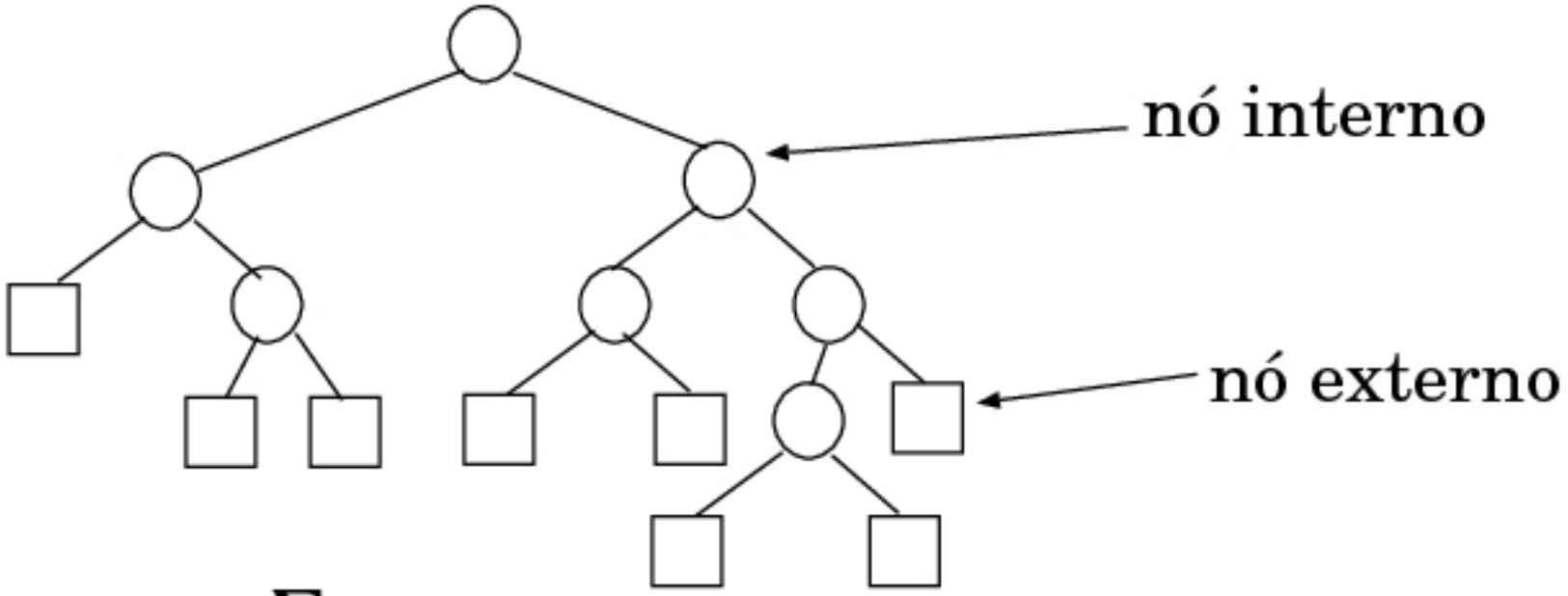
- Caminhos interno e externo ponderados
- Cálculo do custo da busca com frequências diferenciadas
- Programação dinâmica: princípios

Recordação

Recordando comprimento de caminhos interno e externo:

- T = árvore binária de busca
- I(T) = comprimento de caminho interno de T =
  = número total de passos para acessar todas as chaves (buscas com sucesso)
- E(T) = comprimento de caminho externo de T =
  = número total de passos que terminam em cada um dos nós externos de T (buscas sem sucesso)

Recordação



$$I(T) = \sum_{1 \le i \le n} l_i$$

$$E(T) = \sum_{0 \le i \le n} (l'_i - 1)$$

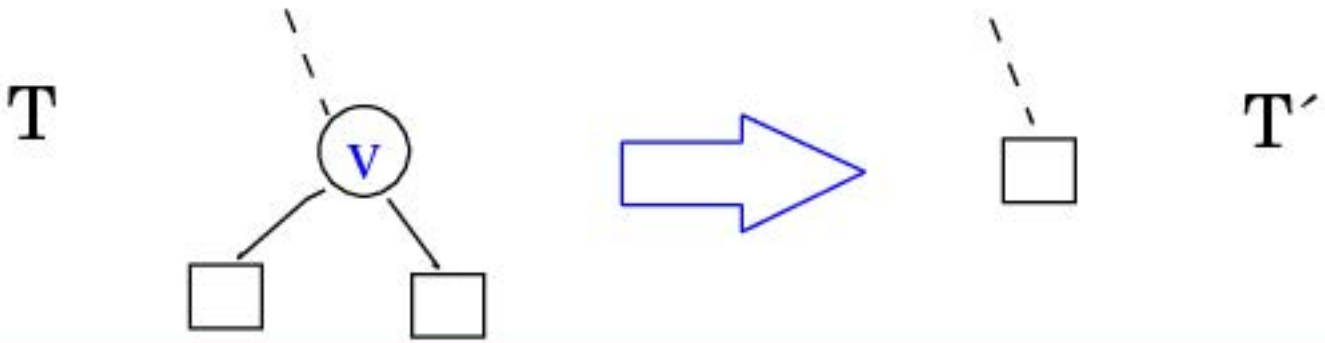
l_i = nível da chave s_i em T

l'_i = nível do intervalo R_i em T

Relação entre I(T) e E(T)

- Lema: $E(T) = I(T) + n$

- Prova: Indução em n. Se n=1, I(T)=1 e E(T)=1+1, vale.
  Se n>1, supor o lema verdadeiro para árvores com até n-1 nós internos. Seja v um nó interno, pai de dois externos.
  Seja T´ a árvore obtida pela remoção de v e seus dois filhos, nós externos, e conseqüente inclusão de um nó externo, em lugar de x.



Relação entre I(T) e E(T)

- Pela hipótese de indução,

$$E(T') = I(T')+n-1 \qquad (1)$$

- Representando por l_v o nível de v em T,

$$I(T') = I(T)-l_v \qquad (2)$$

- Por outro lado,

$$E(T') = E(T)-l_v-1 \qquad (3)$$

- Por (1), (2) e (3):

$$E(T) = I(T)+n$$

Freqüências de Acesso

- Motivação: Em geral, algumas chaves podem ser mais acessadas do que outras.
- $S = \{s_1, ..., s_n\}$, conjunto de chaves s_i
- f_i = freqüência de acesso da chave s_i =
  = número de vezes em que s_i é acessada
- T = árvore binária de busca para S

Freqüências de Acesso

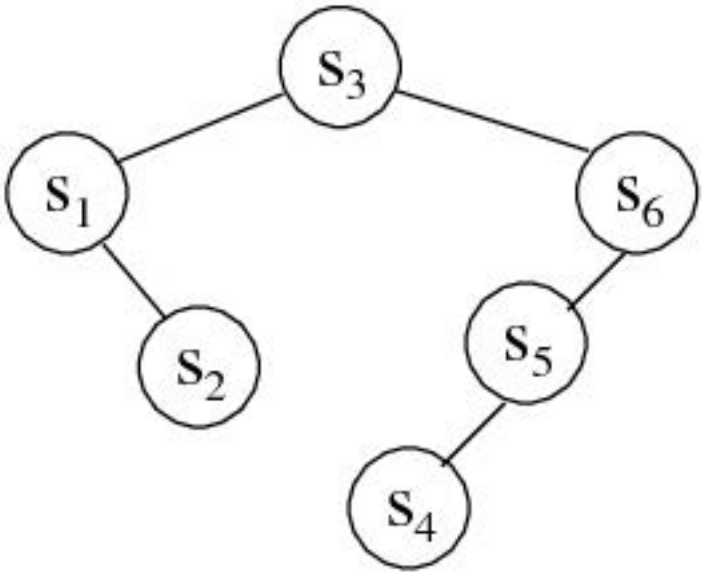
- Buscas com sucesso

- O número total de passos para acessar f_i vezes a chave s_i é $f_i.l_i$, onde l_i é o nível de s_i em T.

- $\sum_{1 \leq i \leq n} f_i . l_i$ = número total de passos para todas as buscas com sucesso
  = comprimento de caminho interno ponderado de T
  = $I_p(T)$

Exemplo

i	1	2	3	4	5	6
f'_i	2	5	0	1	2	3

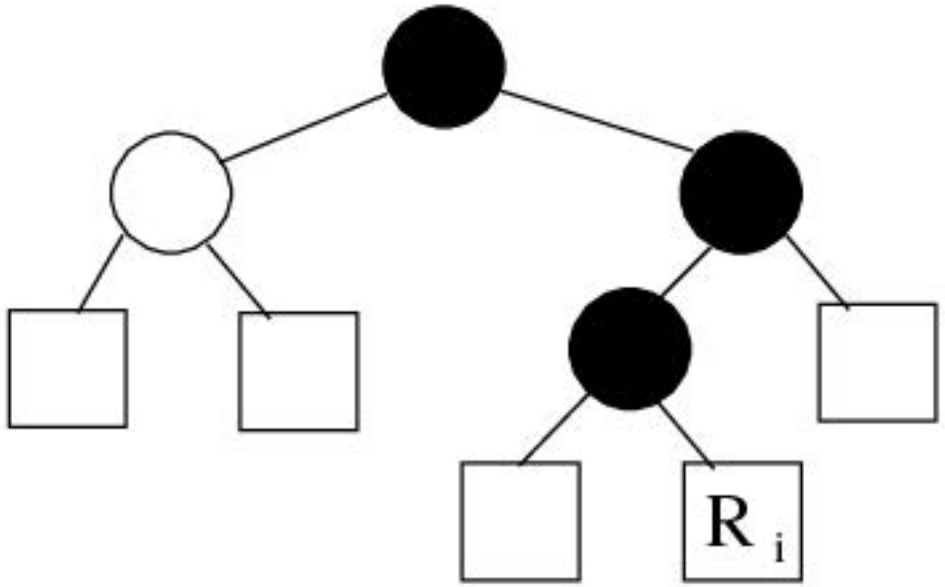


$E_p(T) = 2.2 + 5.3 + 0.1 + 1.4 + 2.3 + 3.2 = 35$

Buscas sem Sucesso

- $S = \{ s_1, ..., s_n \}$ = conjunto de chaves
- $R_0, ..., R_n$ = intervalos das ausências de chaves
- T = árvore binária de busca para S
- f_i' = freqüência de acesso de R_i =
  = número de vezes em que a busca termina no nó externo R_i
- O número total de passos consumidos nas buscas que terminam em R_i é igual a $f_i' . (l_i' - 1)$, onde l_i' é o nível de R_i em T.

Buscas sem Sucesso

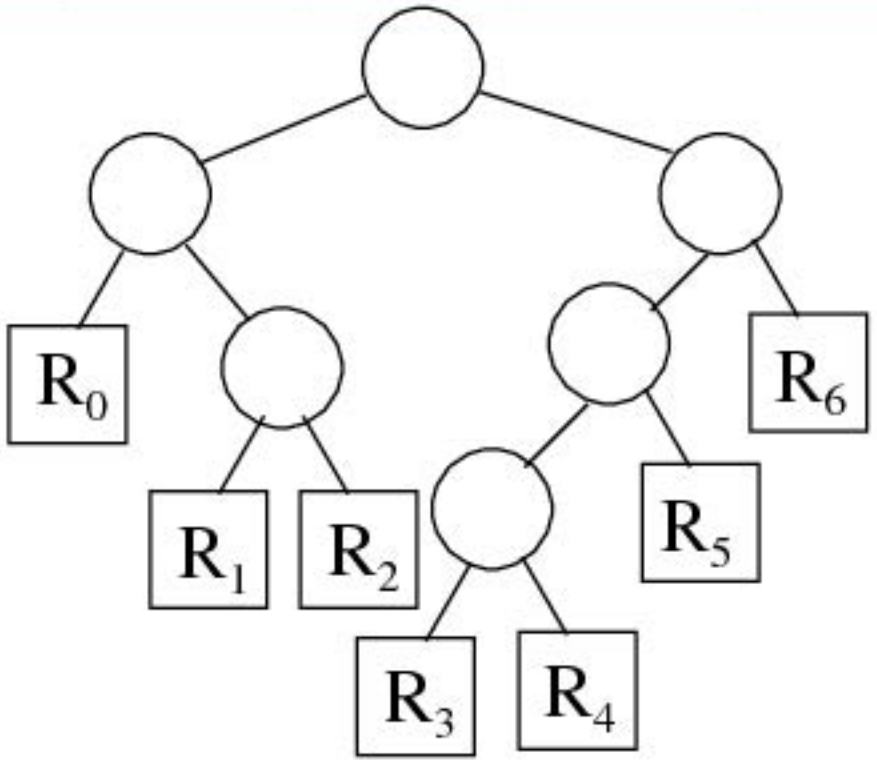


$\Rightarrow \sum_{0<=i<=n} f_i' \cdot (l_i' - 1)$ = número total de passos para todas as buscas sem sucesso =

= comprimento de caminho externo ponderado de T = $E_p(T)$

Exemplo

i	0	1	2	3	4	5	6
f'_i	2	3	0	1	1	0	2



$E_p(T) = 2.2 + 3.3 + 0.3 + 1.4 + 1.4 + 0.3 + 2.2 = 25$

Custo de uma Árvore

- $S = \{ s_1, ..., s_n \}$ = conjunto de chaves
- $R_0, ..., R_n$ = intervalos das chaves
- f_i = freqüência de s_i
- f_i' = freqüência de R_i
- T = árvore binária de busca para S
- l_i = nível de s_i em T
- l_i' = nível de R_i em T

Custo de uma Árvore

$$I_p(T) = \sum_{1 \le i \le n} f_i \cdot l_i \; ; \; E_p(T) = \sum_{0 \le i \le n} f_i' \cdot (l_i' - 1)$$

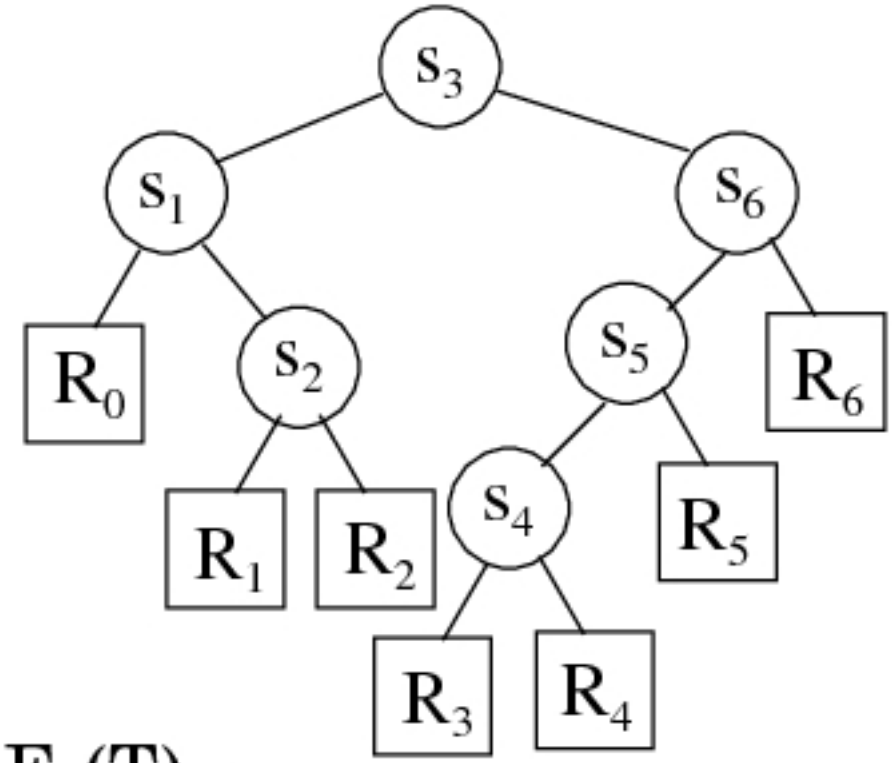
Custo de T = número total de passos em todas as buscas realizadas (com e sem sucesso) =

$= I_p(T) + E_p(T) =$

$= c(T)$

Exemplo

i	0	1	2	3	4	5	6
f_i	-	2	5	0	1	2	3
f_i'	2	3	0	1	1	0	2



$c(T) = I_p(T) + E_p(T) =$

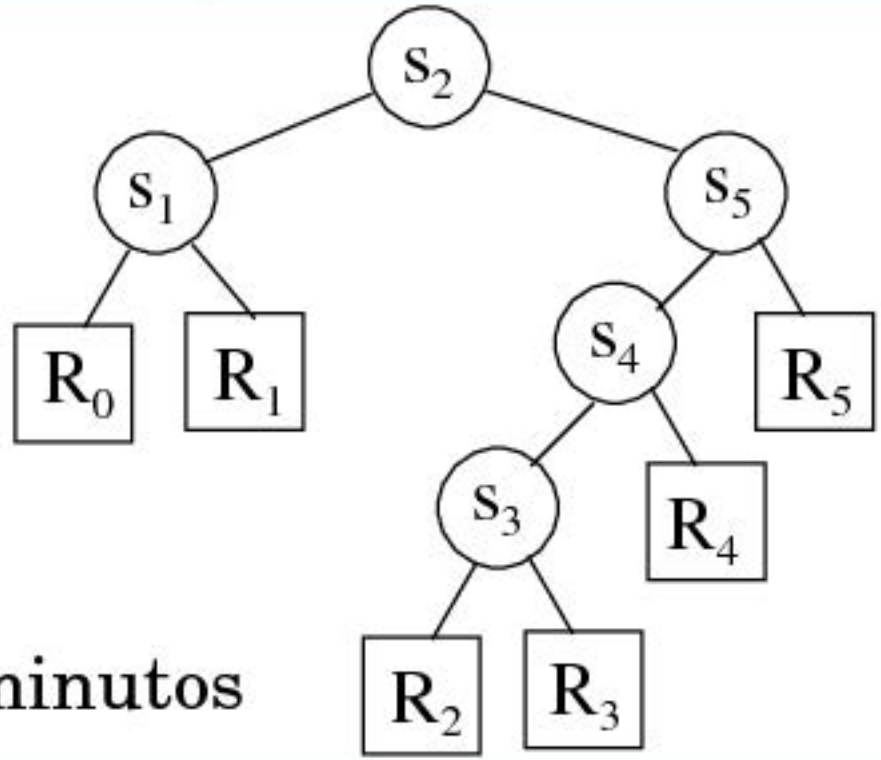
$= 2.2 + 5.3 + 1.0 + 1.4 + 2.3 + 3.2 + 2.2 + 3.3 + 0.3$

$+ 1.4 + 1.4 + 0.3 + 2.2 = 60$

Exercício

Determinar $I_p(T)$, $E_p(T)$ e c(T) para a seguinte árvore binária de busca T, sabendo-se que as freqüências de acesso são as seguintes:

i	0	1	2	3	4	5
f_i	-	1	3	2	1	2
f_i'	2	2	1	0	1	2



Tempo: 2 minutos

Solução

$I_p(T) = 1.2 + 3.1 + 2.4 + 1.3 + 2.2 = 20$

$E_p(T) = 2.2 + 2.2 + 1.4 + 0.4 + 1.3 + 2.2 = 19$

$c(T) = I_p(T) + E_p(T) = 20 + 19 = 39$

Árvore Ótima

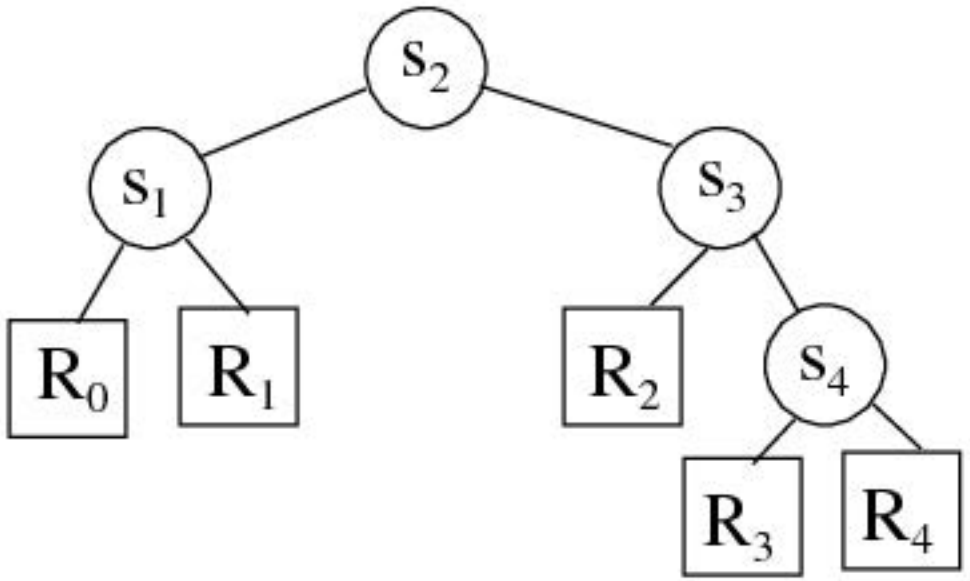
- Uma árvore binária de busca que possui custo mínimo é denominada árvore ótima.

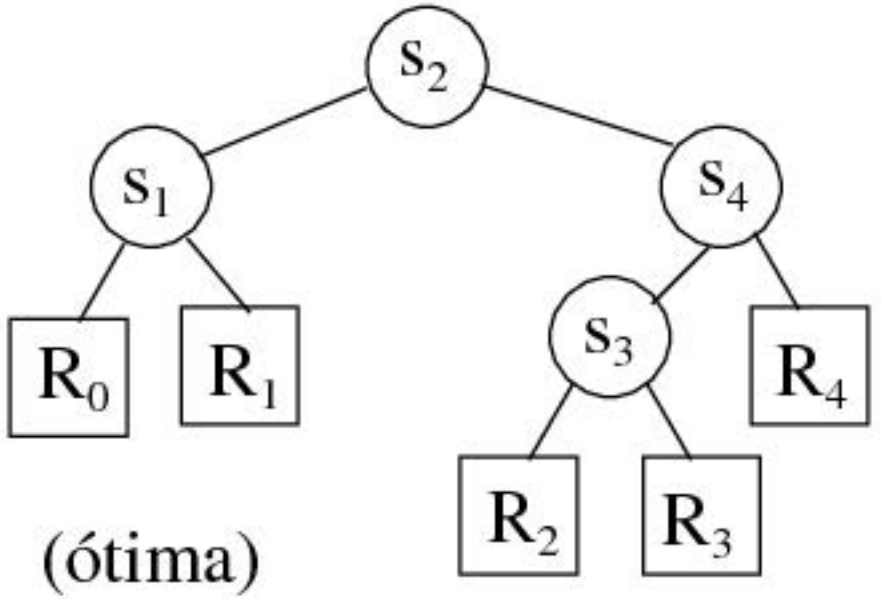
- A árvore ótima minimiza o número total de passos.

Árvore Ótima

Exemplo:

i	0	1	2	3	4
f_i	-	2	3	1	2
f_i'	1	1	1	1	1





(ótima)

$c(T) = 2.2 + 3.1 + 1.2 + 2.3 + 1.2 + 1.2 + 1.2 + 1.3 + 1.3 = 27$

$c(T) = 2.2 + 3.1 + 1.3 + 2.2 + 1.2 + 1.2 + 1.3 + 1.3 + 1.2 = 26$

Problema

- Dado um conjunto de chaves $S = \{ s_1, ..., s_n \}$ e as freqüências f_1', ..., f_n' determinar a árvore binária de busca ótima para S.

- Para resolver este problema será utilizada uma técnica denominada programação dinâmica.

Programação Dinâmica

- Seja P um problema a ser resolvido.
- Decompor P em subproblemas menores P_1, ..., P_k onde cada P_i é da mesma natureza que P, porém aplicada a um conjunto de dados menor.
- O método utiliza uma tabela com uma entrada para cada subproblema. Quando o subproblema P_i é resolvido pela primeira vez, a sua solução é anotada na tabela, na posição correspondente. Nas vezes subseqüentes em que P_i for considerado, ao invés de iniciar novas computações para resolvê-lo, a sua solução é simplesmente obtida dessa tabela

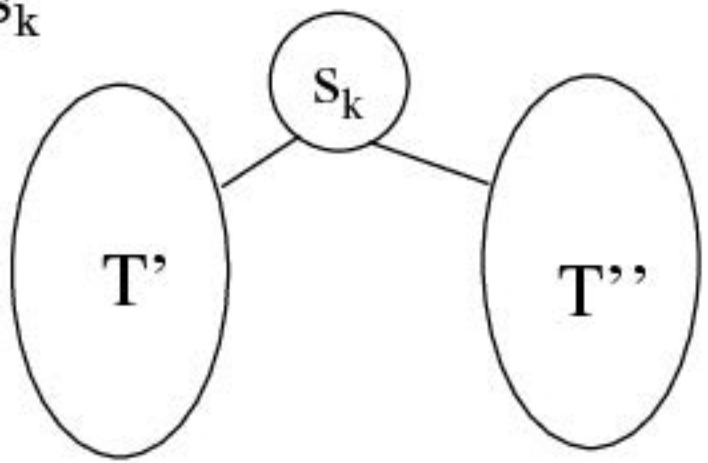
Programação Dinâmica

- Para resolver P_i pela primeira vez, utiliza-se este próprio método, de maneira recursiva.

- Para obter a solução de P, a partir da solução dos subproblemas P_1, ..., P_k, determina-se uma expressão que relacione a solução de P com a dos seus subproblemas P_1, ..., P_k. Normalmente esta expressão é formulada através de relações de recorrência.

Decomposição da Árvore de Busca Ótima

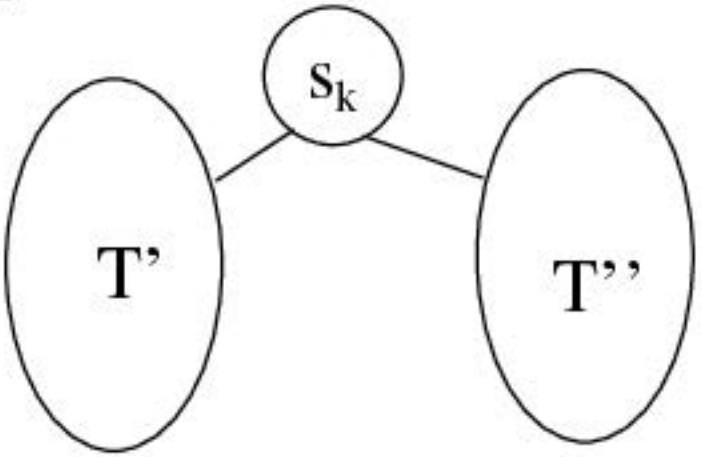
- T = árvore binária de busca ótima para { s_1, ..., s_n }
- s_k = raiz de T
- T' = subárvore esquerda de s_k
- T'' = subárvore direita de s_k



- T' = árvore binária de busca para { s_1, ..., s_{k-1} }
- T'' = árvore binária de busca para { s_{k+1}, ..., s_n }

Decomposição da Árvore de Busca Ótima

- Lema: As subárvores de uma árvore binária de busca ótima são também ótimas.
- Prova: se assim não fosse, a substituição de uma árvore não ótima pela ótima correspondente implicaria uma diminuição do custo da árvore ótima, uma contradição.



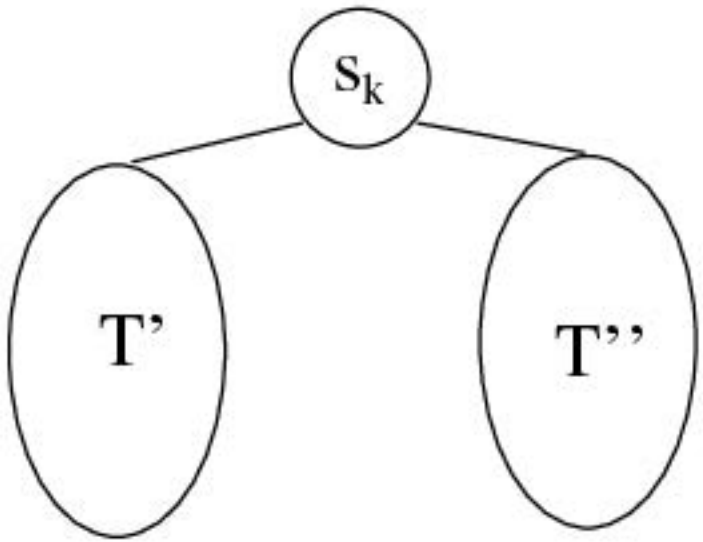
- Conseqüência: T' é uma árvore ótima para $\{ s_1, ..., s_{k-1} \}$

T'' é uma árvore ótima para $\{ s_{k+1}, ..., s_n \}$

Questões a resolver

- (i) como conhecer a raiz da árvore ótima, s_k ?

(ii) como determinar T' e T'' ?



- Respostas: (i) tentar todas as possiblidades (são n nós)
(ii) utilizar este método, de forma recursiva.

Subproblemas

- Cada subproblema corresponde a determinar a árvore binária de busca ótima, para um subconjunto de chaves consecutivas $\{ s_{i+1}, ..., s_j \}$, $0 \leq i \leq j \leq n$. Seja $T(i, j)$ esta árvore ótima.
- Para determinar a árvore ótima $T(i, j)$, constrói-se a árvore ótima para $\{ s_{i+1}, ..., s_j \}$, supondo que a raiz seja a chave s_k, para cada chave s_k, $i < k \leq j$. A árvore ótima será a de menor custo dentre elas.
- Resta encontrar uma relação entre os custos de uma árvore e os custos das subárvores, esquerda e direita, de sua raiz.

Subproblemas (cont.)

- T(i, j) = árvore ótima correspondente ao subconjunto de chaves $\{ s_{i+1}, \ldots, s_j \}$
- $F(i, j) = \sum_{i < k \leq j} f_k + \sum_{i \leq k \leq j} f_k'$
- Lema: seja T(i, j) a árvore ótima para as chaves $\{ s_{i+1}, \ldots, s_j \}$, e denote por s_k a raiz de T(i, j).

  Então

  $c (T (i, j)) = c (T (i, k-1)) + c (T (k, j)) + F (i, j)$
- A prova se baseia nas seguintes observações:
  - subárvores de uma árvore ótima são ótimas;
  - se T' é uma subárvore da raiz de uma árvore T e s_j é um nó de T' então a contribuição de s_j em c(T) é igual à contribuição de s_j em c(T') mais f_j'.

Exercício Final

Determinar a árvore ótima para o conjunto de chaves $\{ s_1, ..., s_n \}$, $s_i < s_{i+1}$, onde

$$f_i = 2. f_{i-1}, \quad 1 < \quad i \leq n, \text{ e}$$

$$f_i' = 0, \qquad 0 \leq i \leq n.$$