SCC 211 - Laboratório de Algoritmos Avançados

Soluções de Problemas de Maratona

- Optimal Binary Search Tree (UVa 10304)

prof. João Luís
Monitor Raphael Ferras
Universidade de São Paulo - São Carlos

2011

Optimal Binary Search Tree

UVa 10304

Descrição:

Dado um Conjunto **S** = (**e**$_1$, **e**$_2$, ... , **e**$_n$) de **n** elementos distintos de forma que **e**$_1$ < **e**$_2$ < .... < **e**$_n$ e considerando uma Árvore de Busca Binária dos elementos de **S**. É desejado que quanto maior a frequência de um elemento mais próximo da raiz ele esteja.

Optimal Binary Search Tree

UVa 10304

Descrição:

O custo de acesso de um elemento **e_i** do conjunto **S** (**cost(ei)**) é o número de nós percorridos da raiz até o elemento.

Dadas as Frequências de cada elemento do conjunto S, (f(e1), f(e2), ... f(e3)), dizemos que o custo total da Árvore é dado pela seguinte somatória:

$f(e_1)*cost(e_1)+ f(e_2)*cost(e_2)+ .... f(e_n)*cost(e_n)$

Optimal Binary Search Tree

UVa 10304

Entrada e Saída:

A entrada possuirá alguns casos de teste, um por linha.

Cada linha começará com um **n** indicando o tamanho de S, seguido por **n** números inteiros indicando a frequência de cada elemento **ei**. A entrada termina com um EOF.

Para cada Entrada você deve imprimir o total do custo da Optimal Binary Search Tree.

Optimal Binary Search Tree

UVa 10304

Exemplo de Entrada:

```
1 5
3 10 10 10
3 5 10 20
```

Exemplo de Saída:

```
0
20
20
```

Optimal Binary Search Tree

UVa 10304

Exemplo de Entrada:

```
1 5
3 10 10 10
3 5 10 20
```

cost(e1)*f(e1) =
5 * 0 =

0

Nivel 0

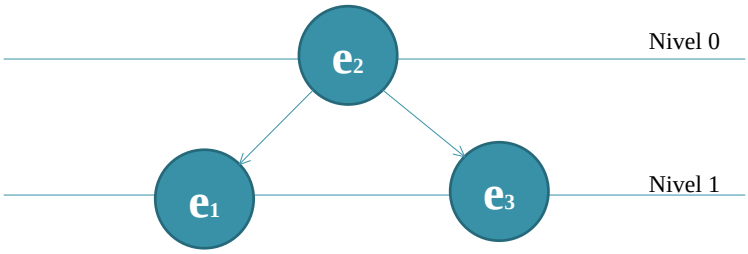
Optimal Binary Search Tree

UVa 10304

Exemplo de Entrada:

```
1 5
3 10 10 10
3 5 10 20
```

cost(e1)*f(e1) +
cost(e2)*f(e2) +
cost(e3)*f(e3) =
10 * 1 + 10* 0 +
10 * 1 =
20



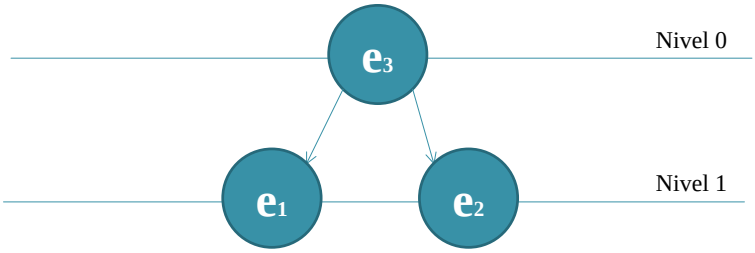
Optimal Binary Search Tree

UVa 10304

Exemplo de Entrada:

```
1 5
3 10 10 10
3 5 10 20
```

cost(e_1)*f(e_1) +
cost(e_3)*f(e_3) +
cost(e_2)*f(e_2) =
5 * 1 + 20* 0 +
10 * 1 =
15



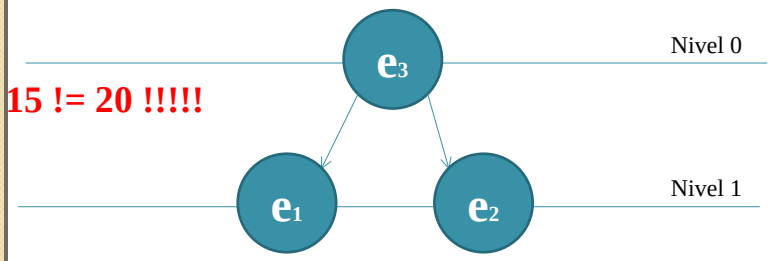
Optimal Binary Search Tree

UVa 10304

Exemplo de Entrada:

```
1 5
3 10 10 10
3 5 10 20
```

cost(e_1)*f(e_1) +
cost(e_3)*f(e_3) +
cost(e_2)*f(e_2) =
5 * 1 + 20* 0 +
10 * 1 =
15



15 != 20 !!!!!

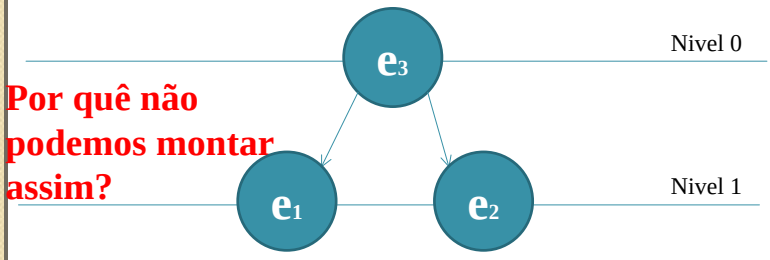
Optimal Binary Search Tree

UVa 10304

Exemplo de Entrada:

```
1 5
3 10 10 10
3 5 10 20
```

cost(e_1)*f(e_1) +
cost(e_3)*f(e_3) +
cost(e_2)*f(e_2) =
5 * 1 + 20* 0 +
10 * 1 =
15

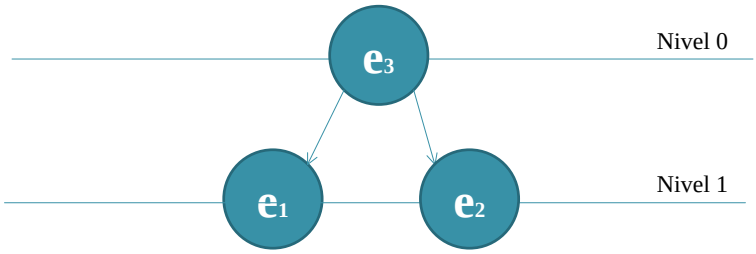


Por quê não podemos montar assim?

Optimal Binary Search Tree

UVa 10304

Em uma Árvore de busca binária os elementos da esquerda devem ser menores que o elemento atual, e os elementos da direita maiores, para que seja possível uma busca por um elemento!



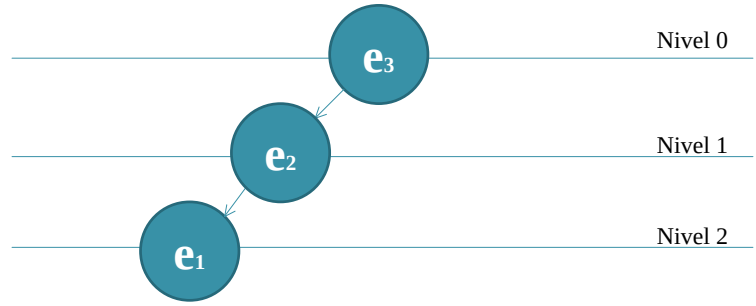
Optimal Binary Search Tree

UVa 10304

Exemplo de Entrada:

```
1 5
3 10 10 10
3 5 10 20
```

cost(e1)*f(e1) +
cost(e2)*f(e2) +
cost(e3)*f(e3) =
5 * 2 + 10* 1 +
20 * 0 =
20



Optimal Binary Search Tree

UVa 10304

Para resolver esse problema, se fossemos listar todas as possíveis soluções e ver qual é a de menor custo entre elas:

Com 3 elementos teríamos 5 árvores distintas.

Optimal Binary Search Tree

UVa 10304

Com 5 elementos teríamos 42 árvores distintas.

Com 10 elementos teríamos 16.796 arvores distintas.

Como o limite total é 250...

Optimal Binary Search Tree

UVa 10304

Com 250 elementos teríamos 46.511.679.596.923.379.649.774.794.725.966.780.740.729.116.008.092.209.611.195.332.652.514.387.519.365.925.7831.340.309.862.635.877.995.262.413.955.019.878.805.418.475.969.029.457.769.094.808.256 árvores distintas.

Aproximadamente 10 ^ 150 árvores distintas. Provavelmente TLE!

Optimal Binary Search Tree

UVa 10304

Como resolver???

Optimal Binary Search Tree

UVa 10304

Como resolver???

Programação Dinâmica!!!

Optimal Binary Search Tree

UVa 10304

Resolver problemas menores e então partir para problemas mais complexos.

Optimal Binary Search Tree

UVa 10304

Resolver problemas menores e então partir para problemas mais complexos.

Como quebrar uma árvore em problemas menores????

Optimal Binary Search Tree

UVa 10304

Suponhamos que tivessemos uma entrada

S = (e_1, e_2, e_3, e_4, e_5, e_6, e_7) com frequência de S, (10, 10, 10, 50, 5,10 , 20).

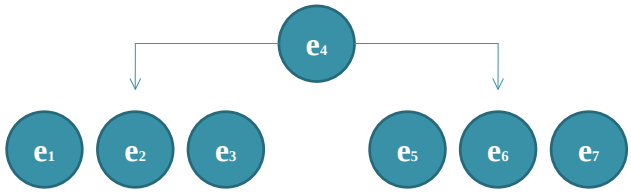
Optimal Binary Search Tree

UVa 10304

Suponhamos que tivessemos uma entrada

S = (e_1, e_2, e_3, e_4, e_5, e_6, e_7) com frequência de S, (10, 10, 10, 20, 5,10 , 20).

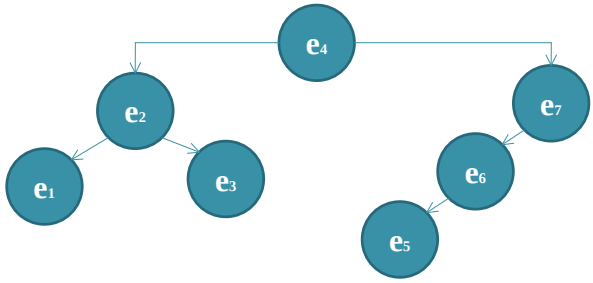
Se soubessemos que e_4 é a raiz, que problemas teríamos que resolver?



Optimal Binary Search Tree

UVa 10304

Subproblemas de uma árvore também são uma árvore!!! Como as sub-árvores são independentes e já foram calculadas no exemplo de entrada. Podemos dizer que a melhor solução dessa árvore será....



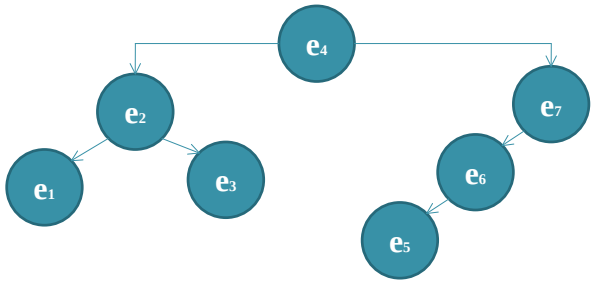
Optimal Binary Search Tree

UVa 10304

Frequência de S, (10, 10, 10, 20, 5,10 , 20).

20*0 + PD(10,10,10)+ PD(5,10,20)

= 0 + 20 + 20 = 40!



Optimal Binary Search Tree

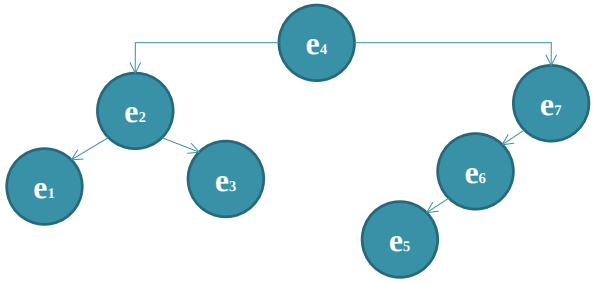
UVa 10304

Frequência de S, (10, 10, 10, 20, 5,10 , 20).

20*0 + PD(10,10,10)+ PD(5,10,20)

= 0 + 20 + 20 = 40!

ERRADO! Precisamos considerar que todos os nós desceram um nível...



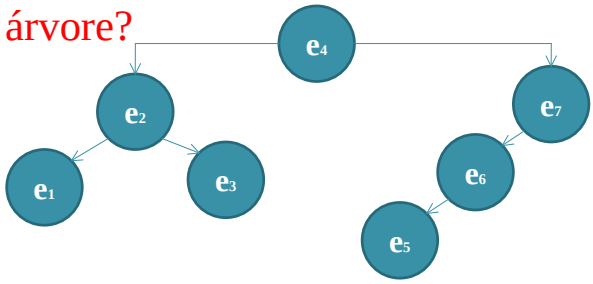
Optimal Binary Search Tree

UVa 10304

Frequência de S, (10, 10, 10, 20, 5,10 , 20).

20*0 + PD(10,10,10)+ PD(5,10,20)

= 0 + 20 + 20 = 40! ERRADO! Precisamos considerar que todos os nós desceram um nível... Para isso somamos quanto na árvore?

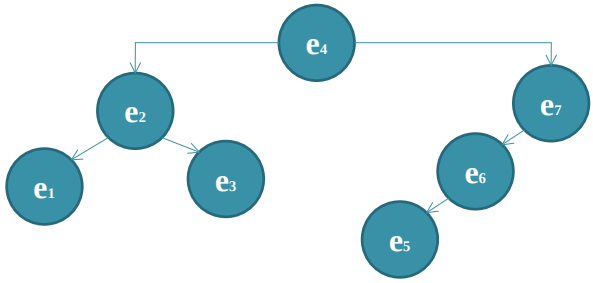


Optimal Binary Search Tree

UVa 10304

Se antes PD(10,10,10) é 20. Supondo que não saibamos o custo que cada um tinha então chamaremos de c_1,c_2 e c_3

$f(e_1)*c_1+f(e_2)*c_2+f(e_3)*c_3 = 20$



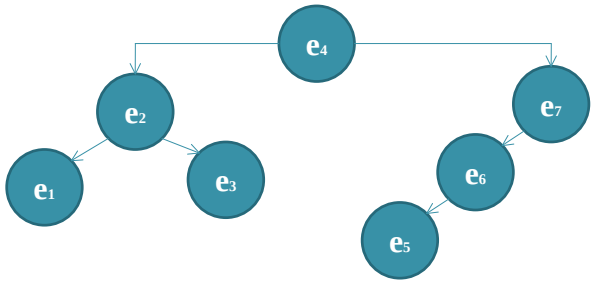
Optimal Binary Search Tree

UVa 10304

Chamaremos de K a diferença no custo. Se somamos um no custo de cada nó:

$f(e_1)*(c_1+1)+f(e_2)*(c_2+1)+f(e_3)*(c_3+1) = 20+K$

$f(e_1)*c_1+f(e_1)+f(e_2)*c_2+f(e_2)+f(e_3)*c_3+ f(e_3)= 20+K$



Optimal Binary Search Tree

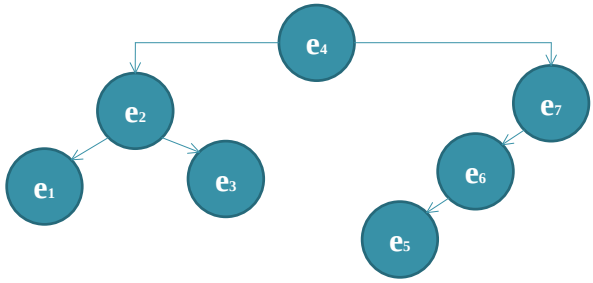
UVa 10304

Subtraindo a equação inicial teremos...

$f(e_1)*c_1+f(e_1)+f(e_2)*c_2+f(e_2)+f(e_3)*c_3+ f(e_3)= 20+K$

$-(f(e_1)*c_1+f(e_2)*c_2+f(e_3)*c_3 = 20)$

Teremos que $K = f(e_1)+ f(e_2)+f(e_3)$. Portanto o custo da árvore deslocada um nível para baixo aumenta segundo a soma de todos os valores dos elementos deslocados.



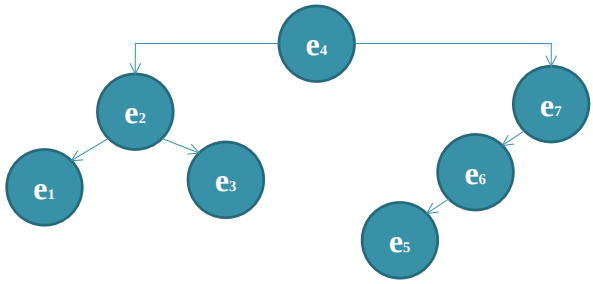
Optimal Binary Search Tree

UVa 10304

Frequência de S, (10, 10, 10, 20, 5,10 , 20).

20*0 + PD(10,10,10)+(f(e1)+...+f(e3))+
PD(5,10,20) + (f(e4)+...+f(e6))

= 0+20+20+30+35 = 105!



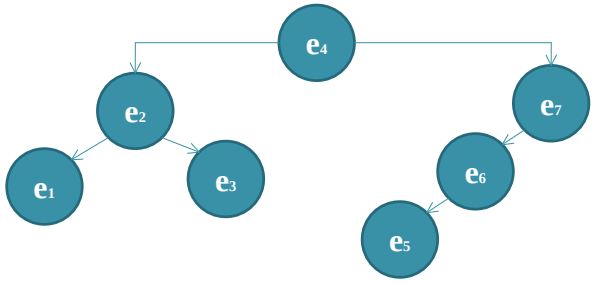
Optimal Binary Search Tree

UVa 10304

Frequência de S, (10, 10, 10, 20, 5,10 , 20).

20*0 + PD(10,10,10)+(f(e1)+...+f(e3))+
PD(5,10,20) + (f(e4)+...+f(e6))

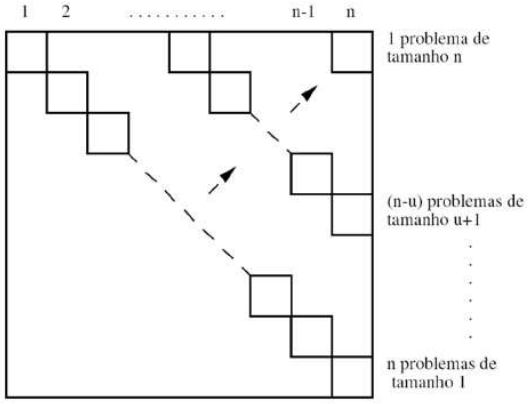
= 0+20+20+30+35 = 105!



Optimal Binary Search Tree

UVa 10304

Como visto em aula, uma forma de memorizar as soluções é com uma tabela **n** x **n**.



Optimal Binary Search Tree

UVa 10304

Para facilitar os cálculos da soma de uma sequência de frequências utilizaremos um vetor com o acumulado das somas.

Por exemplo, com frequência de S, (10, 10, 10, 20, 5,10 , 20).

i	1	2	3	4	5	6	7
Acumulado das somas de f(e_i)	10	20	30	50	55	65	85

Optimal Binary Search Tree

UVa 10304

Por exemplo, com frequência de S, (10, 10, 10, 20, 5,10 , 20).

Para calcular as somas de e_2 até e_6 basta calcular: Acumulado(6) – Acumulado(1)

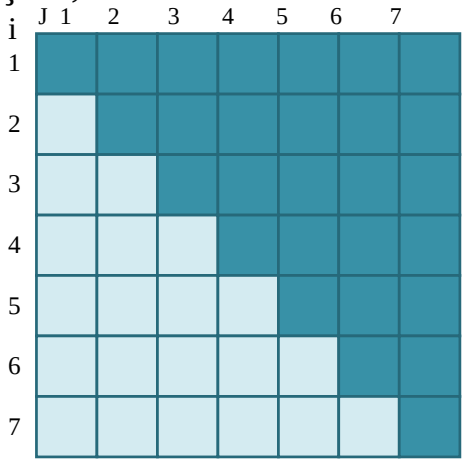
65 – 10 = 10+10+20+5+10.

i	1	2	3	4	5	6	7
Acumulado das somas de f(e_i)	10	20	30	50	55	65	85

Optimal Binary Search Tree

UVa 10304

Montaremos uma Matriz para memorizar as soluções, mas utilizaremos somente metade.



Optimal Binary Search Tree

UVa 10304

Na posição [i,j] estará calculado o melhor custo de uma árvore dos elementos de e_i até e_j, quando i = j, $m_{i,j} = 0$.

i \ J	1	2	3	4	5	6	7
1	0						
2		0					
3			0				
4				0			
5					0		
6						0	
7							0

Optimal Binary Search Tree

UVa 10304

Na posição [i,j] estará calculado o melhor custo de uma árvore dos elementos de e_i até e_j, quando i = j, $m_{i,j} = 0$.

Para cada outro caso testa-se todas as possíveis raízes entre i e j.

Preenche-se a matriz diagonal por diagonal até o índice $m_{1,7}$

i \ J	1	2	3	4	5	6	7
1	0	10	20				
2		0	10	30			
3			0	10	15		
4				0	5		
5					0	5	
6						0	10
7							0

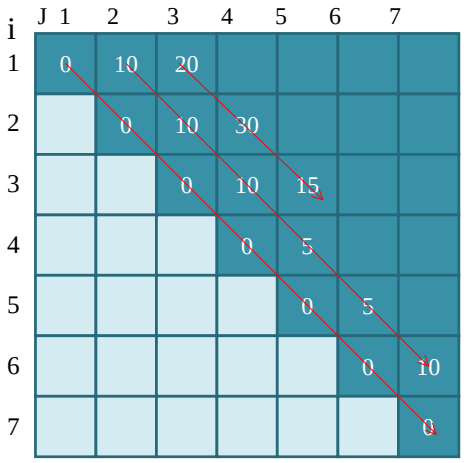
Optimal Binary Search Tree

UVa 10304

Na posição [i,j] estara calculado o melhor custo de uma arvore dos elementos de e_i até e_j, quando i = j, $m_{i,j} = 0$.

Para cada outro caso testa-se todas as possíveis raízes entre i e j.

Preenche-se a matriz diagonal por diagonal até o índice $m_{1,7}$



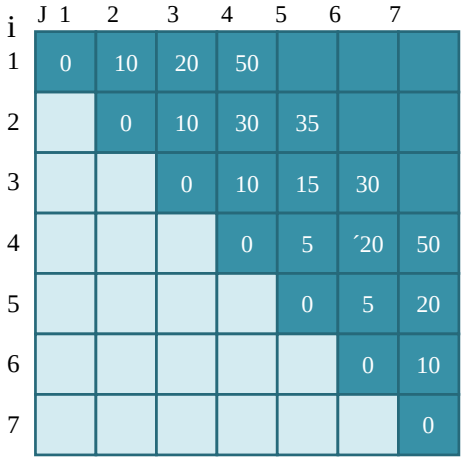
Optimal Binary Search Tree

UVa 10304

Na posição [i,j] estara calculado o melhor custo de uma arvore dos elementos de e_i até e_j, quando i = j, $m_{i,j} = 0$.

Para cada outro caso testa-se todas as possíveis raízes entre i e j.

Preenche-se a matriz diagonal por diagonal até o índice $m_{1,7}$



Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60
Raiz no 3
M1,2 + M4,5 + A1,2 + A3,5 =

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60
Raiz no 3
M1,2 + M4,5 + A1,2 + A3,5 =
10 + 5 + 20 + 25 = 60

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60
Raiz no 3
M1,2 + M4,5 + A1,2 + A4,5 =
10 + 5 + 20 + 25 = 60
Raiz no 4
M1,3 + M5,5 + A1,3 + A5,5 =

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60
Raiz no 3
M1,2 + M4,5 + A1,2 + A4,5 =
10 + 5 + 20 + 25 = 60
Raiz no 4
M1,3 + M5,5 + A1,3 + A5,5 =
20 + 0 + 30 + 5 = 55

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60
Raiz no 3
M1,2 + M4,5 + A1,2 + A4,5 =
10 + 5 + 20 + 25 = 60
Raiz no 4
M1,3 + M5,5 + A1,3 + A5,5 =
20 + 0 + 30 + 5 = 55

Raiz no 5
M1,4 + A1,4 =

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60
Raiz no 3
M1,2 + M4,5 + A1,2 + A4,5 =
10 + 5 + 20 + 25 = 60
Raiz no 4
M1,3 + M5,5 + A1,3 + A5,5 =
20 + 0 + 30 + 5 = 55
Raiz no 5
M1,4 + A1,4 =
50 + 45 = 95

i \ J	1	2	3	4	5	6	7
1	0	10	20	50			
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Exemplo de Calculo do M1,5
Testa todas as possíveis raizes entre 1 e 5.
Raiz no 1
M2,5 + A2,5 =
35 + 45 = 80
Raiz no 2
M1,1 + M3,5 + A1,1+ A3,5 =
0 + 15 + 45 = 60
Raiz no 3
M1,2 + M4,5 + A1,2 + A4,5 =
10 + 5 + 20 + 25 = 60

Raiz no 4
M1,3 + M5,5 + A1,3 + A5,5 =
20 + 0 + 30 + 5 = 55

Raiz no 5
M1,4 + A1,4 =
50 + 45 = 95

i \ J	1	2	3	4	5	6	7
1	0	10	20	50	**55**		
2		0	10	30	35		
3			0	10	15	30	
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Continuando os Calculos...

i \ J	1	2	3	4	5	6	7
1	0	10	20	50	55		
2		0	10	30	35	50	
3			0	10	15	30	65
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Continuando os Calculos...

i \ J	1	2	3	4	5	6	7
1	0	10	20	50	55	70	
2		0	10	30	35	50	85
3			0	10	15	30	65
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Continuando os Calculos...

i \ J	1	2	3	4	5	6	7
1	0	10	20	50	55	70	105
2		0	10	30	35	50	85
3			0	10	15	30	65
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Resposta calculada se encontrará no $M_{1,7}$

i \ J	1	2	3	4	5	6	7
1	0	10	20	50	55	70	**105**
2		0	10	30	35	50	85
3			0	10	15	30	65
4				0	5	´20	50
5					0	5	20
6						0	10
7							0

Optimal Binary Search Tree

UVa 10304

Qual a Complexidade deste algoritmo?

Optimal Binary Search Tree

UVa 10304

Qual a Complexidade deste algoritmo?

Matriz N x N e para cada no maximo N testes

Portanto solução

Solução N^3

Optimal Binary Search Tree

UVa 10304

Código de cálculo do acumulado:

```
for( i =1; i <=n; i++){
    scanf("%d", &peso[i]);
    acc[i] = acc[i-1] + peso[i];
}
```

Optimal Binary Search Tree

UVa 10304

Código de cálculo da matriz:

```
for( i =1; i <=n; i++){
    for( j =0; j <=n-i; j++){
        ii = i+j; jj = j+1;
        minn = dp[ii-i][jj] + dp[ii][jj+1] + acc[ii] -
acc[jj-1] - peso[jj];
        for( k =1;k < i; k++){
            aux = dp[ii-i+k][jj] + dp[ii][jj+k+1] +
acc[ii] - acc[jj-1] - peso[jj+k];
            if( aux < minn) minn = aux;
        }
        dp[ii][jj] = minn;
    }
}
```

Optimal Binary Search Tree

UVa 10304

Existe outro Jeito de implementar sem utilizar a matriz?

Optimal Binary Search Tree

UVa 10304

Existe também uma otimização que reduz a complexidade para n^2!

Conhecida como otimização de knuth, existe uma explicação dela no Cormen.