FLUENDO CRESCIT

UFS

PROCC

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Atividade:

Árvore Binária de Busca Ótima - OBST

Otimização de buscas considerando probabilidades.

Professor:

Dr. Leonardo Nogueira Matos

Equipe

Francisco Farias Gomes

Silas Lopes Santos Silva Amancio do Vale

Luciano Torres Marques

Ramon Adller de Santana

Introdução - OBST

Introdução - OBST

Uma árvore de busca binária ótima (**O**ptimal **B**inary **S**earch **T**ree – **OBST**) é uma estrutura de dados usada quando há a necessidade de realizar muitas buscas em um conjunto de chaves, mas sabemos com que frequência cada chave será pesquisada.

Introdução - OBST

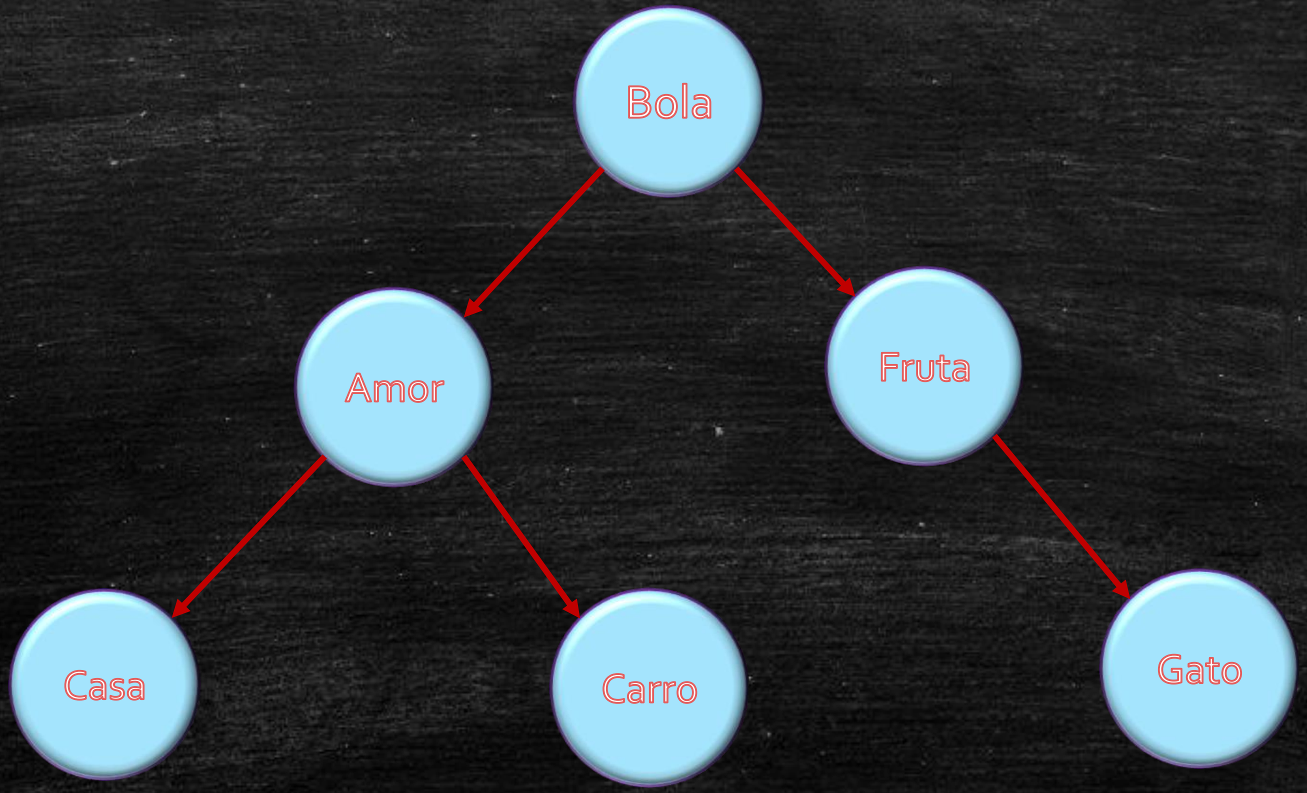
Em uma árvore binárias de busca (**B**inary **S**earch **T**rees) comum, cada busca teria tempo garantido de $O\ (\log n)$. Porque a altura da árvore é proporcional ao logaritmo do número de nós ($h \approx \log n$). Se a BST não for balanceada, no pior caso, a complexidade pode degradar para $O(n)$.

Introdução - OBST

A profundidade máxima de qualquer nó é $O(\log n)$. Isso garante que as operações de: inserção, remoção e busca. Sejam feitas em tempo proporcional a $\log n$.

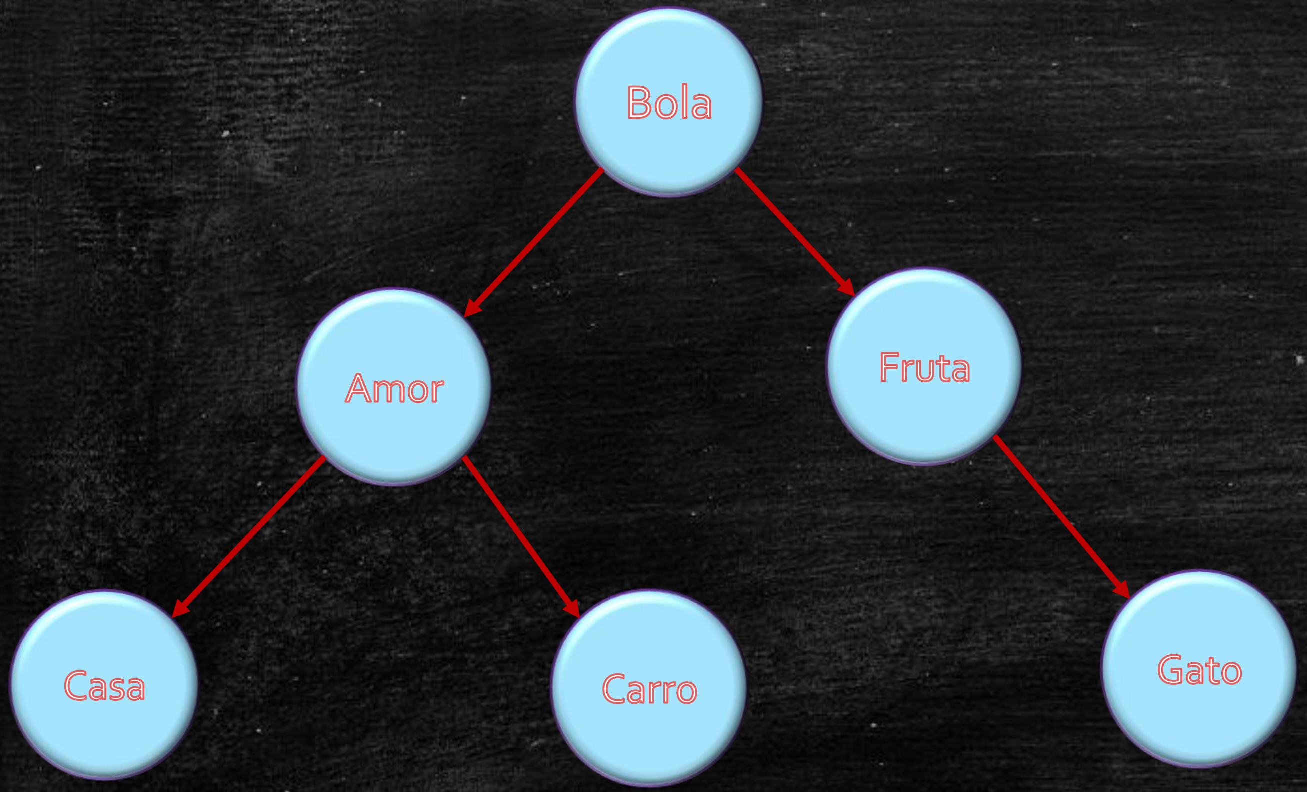
Introdução - OBST

Exemplo da organização de uma árvore binária de busca balanceada:



Introdução - OBST

Análise de profundidade e custo de busca



Palavra	Profundidade (Raiz = 0)	Custo de busca (nós visitados)
Bola	0	1
Amor	1	2
Fruta	1	2
Casa	2	3
Carro	2	3
Gato	2	3

Introdução - OBST

Análise de profundidade e custo de busca

Palavra	Profundidade (Raiz = 0)	Custo de busca (nós visitados)
Bola	0	1
Amor	1	2
Fruta	1	2
Casa	2	3
Carro	2	3
Gato	2	3

Observa-se que quando tem-se n = 6 palavras. O logaritmo é $\log_2(6) \approx 2{,}58$, ou seja, no máximo 3 comparações são necessárias para achar qualquer palavra.

Introdução - OBST

Isso acontece porque, em uma árvore balanceada, cada comparação elimina metade dos possíveis candidatos.

Para n = 8 palavras, no máximo $\log_2(8) = 3$ passos são necessários.

Para n = 1.000 palavras, no máximo $\log_2(1000) \approx 10$ passos.

Para n = 1.000.000 palavras, no máximo $\log_2(1.000.000) \approx 20$ passos.

Introdução - OBST

- Diferentes formas de organizar a mesma BST podem resultar em custos médios de busca muito diferentes.

Introdução - OBST

- Diferentes formas de organizar a mesma BST podem resultar em custos médios de busca muito diferentes.
- Minimizar a altura máxima da árvore, garantindo eficiência no pior caso.

Introdução - OBST

- Diferentes formas de organizar a mesma BST podem resultar em custos médios de busca muito diferentes.
- Minimizar a altura máxima da árvore, garantindo eficiência no pior caso.
- Mas em alguns contextos, não todos os elementos são igualmente prováveis de serem acessados.

Introdução - OBST

- Diferentes formas de organizar a mesma BST podem resultar em custos médios de busca muito diferentes.
- Minimizar a altura máxima da árvore, garantindo eficiência no pior caso.
- Mas em alguns contextos, não todos os elementos são igualmente prováveis de serem acessados.
- Colocar os mais acessados em posições mais fáceis de alcançar.

Motivação - OBST

A Árvore Binária de Busca Ótima minimiza o custo esperado de busca, ponderando as profundidades dos nós pelas suas probabilidades de acesso.

Motivação (Exemplo Prático)

Considere um sistema de busca em um **dicionário eletrônico.**

Motivação (Exemplo Prático)

Considere um sistema de busca em um **dicionário eletrônico** que contém as palavras:

Motivação (Exemplo Prático)

Considere um sistema de busca em um **dicionário eletrônico** que contém as palavras:

Bola, Amor, Casa, Carro, Fruta, Gato

Elas já estão ordenadas lexicograficamente:

Motivação (Exemplo Prático)

Considere um sistema de busca em um **dicionário eletrônico** que contém as palavras:

Bola, Amor, Casa, Carro, Fruta, Gato

Elas já estão ordenadas lexicograficamente:

Amor<Bola<Carro<Casa<Fruta<Gato

Motivação (Exemplo Prático)

Probabilidades de Acesso:

$$p(\text{Amor})=0.05$$

$$p(\text{Bola})=0.25$$

p(Carro)=0.10

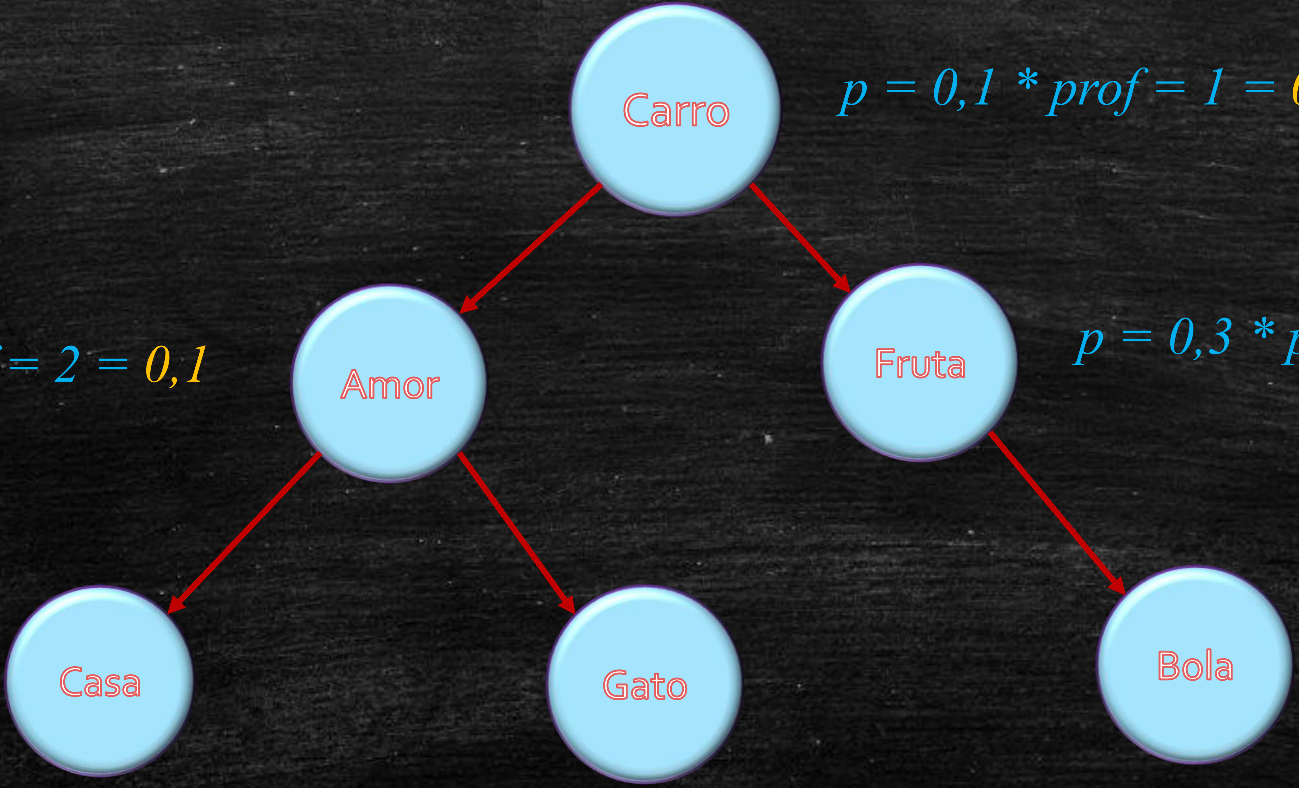
p(Casa)=0.15

p(Fruta)=0.30

p(Gato)=0.15

Motivação (Exemplo Prático)

Árvore balanceada (Carro como raiz):



$p = 0{,}1 * prof = 1 = 0{,}1$

$p = 0{,}05 * prof = 2 = 0{,}1$

$p = 0{,}3 * prof = 2 = 0{,}6$

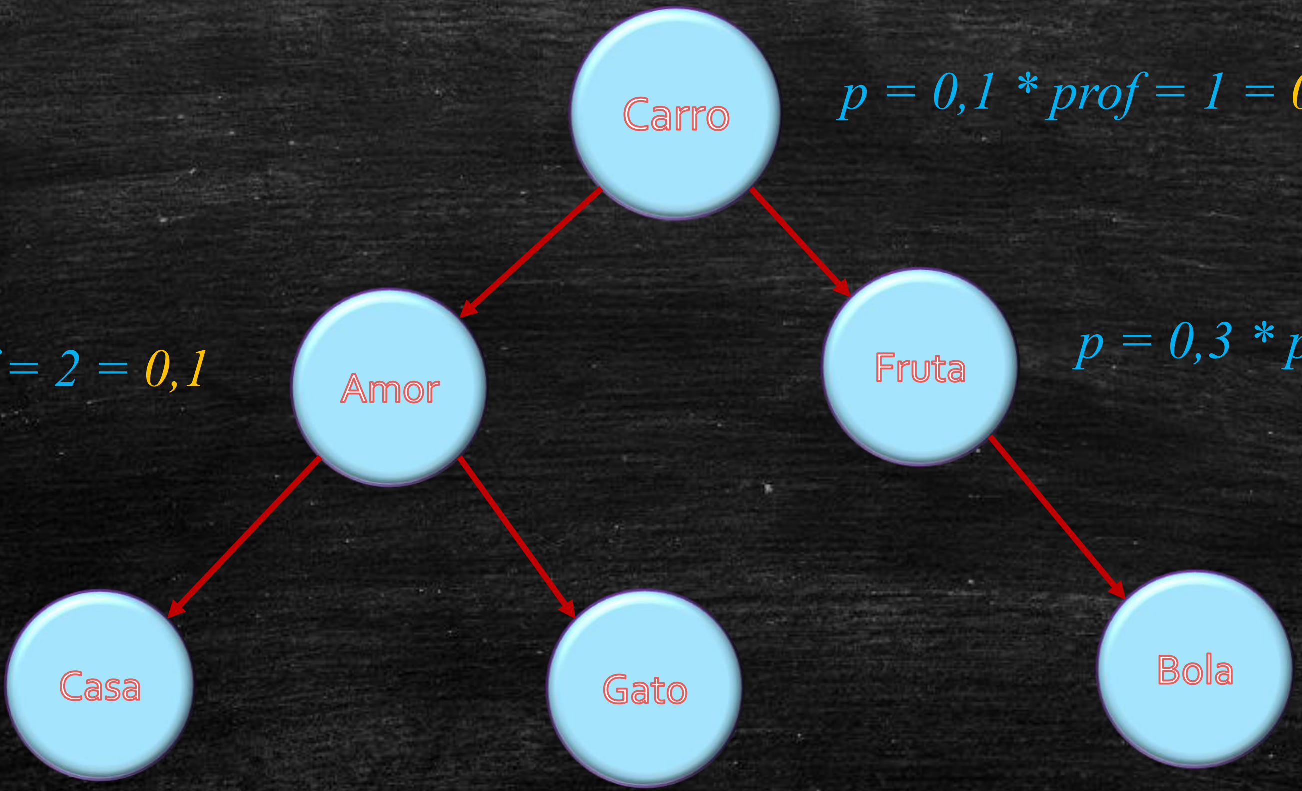
$p = 0{,}15 * prof = 3 = 0{,}45$

$p = 0{,}15 * prof = 3 = 0{,}45$

$p = 0{,}25 * prof = 3 = 0{,}75$

Motivação (Exemplo Prático)

Árvore balanceada (Carro como raiz):



$p = 0{,}1 * prof = 1 = 0{,}1$

$p = 0{,}05 * prof = 2 = 0{,}1$

$p = 0{,}3 * prof = 2 = 0{,}6$

$p = 0{,}15 * prof = 3 = 0{,}45$

$p = 0{,}15 * prof = 3 = 0{,}45$

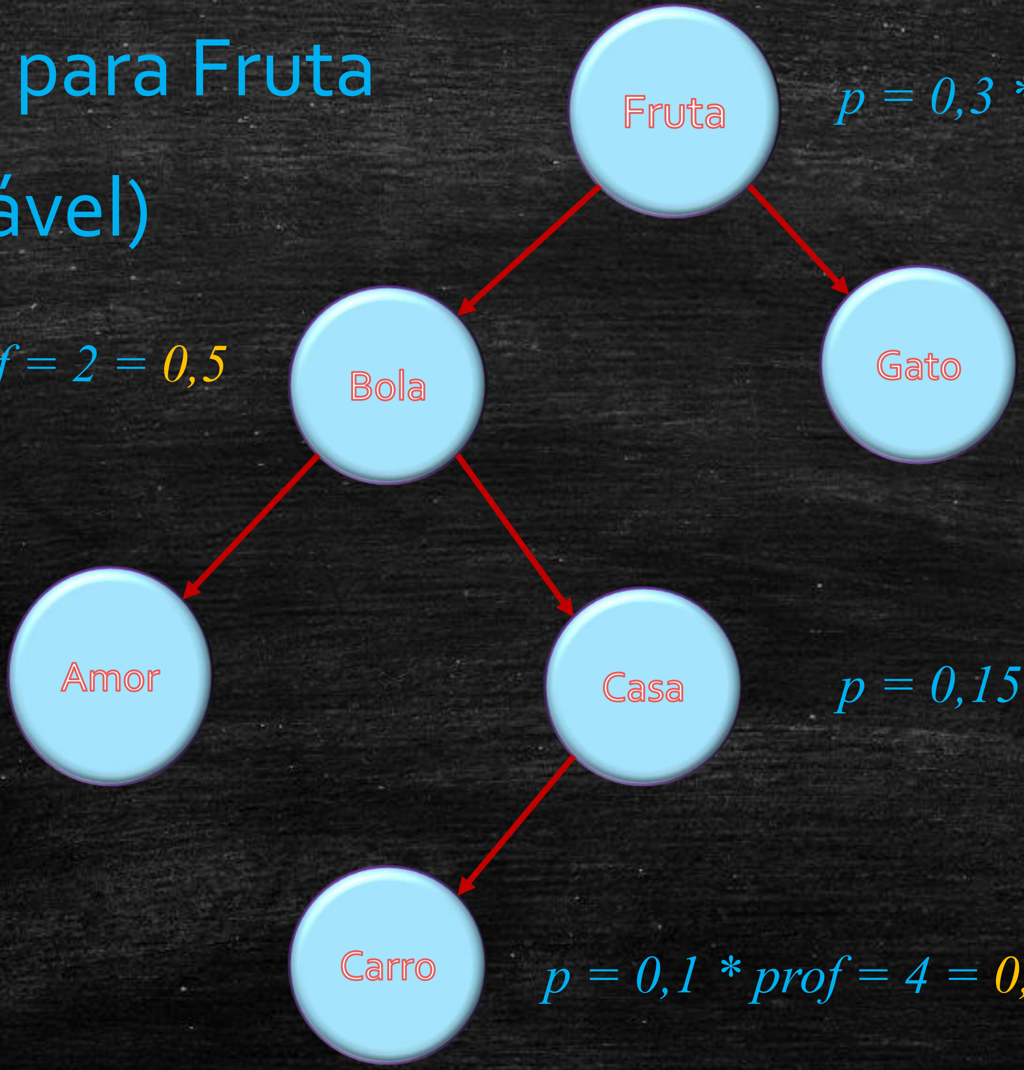
$p = 0{,}25 * prof = 3 = 0{,}75$

Total (balanceada) = 0.10+0.75+0.10+0.45+0.60+0.45=2.450.10+0.75+0.10+0.45+0.60+0.45 = **2.45**.

Motivação (Exemplo Prático)

Árvore enviesada para Fruta

(mais provável)



$p = 0{,}3 * prof = 1 = 0{,}3$

$p = 0{,}25 * prof = 2 = 0{,}5$

$p = 0{,}15 * prof = 2 = 0{,}3$

$p = 0{,}05 * prof = 3 = 0{,}15$

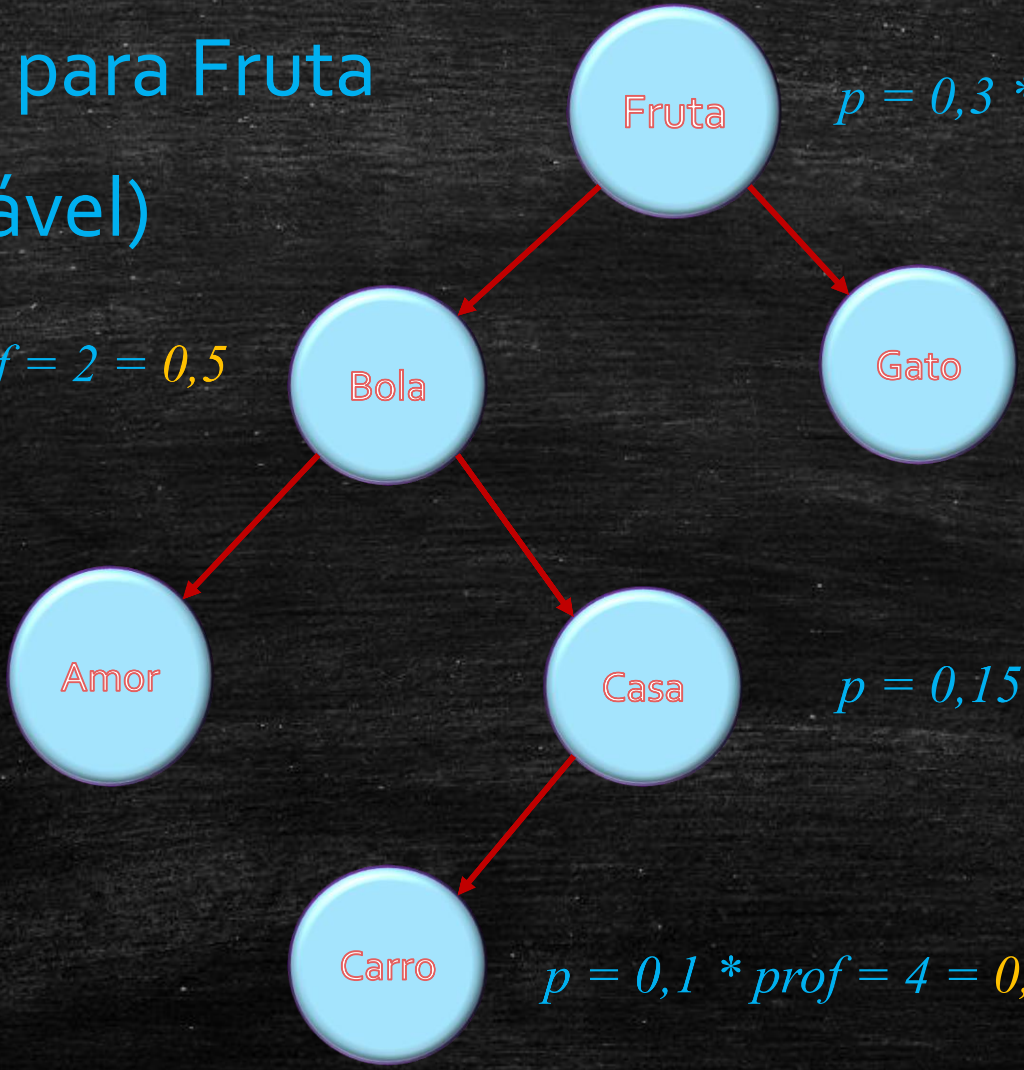
$p = 0{,}15 * prof = 3 = 0{,}45$

$p = 0{,}1 * prof = 4 = 0{,}4$

Motivação (Exemplo Prático)

Árvore enviesada para Fruta

(mais provável)



$p = 0,3 * prof = 1 = 0,3$

$p = 0,25 * prof = 2 = 0,5$

$p = 0,15 * prof = 2 = 0,3$

$p = 0,05 * prof = 3 = 0,15$

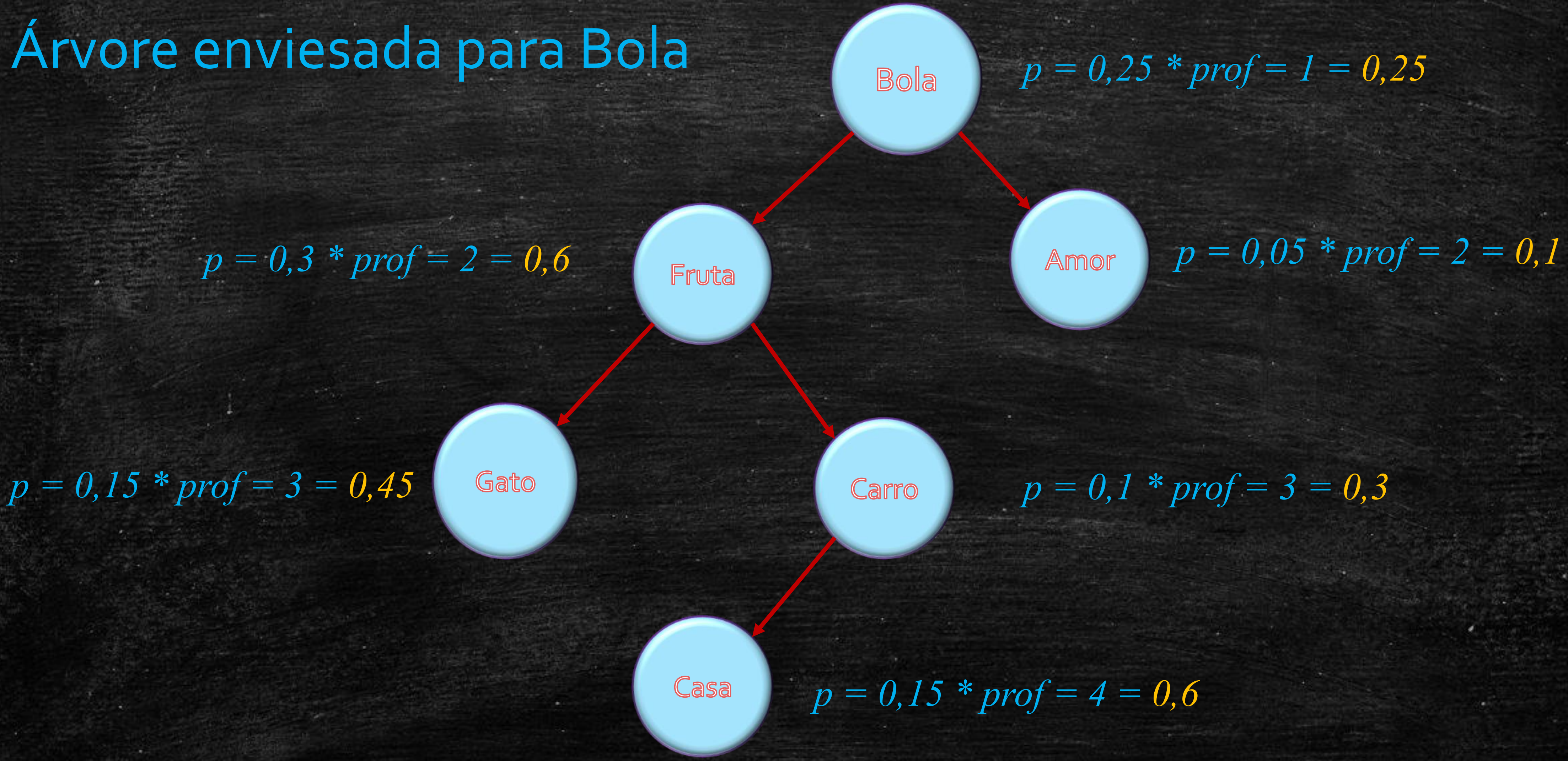
$p = 0,15 * prof = 3 = 0,45$

$p = 0,1 * prof = 4 = 0,4$

Total (Fruta raiz) = 0.30+0.50+0.30+0.15+0.30+0.60=2.150.30+0.50+0.30+0.15+0.30+0.60 = **2.15**.

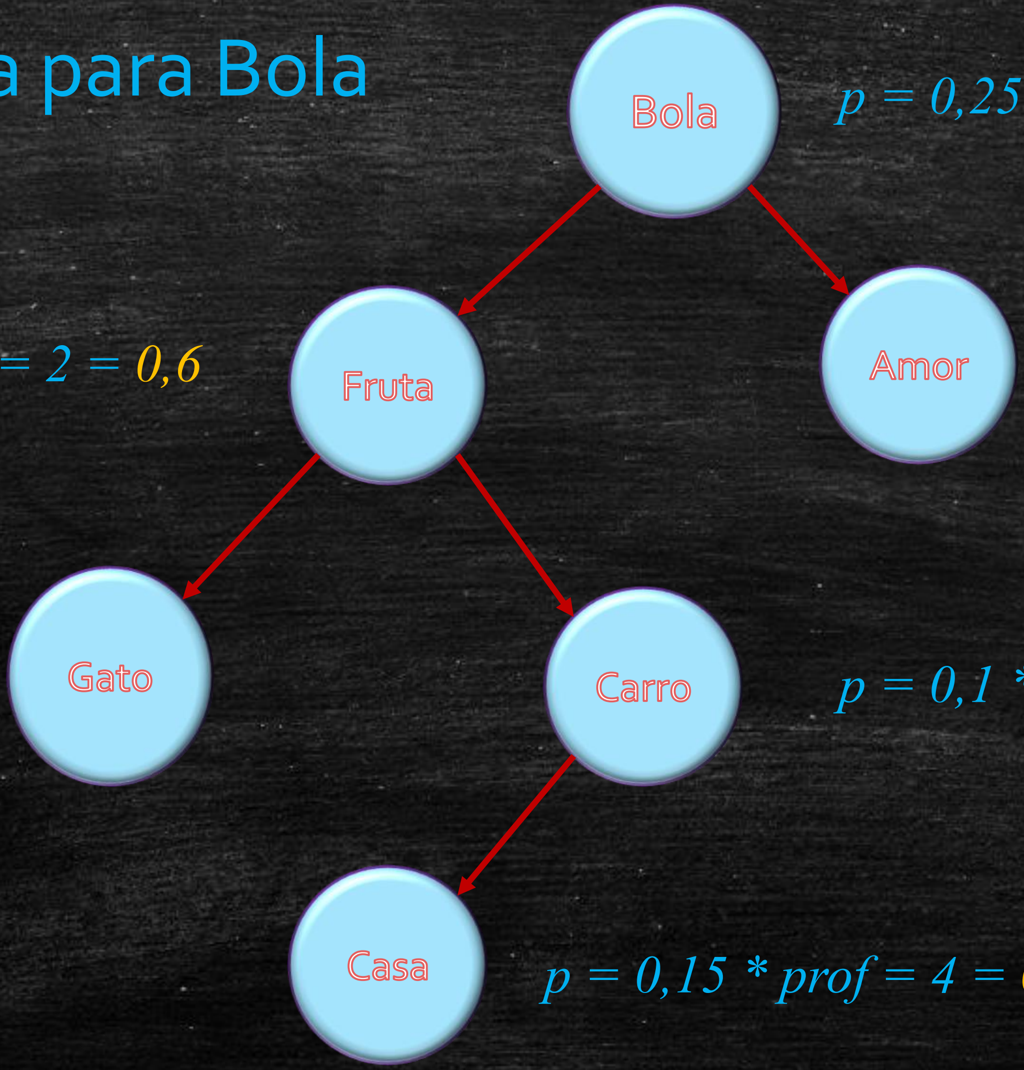
Motivação (Exemplo Prático)

Árvore enviesada para Bola



Motivação (Exemplo Prático)

Árvore enviesada para Bola



$p = 0{,}25 * prof = 1 = 0{,}25$

$p = 0{,}3 * prof = 2 = 0{,}6$

$p = 0{,}05 * prof = 2 = 0{,}1$

$p = 0{,}15 * prof = 3 = 0{,}45$

$p = 0{,}1 * prof = 3 = 0{,}3$

$p = 0{,}15 * prof = 4 = 0{,}6$

Total (Bola raiz) = 0.25+0.10+0.60+0.30+0.60+0.45=2.300.25+0.10+0.60+0.30+0.60+0.45 = **2.30.**

Formalização Matemática do Problema

Definição Formal do Problema

Temos um conjunto de n **chaves ordenadas**:

$$K = \{k_1, k_2, \ldots, kn\}, com\ k_1 < k_2 < \cdots < kn.$$

Além disso, conhecemos probabilidades associadas:

- p_i : probabilidade de sucesso ao buscar a chave k_i , com i=1,...,ni=1,...,n.
- q_i : probabilidade de uma **busca malsucedida** em um intervalo, isto é, entre k_i e k_{i+1}.
- Por convenção, q_0 corresponde a buscas menores que k_1, e q_n a buscas maiores que k_n.
- Assim, os valores $\{p_i\}$ e $\{q_i\}$ representam a **distribuição completa das consultas.**

Custo Esperado de uma Árvore

Seja $d(x)$ a profundidade de um nó x na árvore, definida como o número de nós no caminho da raiz até x.

- Para uma chave real k_i, o custo esperado de acessá-la é $p_i \cdot d(k_i)$.
- Para uma chave dummy d_i (representando falha entre chaves), o custo é $q_i \cdot d(d_i)$

Custo Esperado de uma Árvore

Portanto, o custo esperado total da árvore T é:

$$E[T] = \sum_{i=1}^{n} p_i * d(k_i) + \sum_{i=0}^{n} q_i * d(d_i)$$

Subproblemas e Recorrência

Considere o intervalo de chaves $k_i, \ldots, k_j$.

Se construirmos uma árvore ótima para esse intervalo, o custo esperado mínimo é:

$e[i,j]$ = custo mínimo de uma **OBST** contendo $k_i, \ldots, k_j$.

Custo Esperado de uma Árvore

Portanto, o custo esperado total da árvore T é:

$$w(i,j) = \sum_{k=i}^{j} p_k + \sum_{k=i-1}^{j} qq_k$$

Custo Esperado de uma Árvore

Assim, a recorrência é:

$$e[i,j] = \ min_{i \le r \le j}(\ e[i,r\ -1\ ] + e[r+1,j] + w(i,j))$$

Condição base:

$$e[i,i\ -1\ ] = q_{i-1}, \qquad para\ i = 1, \ldots, n+1$$

Abordagem de Programação Dinâmica

Abordagem de Programação Dinâmica

A programação dinâmica, é usada para explorar o princípio da subestrutura ótima:

- Uma árvore ótima que contém chaves de k_i até k_j deve ter como subárvores **árvores ótimas** para os intervalos à esquerda e à direita da raiz escolhida.

Abordagem de Programação Dinâmica

Na implementação, usamos três tabelas como estrutura auxiliar:

- e[i, j]: custo esperado mínimo de uma OBST com chaves de k_i a k_j.
- w[i, j]: soma das probabilidades (de sucesso e falha) no intervalo $[i, j]$.
- root[i, j]: índice da chave escolhida como raiz ótima para o intervalo.

Abordagem de Programação Dinâmica

Inicialização:

- Para todo i, definimos:

$$e[i, i-1] = q_{i-1}, \qquad w[i, i-1] = q_{i-1},$$

Abordagem de Programação Dinâmica

Preenchimento das Tabelas

O algoritmo percorre intervalos de tamanhos crescentes:

- Para cada comprimento $l = 1,2,\ldots,n$:
  - Considera-se cada intervalo $[i,j]$ com $j = i+l-1$.
  - Calcula-se:

$$e[i,j] = min_{r=i..j}(e[i,r-1] + e[r+1,j] + w[i,j])$$

Abordagem de Programação Dinâmica

- Registra-se a raiz ótima:

$$root[i,j] = arg\ min_{r=i..j}(\ e[i,r\ -1\ ] + e[r+1,j])$$

Também se atualiza:

$$w[i,j] = w[i,j\ -1\ ] + p_j + q_j$$

Abordagem de Programação Dinâmica

Prova (Cut-and-Paste Argument)

A validade do algoritmo de programação dinâmica para OBST depende de mostrar que o problema tem subestrutura ótima.

Abordagem de Programação Dinâmica

Esboço da Prova (Cut-and-Paste Argument)

Suponha, por contradição, que exista uma árvore ótima T para o intervalo $k_i, \ldots, k_j$, com raiz k_r, mas cuja subárvore esquerda **não** seja ótima.

- Seja T_L a subárvore esquerda de T, contendo $k_i, \ldots, k_{r-1}$.
- Seja T_L' outra árvore ótima para esse mesmo intervalo, com custo esperado menor que T_L.

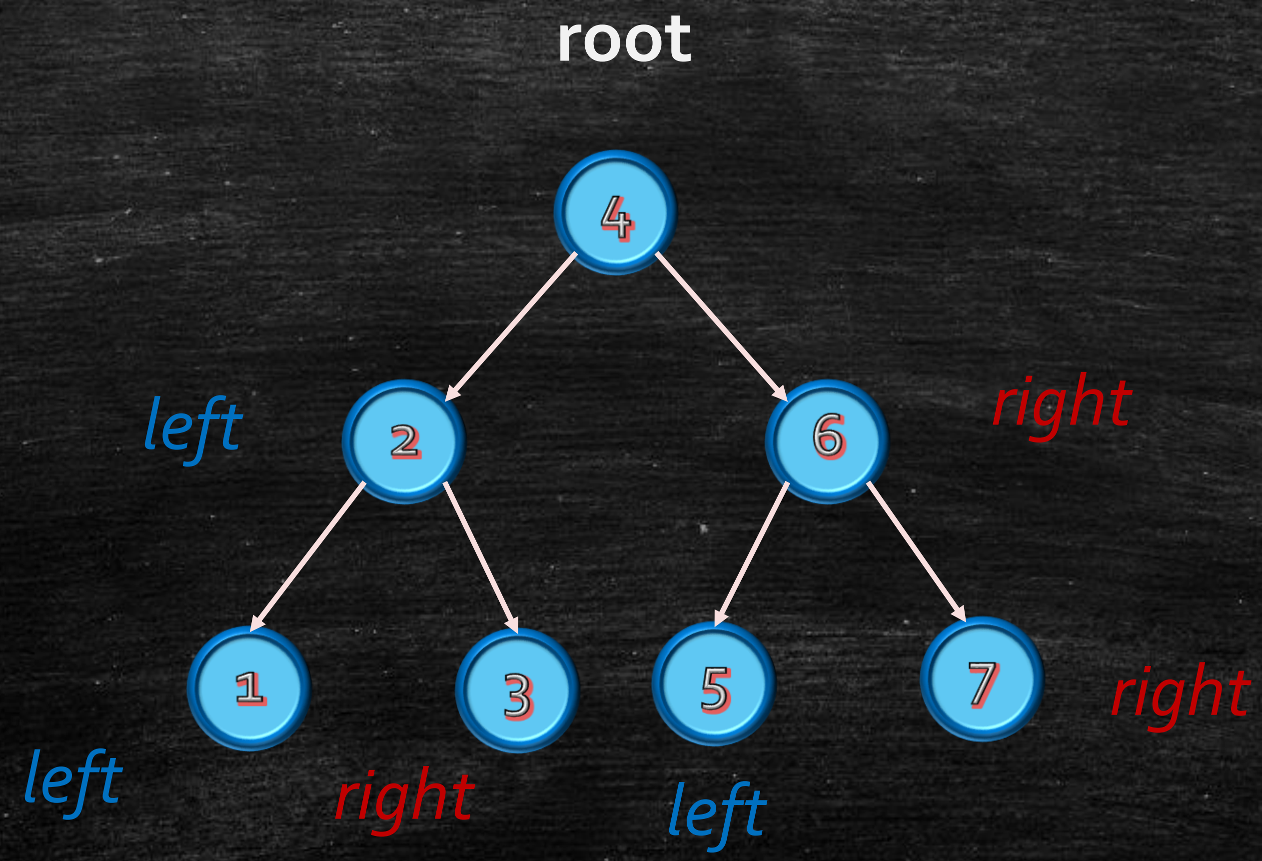
Abordagem de Programação Dinâmica

Se substituirmos T_L por T_L' dentro de T, obtemos uma nova árvore T'.

- Essa substituição não viola as propriedades de árvore binária de busca, pois ambas as árvores têm as mesmas chaves ordenadas no mesmo intervalo.
- O custo esperado de T' será menor que o de T, pois T_L' tem custo menor e todos os demais custos permanecem iguais.

Isso contradiz a suposição de que T era ótimo.

Obrigado



Árvore Binária de Busca Ótima - OBST

Otimização de buscas considerando probabilidades.