Inteligência Artificial
Aula 3 - vídeo 3 - Listas

26 de agosto de 2020

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- $append([a, b], [c, d], [a, b, c, d])$ é verdadeiro.
- $append([a, b], [c, d], [a, b, a, c, d])$ é falso.

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- Casos mais simples de concatenação de listas:

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- Casos mais simples de concatenação de listas:
  - $L_1 = [\,]$ e $L_2 = [\,]$: $L_3 = [\,]$

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- Casos mais simples de concatenação de listas:
  - $L_1 = [\,]$ e $L_2 = [\,]$: $L_3 = [\,]$
  - $L_1 = [\,]$ e $L_2 \neq [\,]$: $L_3 = L_2$

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- Casos mais simples de concatenação de listas:
  - $L_1 = [\,]$ e $L_2 = [\,]$: $L_3 = [\,]$
  - $L_1 = [\,]$ e $L_2 \neq [\,]$: $L_3 = L_2$
- Observe que podemos juntar os casos:

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- Casos mais simples de concatenação de listas:
  - $L_1 = [\,]$ e $L_2 = [\,]$: $L_3 = [\,]$
  - $L_1 = [\,]$ e $L_2 \neq [\,]$: $L_3 = L_2$
- Observe que podemos juntar os casos:
  - $L_1 = [\,]$ e L_2: $L_3 = L_2$

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- Casos mais simples de concatenação de listas:
  - $L_1 = [\,]$ e $L_2 = [\,]$: $L_3 = [\,]$
  - $L_1 = [\,]$ e $L_2 \neq [\,]$: $L_3 = L_2$
- Observe que podemos juntar os casos:
  - $L_1 = [\,]$ e L_2: $L_3 = L_2$
- Isso nos permite definir o caso base:

$$append([\,], L, L).$$

Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- Isso nos permite definir o caso base:

$$append([\,], L, L).$$

- **Caso mais geral:** procurar uma forma de reduzir ao caso base.

Recursão em Prolog

Recursão em Prolog

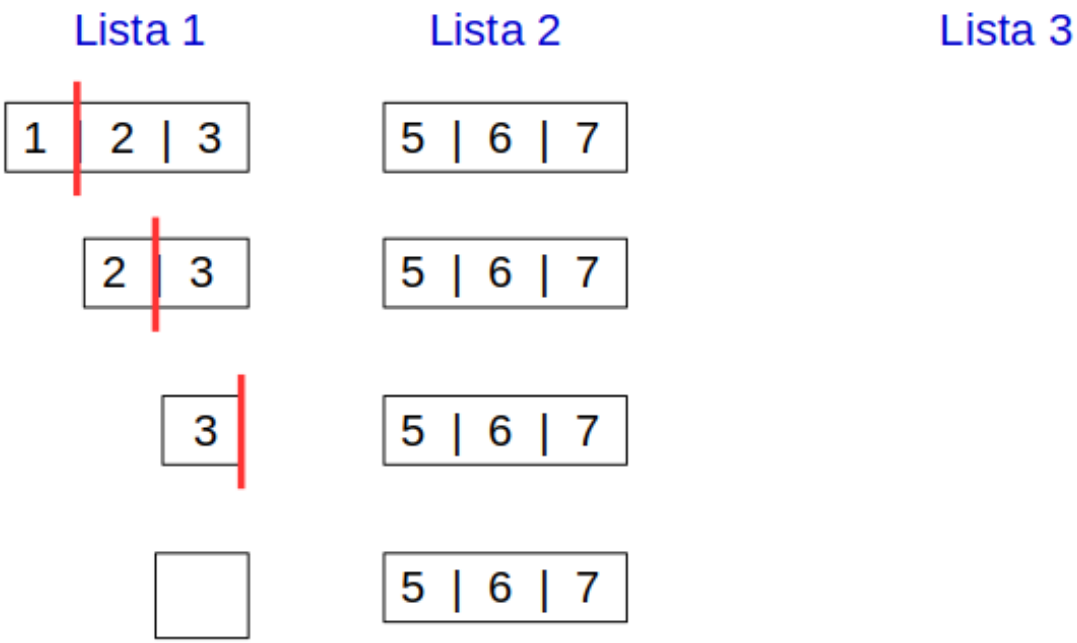
Recursão em Prolog

Recursão em Prolog

Recursão em Prolog

Recursão em Prolog

Recursão em Prolog

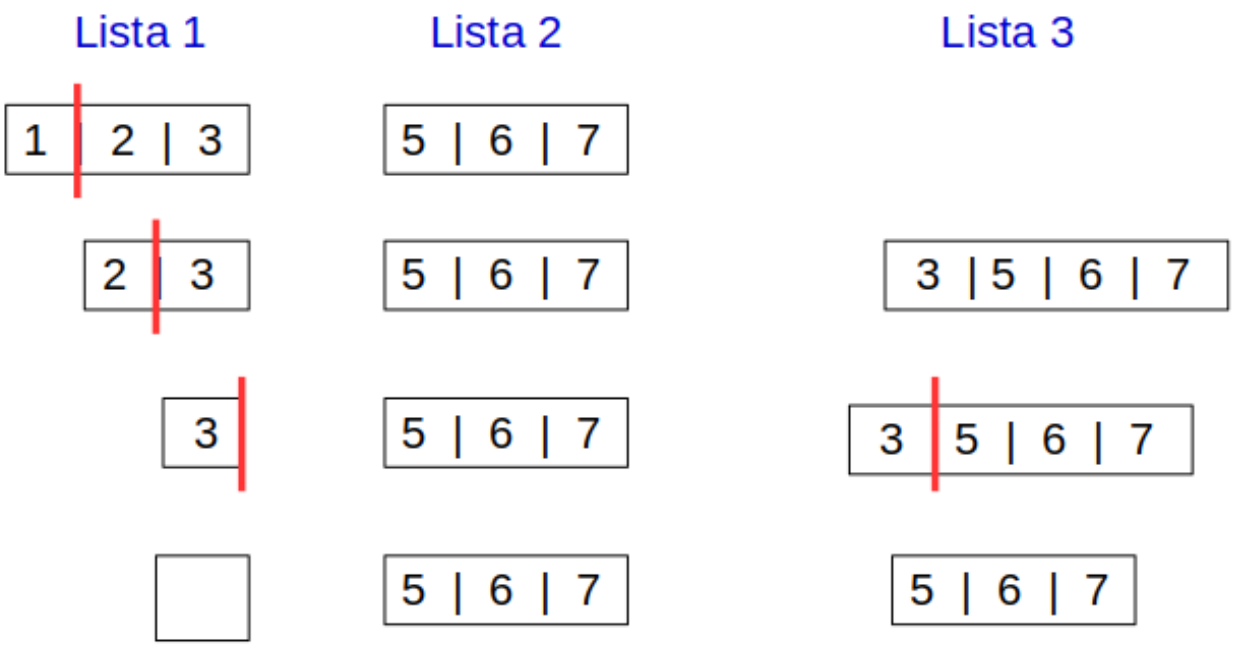


Recursão em Prolog

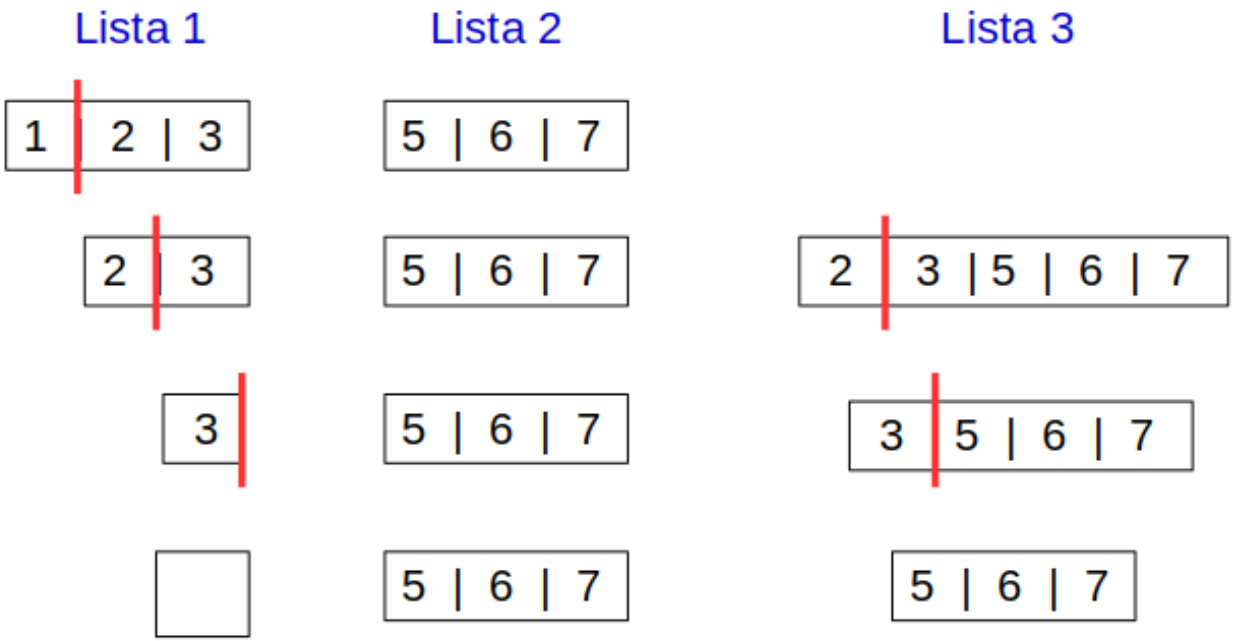
Recursão em Prolog

Recursão em Prolog

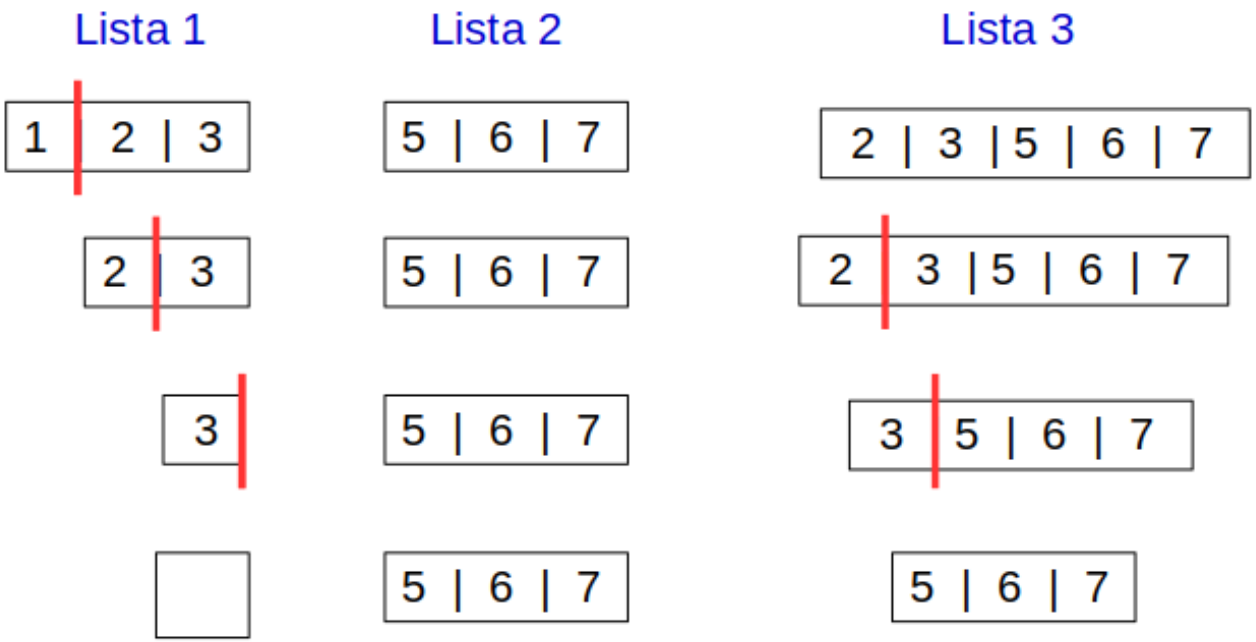
Recursão em Prolog



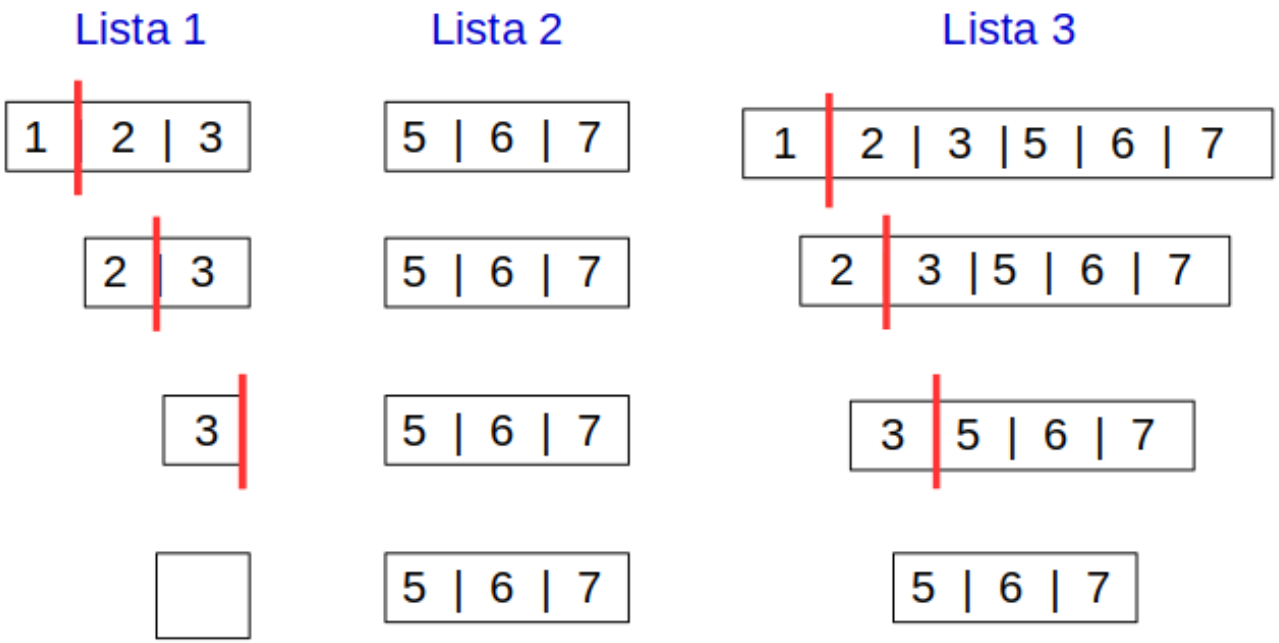
Recursão em Prolog



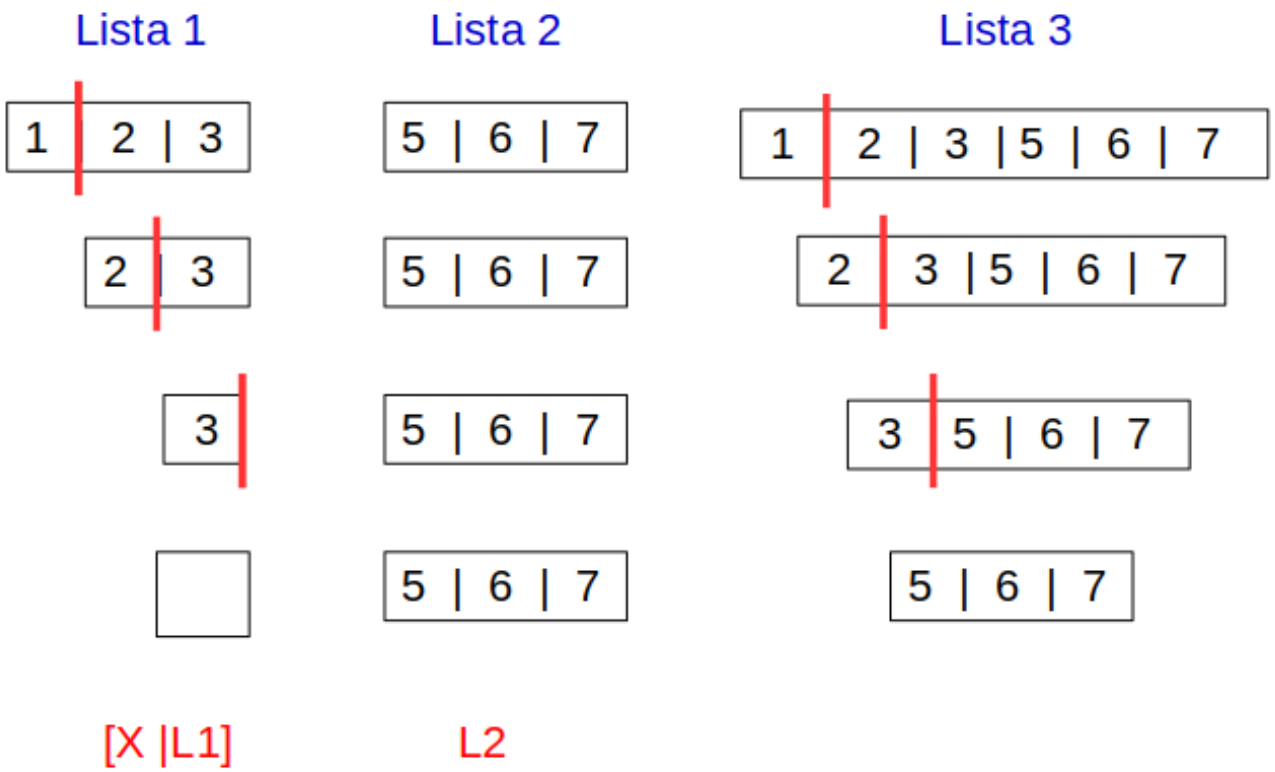
Recursão em Prolog



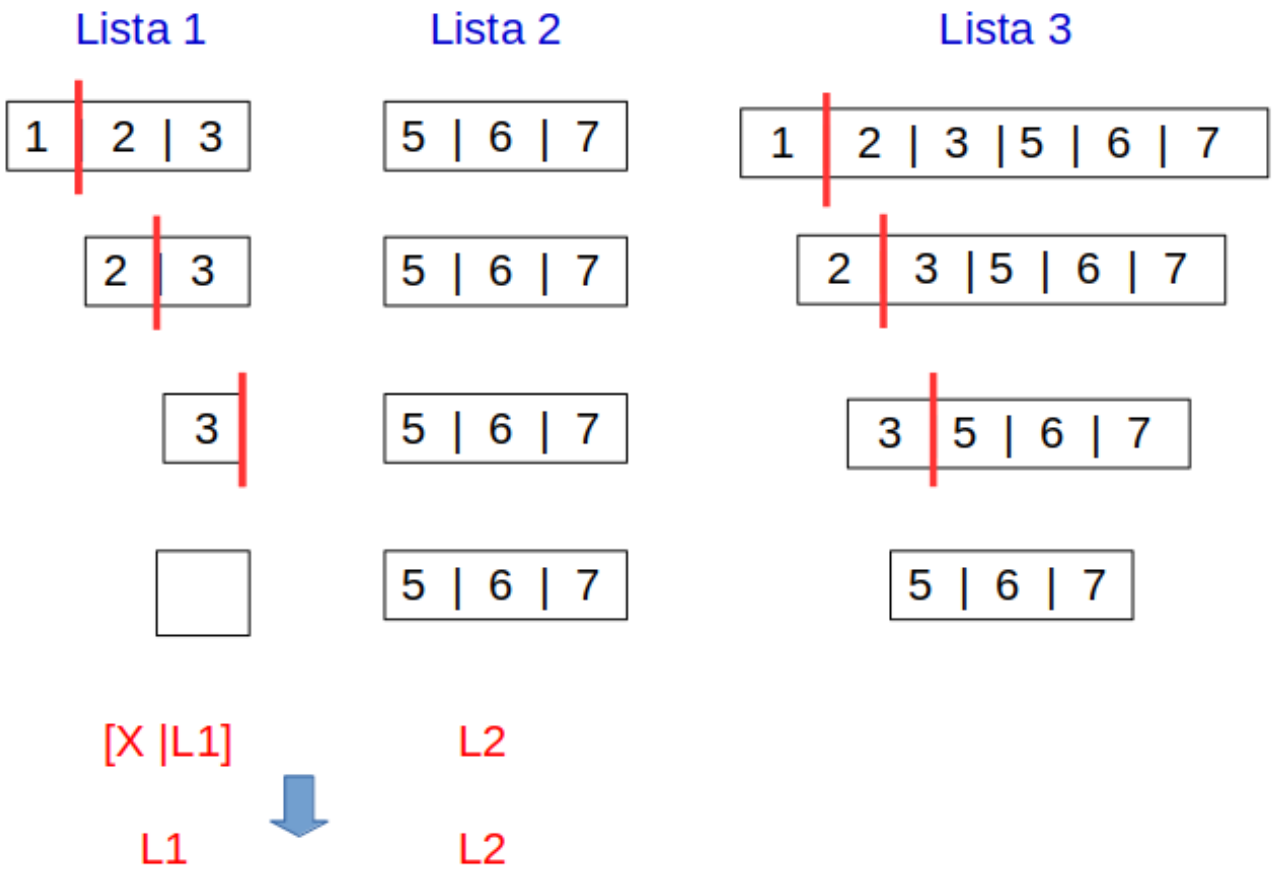
Recursão em Prolog



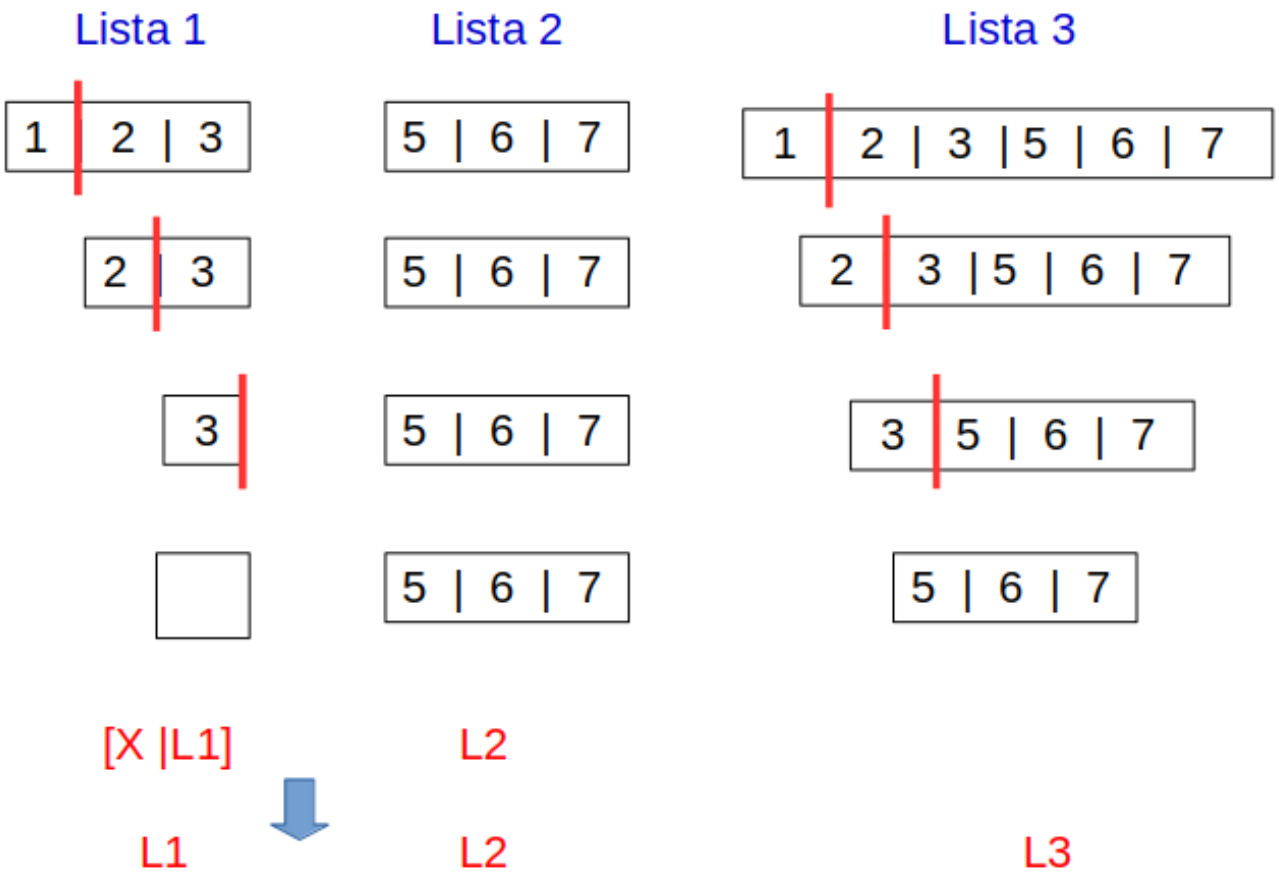
Recursão em Prolog



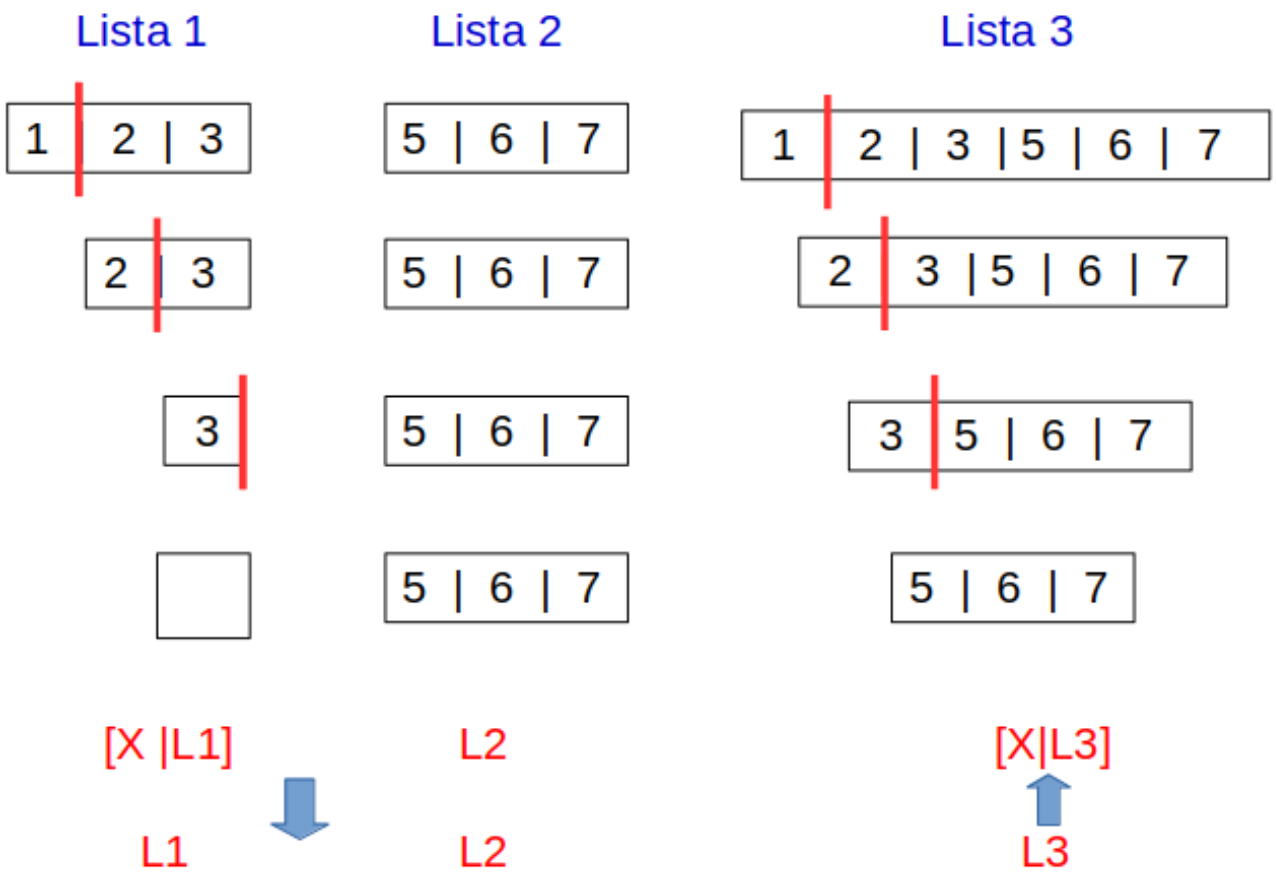
Recursão em Prolog



Recursão em Prolog



Recursão em Prolog



Recursão em Prolog

Faça um programa que dadas duas listas L_1 e L_2, gera uma lista L_3 que é a concatenação das listas L_1 e L_2.

- $append([a, b], [c, d], [a, b, c, d])$ é verdadeiro.
- $append([a, b], [c, d], [a, b, a, c, d])$ é falso.

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

```
?- append([a,b],[1,2],L).
```

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

```
?- append([a,b],[1,2],L).
L = [a, b, 1, 2].
```

```
?- append([a,b],Y,[a,b,1,2]).
```

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

```
?- append([a,b],[1,2],L).
L = [a, b, 1, 2].
```

```
?- append([a,b],Y,[a,b,1,2]).
Y = [1, 2].
```

```
?- append(X,[1,2],[a,b,1,2]).
```

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).

?- append([a,b],[1,2],L).
L = [a, b, 1, 2].

?- append([a,b],Y,[a,b,1,2]).
Y = [1, 2].

?- append(X,[1,2],[a,b,1,2]).
X = [a, b] ;
false.
```

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

?- append(X,Y,[1,2,3]).

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

```
?- append(X,Y,[1,2,3]).
X = [ ],
Y = [1, 2, 3] ;
X = [1],
Y = [2, 3] ;
X = [1, 2],
Y = [3] ;
X = [1, 2, 3],
Y = [ ] ;
false.
```

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

Como podemos programar a relação **member** usando a relação **append** ?

Recursão em Prolog

```
append([ ],L,L).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).
```

Como podemos programar a relação **member** usando a relação **append** ?

```
member(X,L) :- append(_,[X|_],L).
```

Inteligência Artificial
Aula 3 - vídeo 3 - Listas

26 de agosto de 2020