Tópicos Avançados em Algoritmos

Hamilton José Brumatto

Bacharelado em Ciências da Computação - UESC

12 de março de 2019

Árvore de Busca *MinMax* e Poda *Alfa-Beta*

Jogos de Adversários

- Existem vários tipos de jogos, com informação perfeita ou imperfeita, determinístico ou de sorte.

	Determinístico	*Sort*
Informação perfeita	Xadrez, Damas, Go, Jogo da Velha, ...	Gamão, Ludo, Banco Imobiliário
Informação imperfeita		Bridge, Poker, War, ...

- Vamos trabalhar com jogos Determinísticos de Informação perfeita com 2 jogadores

Técnicas de IA

- Jogos são interessantes para IA.
- Representa um ambiente acessível.
- Trabalha com abstrações (representação simplificada de problemas reais).
- Utiliza técnicas de busca.

Árvore de Busca

- É a representação de jogos para dois jogadores utilizando árvores, onde:
  - **Nó raiz**: estado antes de qualquer movimento do jogo;
  - **Nós da árvore**: possíveis estados do jogo;
  - **Arestas**: movimentos dos jogadores;
  - **Folhas**: estados finais do jogo, situações de vitória ou derrota.
  - **Níveis**: vez de cada jogador; nível MAX, você joga, nível MIN o adversário joga.
- Pode-se criar a árvore com o nó raiz no nível MAX (você é o primeiro a jogar) ou no nível MIN (o adversário joga primeiro).

Árvore de Busca: Problema desafiador

- Tamanho do espaço de busca: limitação de tempo
  - Jogo da velha 9! nós = 362.880 nós
  - Jogo de Xadrez, se considerarmos uma média de 40 movimentos, são 10^{120} estados (maior que a quantidade de átomos no universo). Se computar-se 200 milhões de posições por segundo serão 10^{100} anos para resolver. A idade do universo atual é 10^{10} anos.
  - Técnicas de poda na árvore de busca ou de antecipação podem ser usadas.

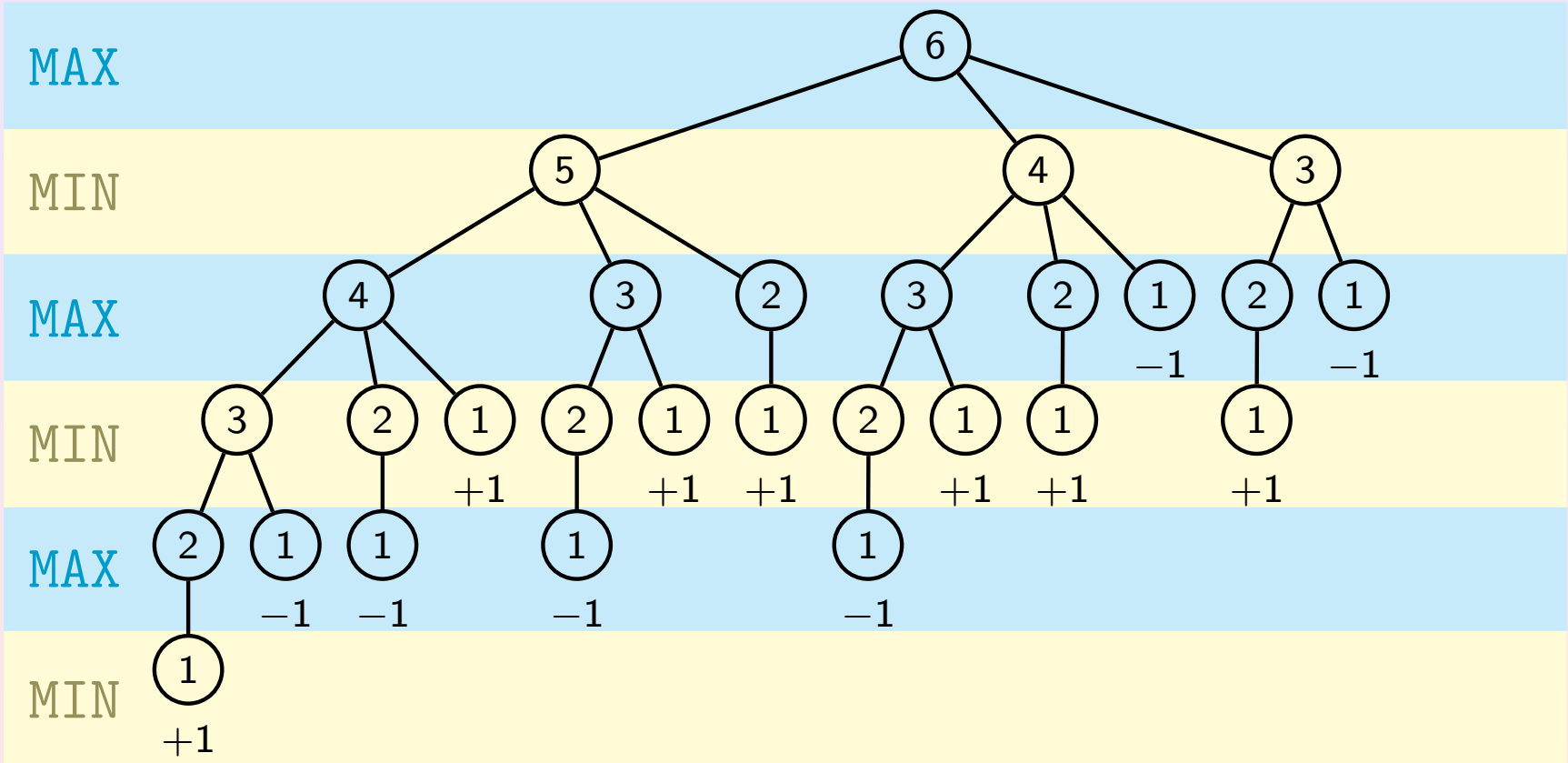
Árvore de Busca: Construção

- Constrói-se a árvore, ramificando cada jogada, e indicando nos níveis, se é MAX, ou MIN. Dependendo de quem é a vez de jogar.
- Nas folhas são colocados os resultados: $+1$ se for vitória, -1 se for derrota, ou 0 se for empate.
- A propagação dos valores vai da folha para o raiz.
- Um nó MAX escolhe o valor mais positivo de seus filhos diretos. Um nós MIN escolhe o valor negativo de seus filhos diretos.
- Ao final, se o raiz tiver um valor positivo, então há uma estratégia vencedora, basta sequir os valores positivos do raiz à folha

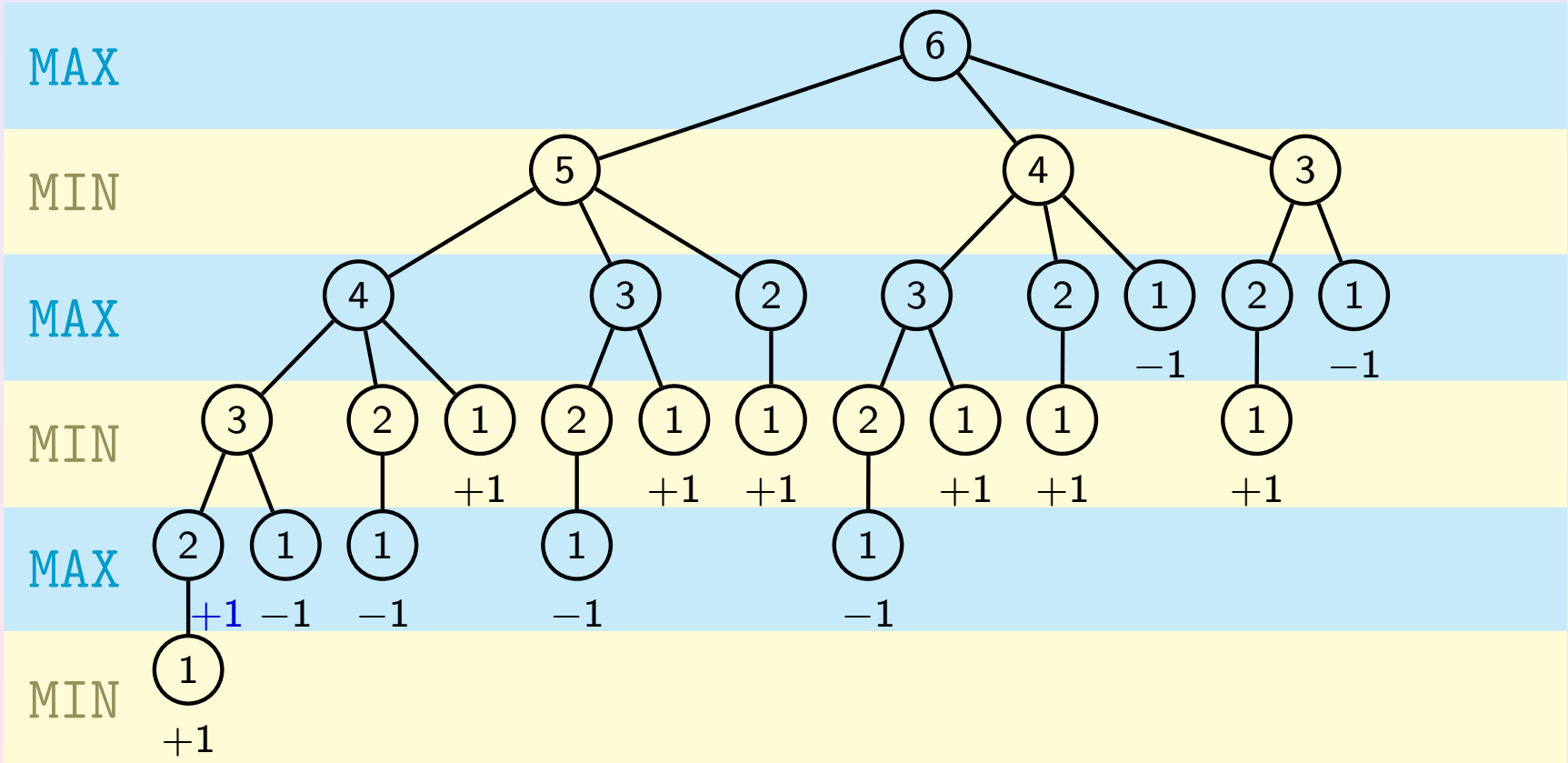
Árvore de Busca: Jogo Simples

- Vamos considerar o jogo das moedas:
  - Começa-se o jogo com 6 moedas.
  - Cada jogador pode retirar 1, 2 ou 3 moedas.
  - Quem retirar a última moeda perde.
- Vamos construir a árvore, indicando a vez de jogar, com MAX para você e MIN para o adversário.
- Se sobre uma moeda no nível MAX, você perde. Se sobra uma moeda no nível MIN, você ganha.
- A folha ganha valor $+1$ para vitória e -1 para derrota.

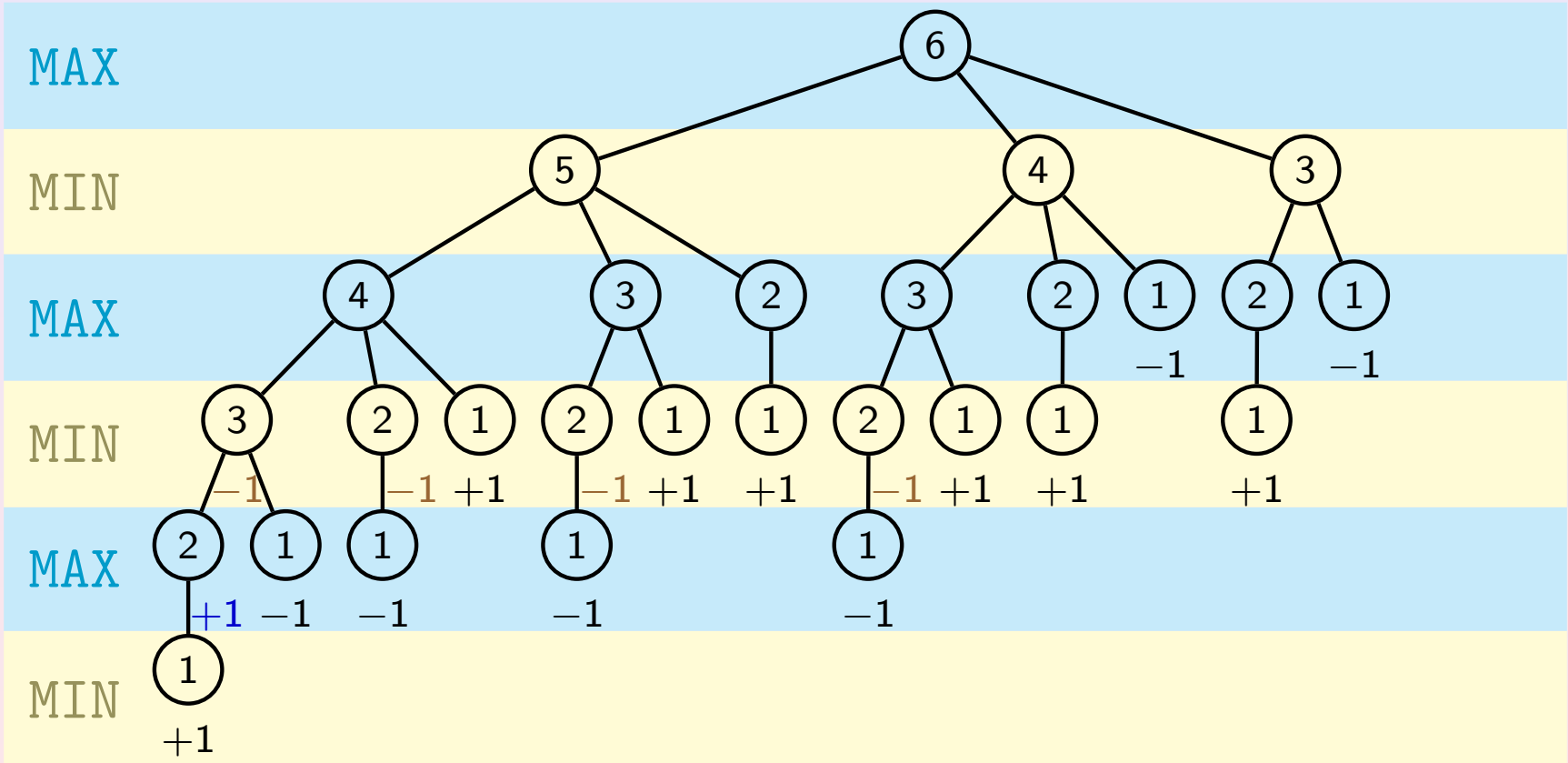
Ávore de Busca: Jogo das moedas



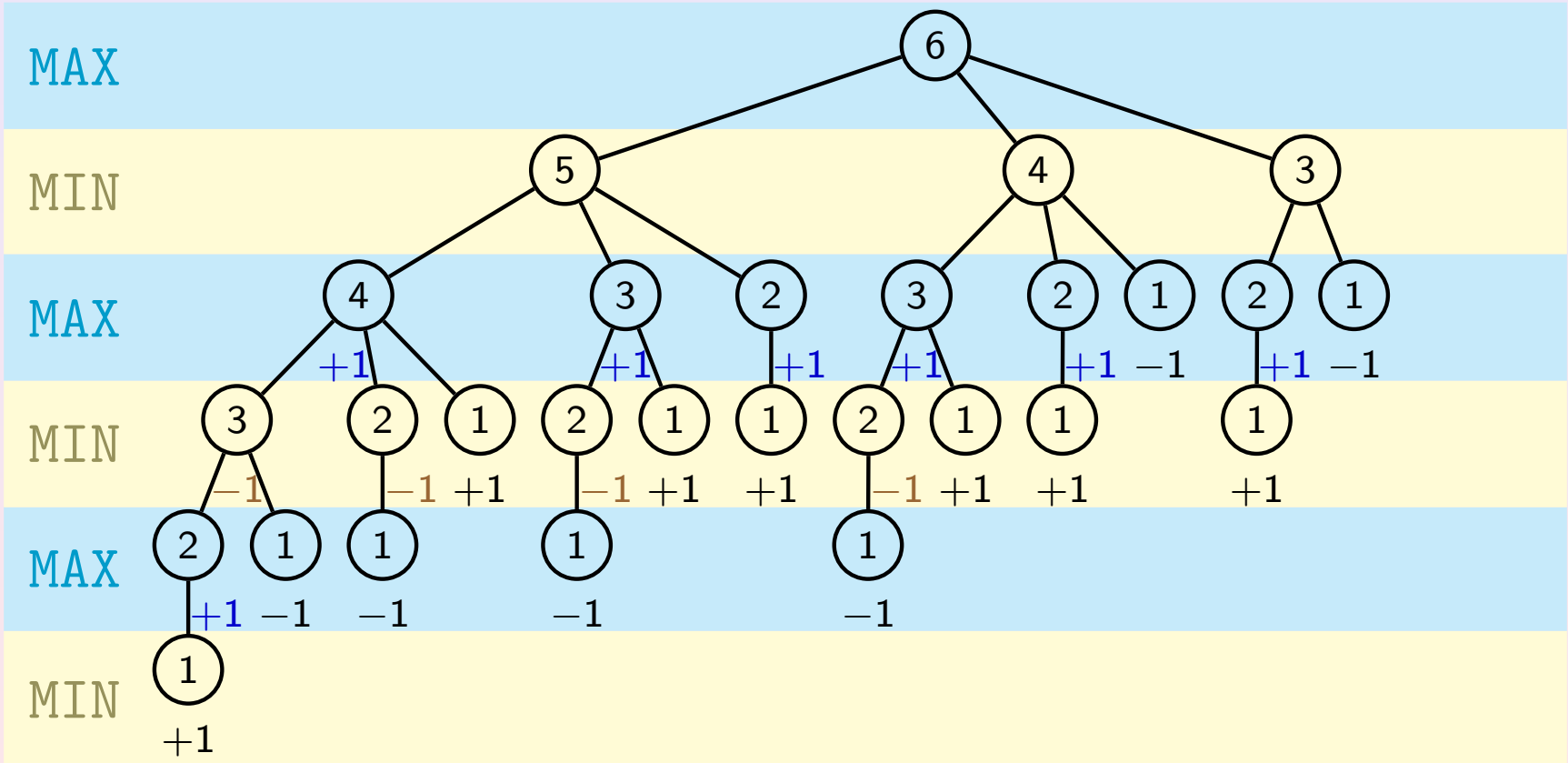
Ávore de Busca: Jogo das moedas - Propagando no segundo nível



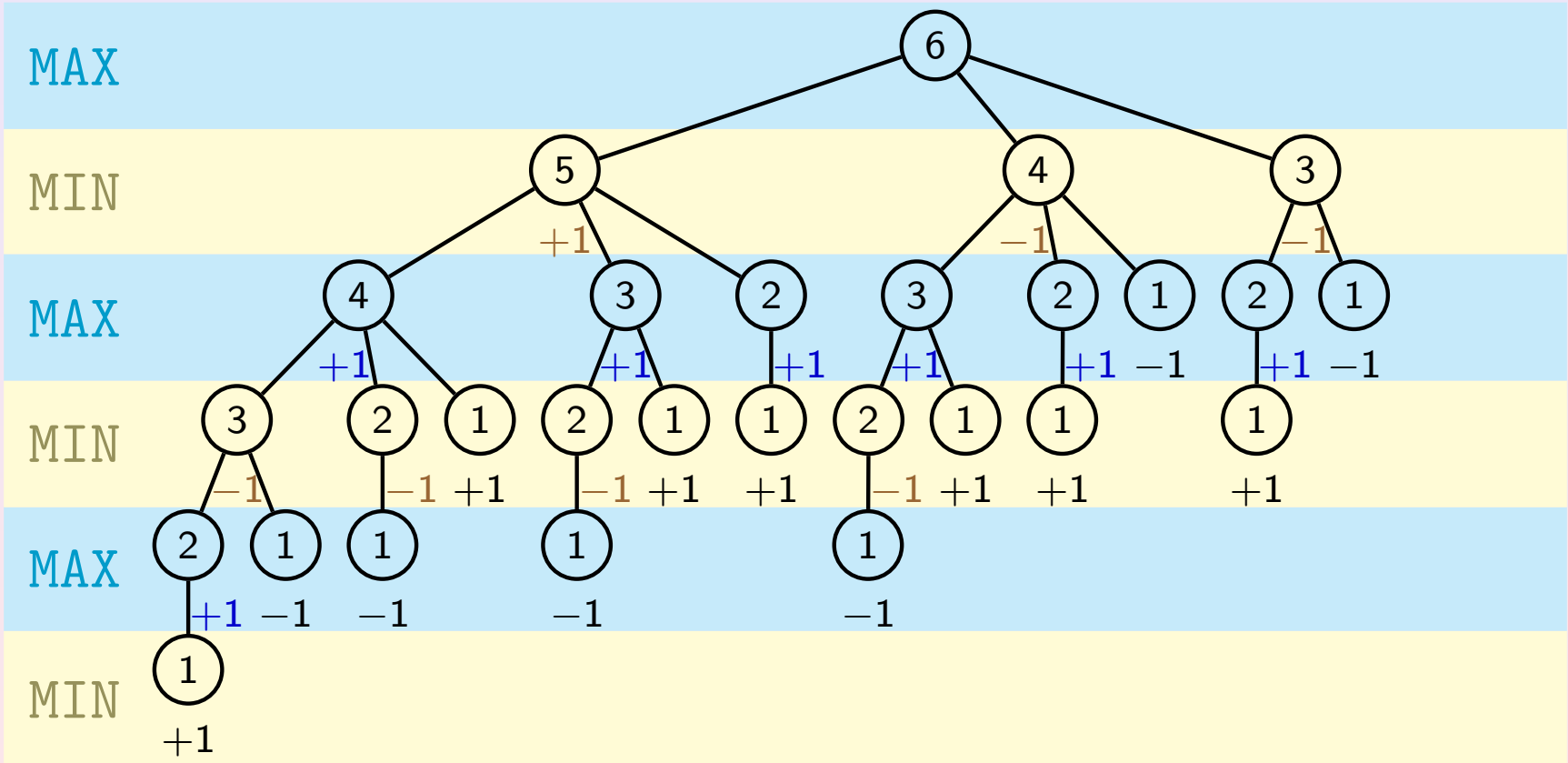
Ávore de Busca: Jogo das moedas - Propagando no terceiro nível



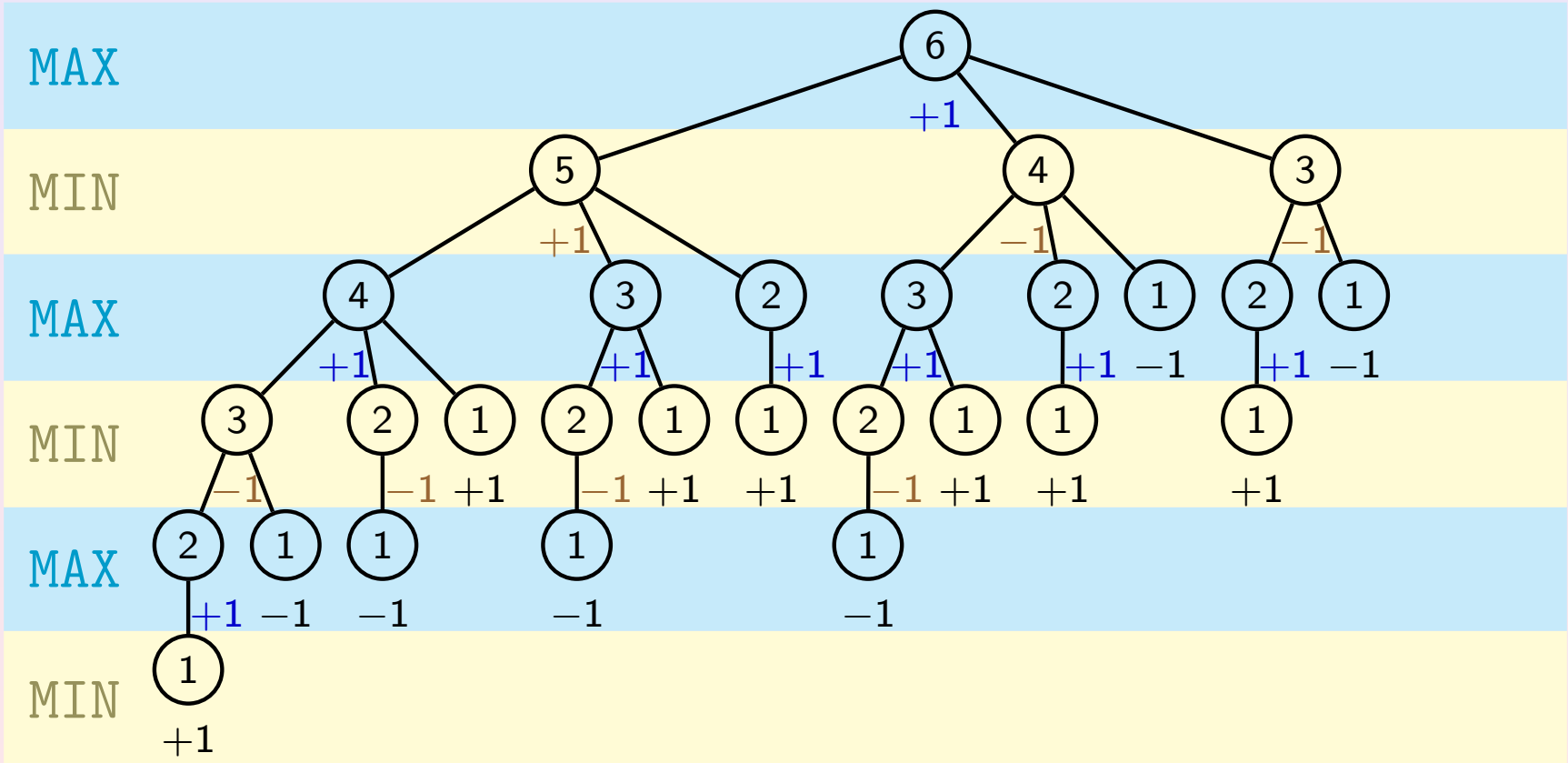
Ávore de Busca: Jogo das moedas - Propagando no quarto nível



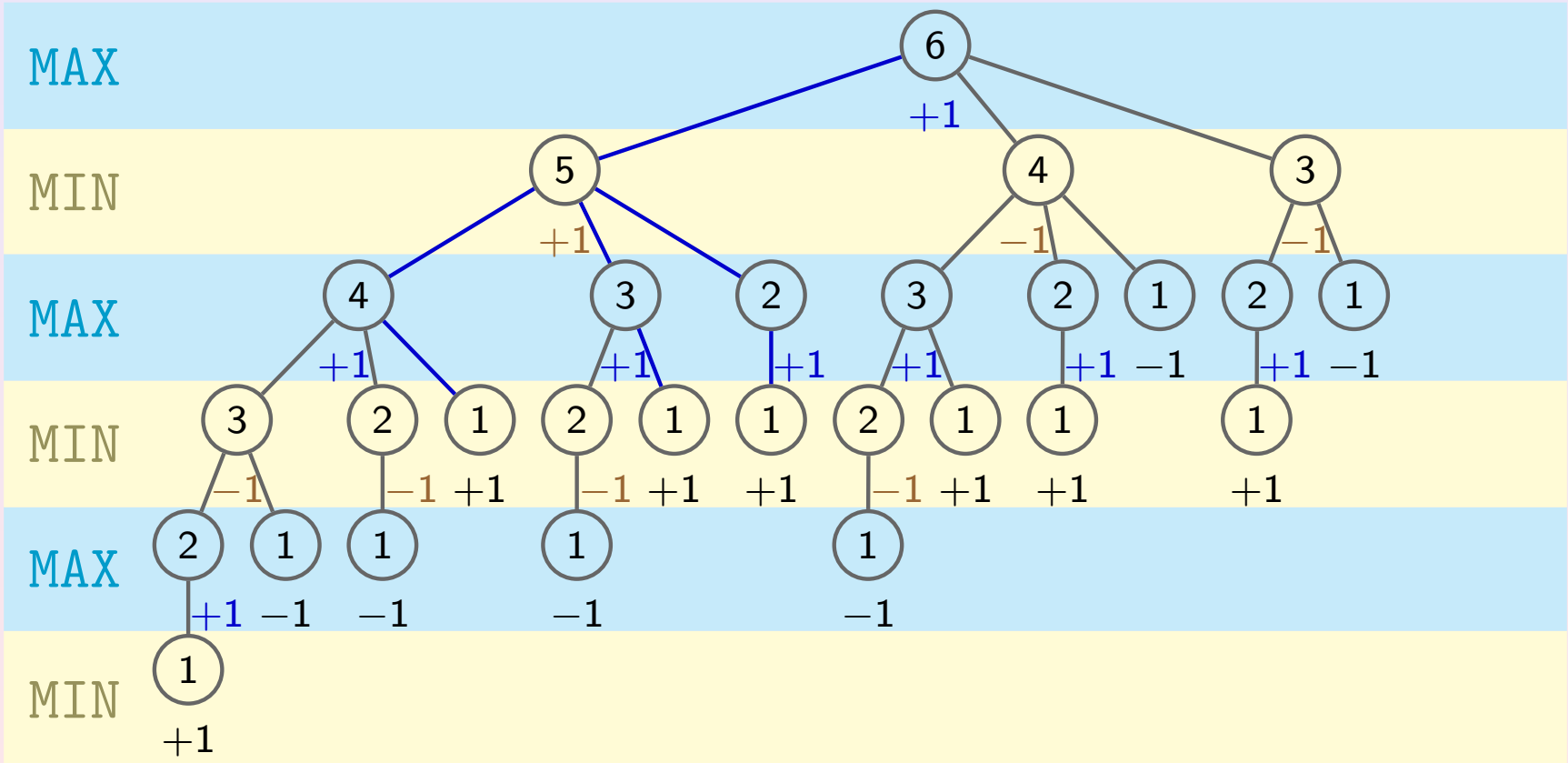
Ávore de Busca: Jogo das moedas - Propagando no quinto nível



Ávore de Busca: Jogo das moedas - Propagando no sexto nível



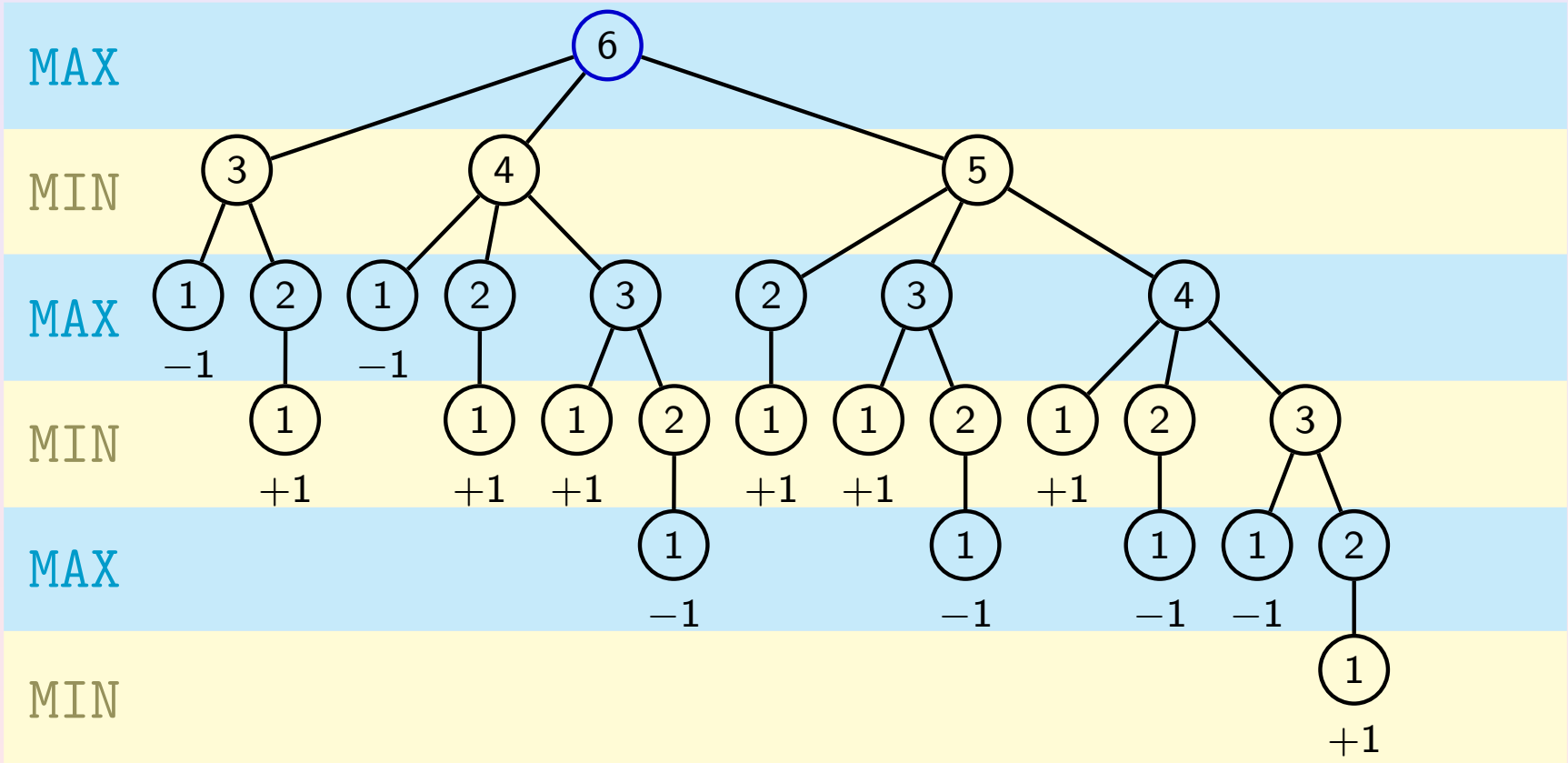
Ávore de Busca: Jogo das moedas - Estratégia vencedora



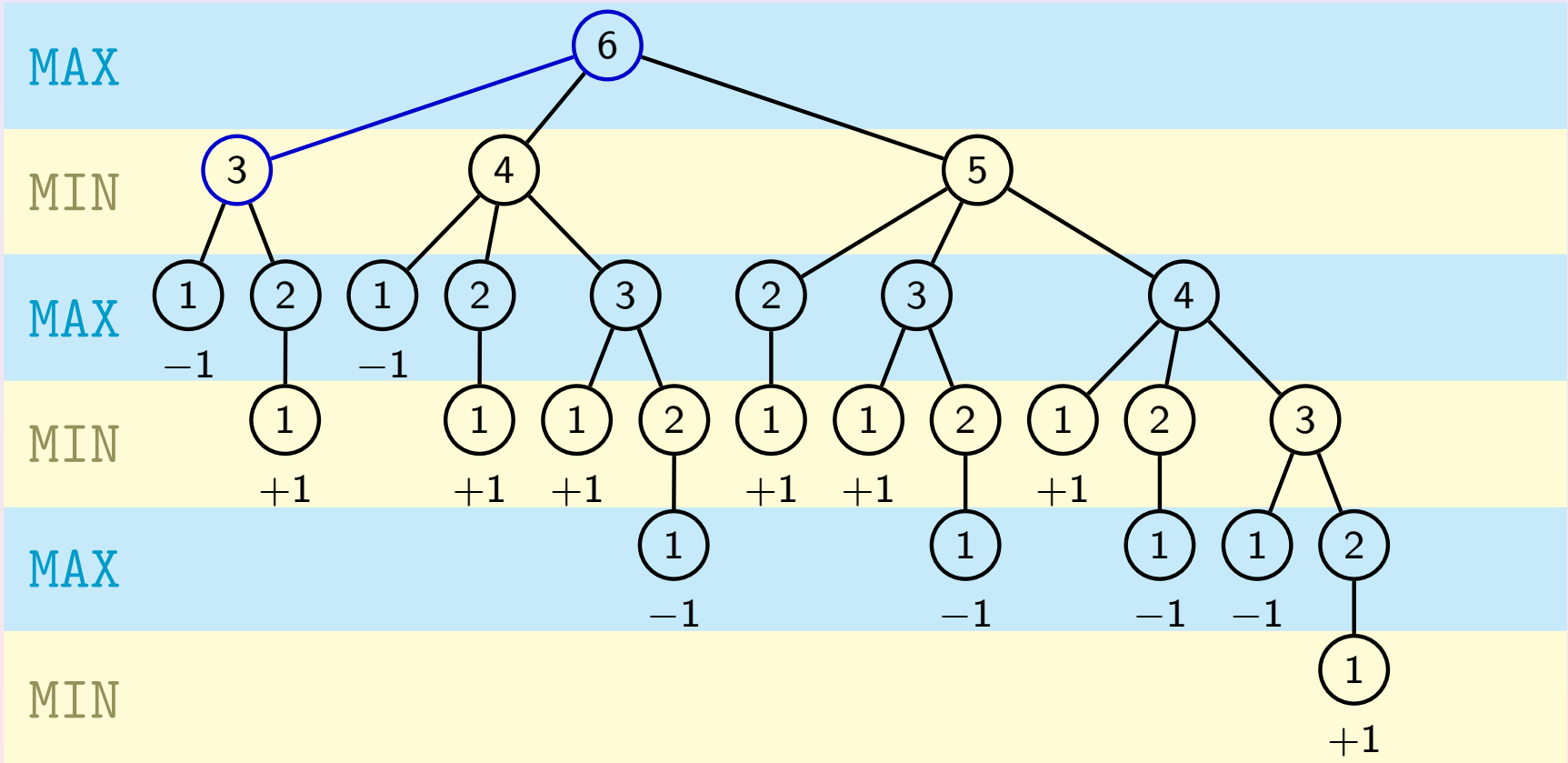
Preenchendo a árvore de busca

- Normalmente para preencher a árvore de busca, utiliza-se uma busca em profundidade.
- O custo de preencher a árvore toda é muito alto.
- Se considerarmos que a árvore tenha altura d e cada jogador possui b opções, então o algoritmo irá percorrer $O(b^d)$ nós.
- No nível MIN, sempre troco o valor atual do nó se a busca em um filho retornar valor menor.
- No nível MAX, sempre troco o valor atual do nó se a busca em um filho retornar valor maior.

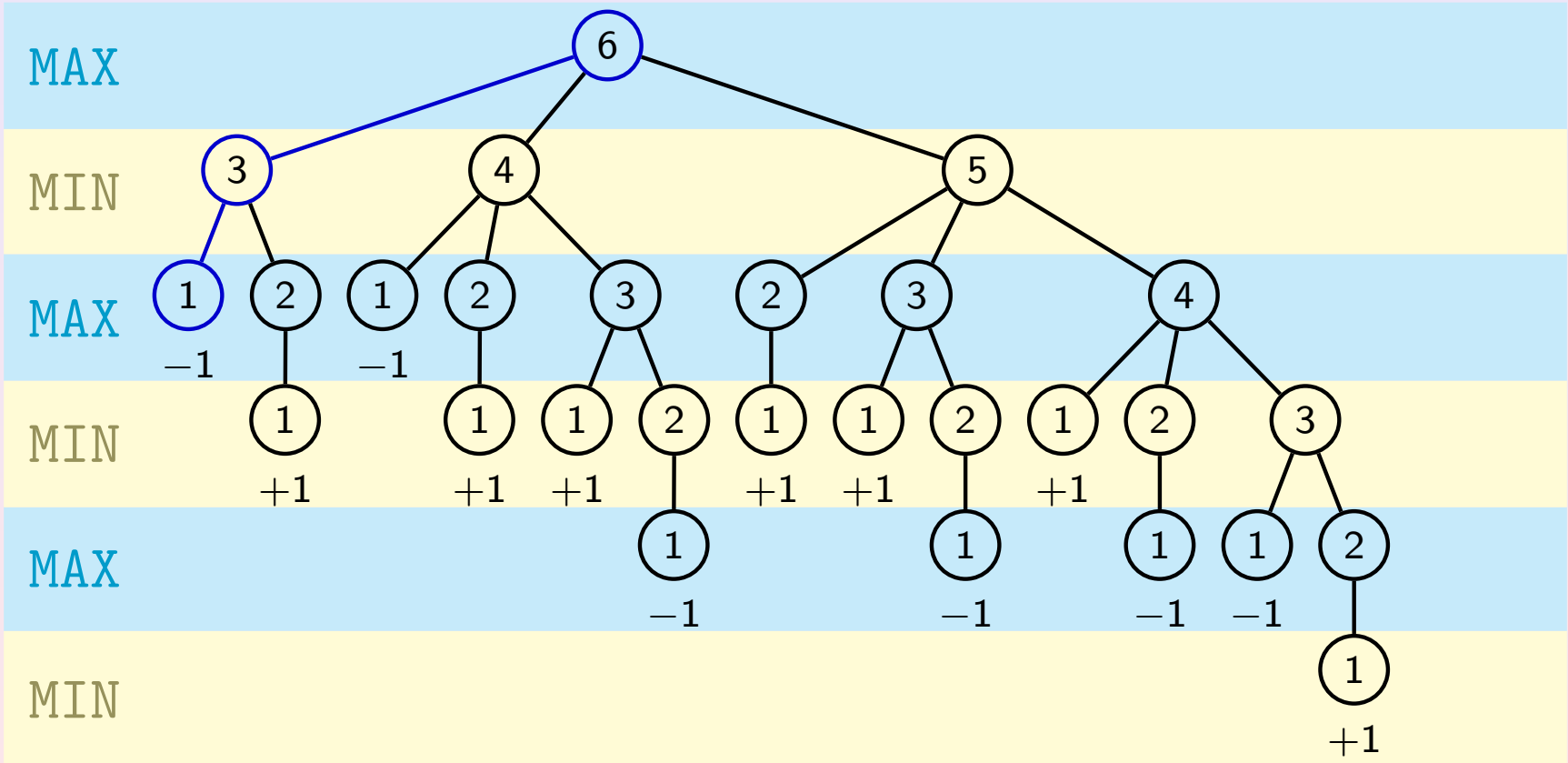
Ávore de Busca: Busca em profundidade



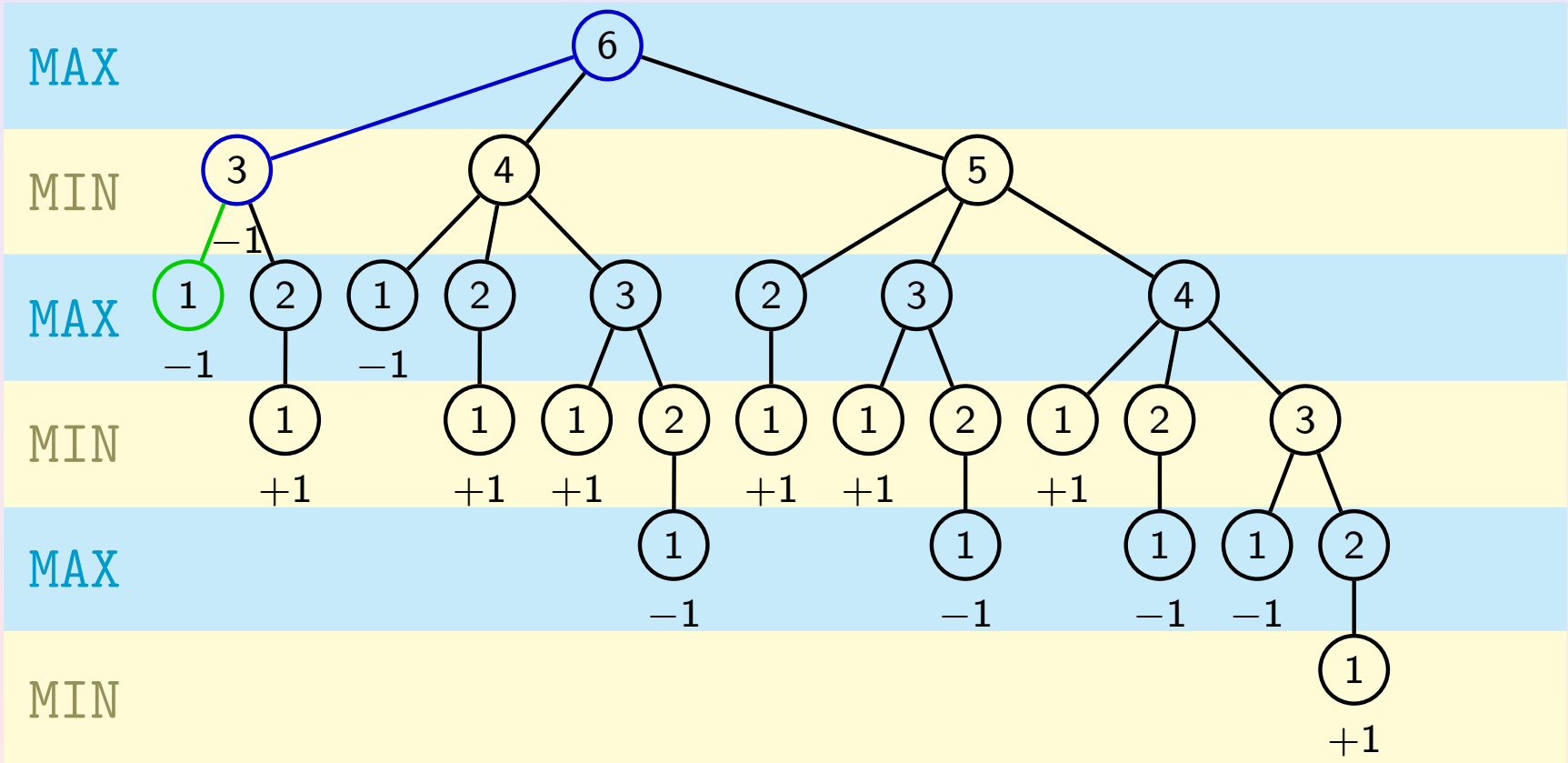
Ávore de Busca: Busca em profundidade



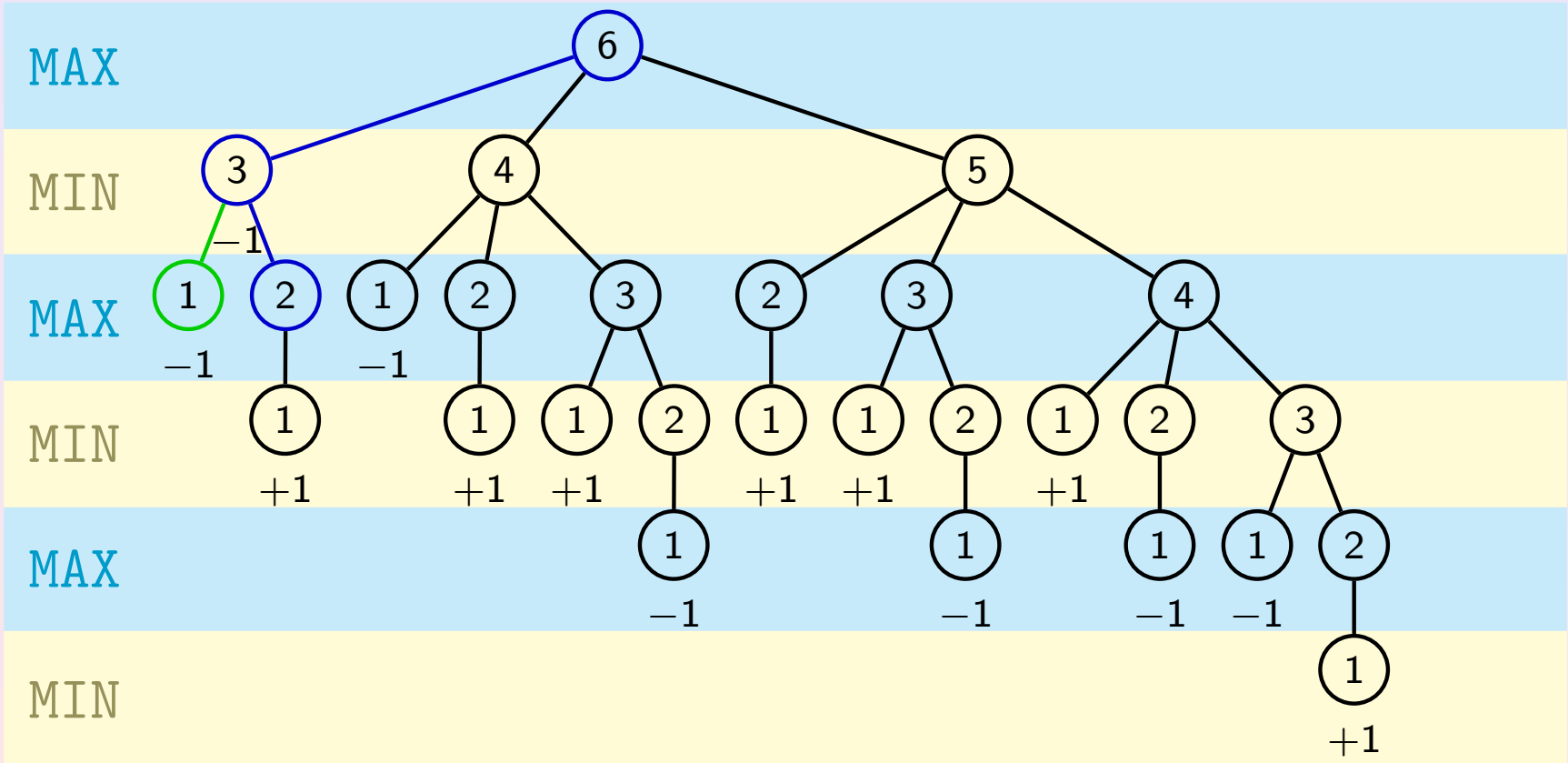
Ávore de Busca: Busca em profundidade



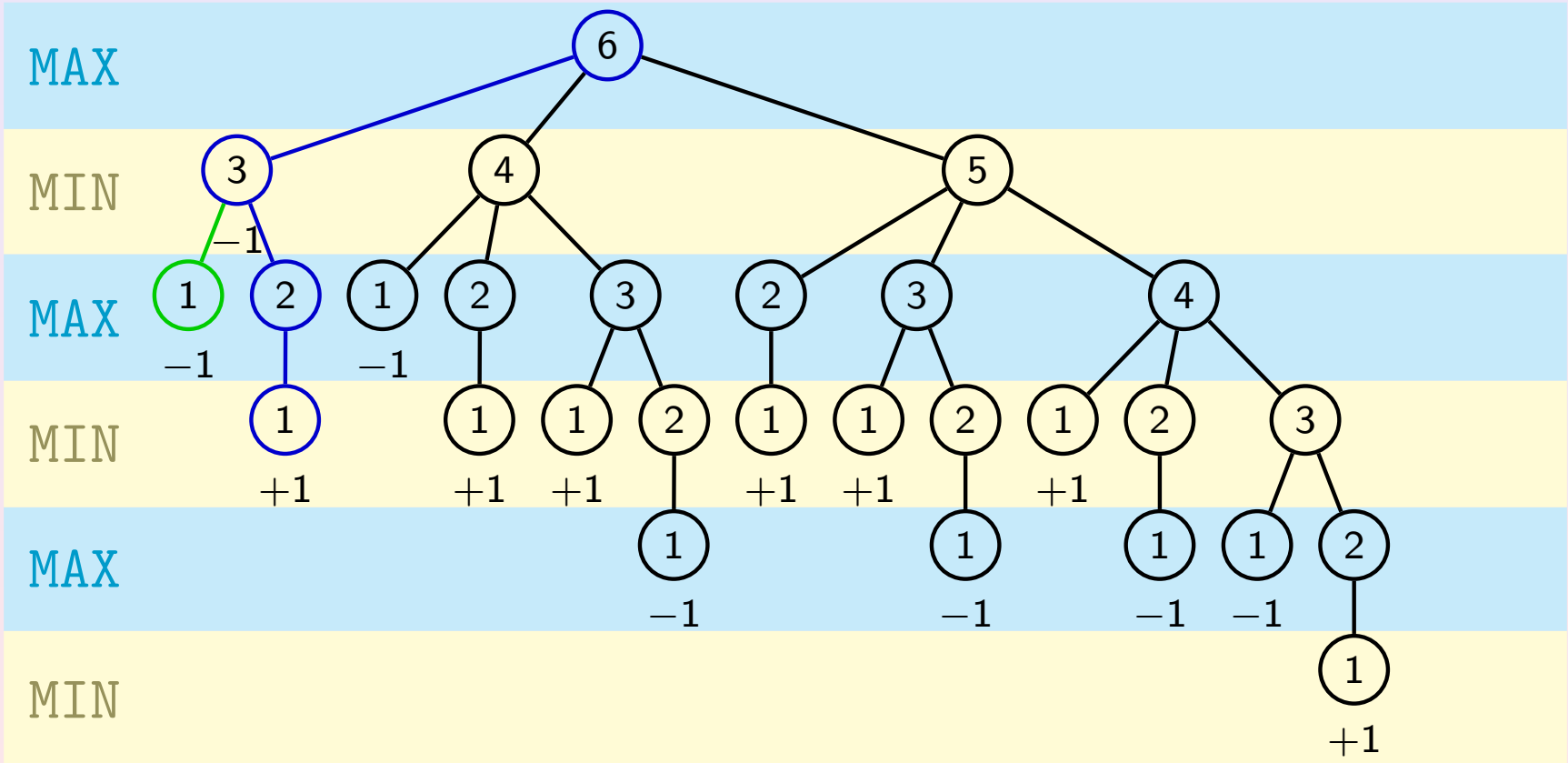
Ávore de Busca: Busca em profundidade



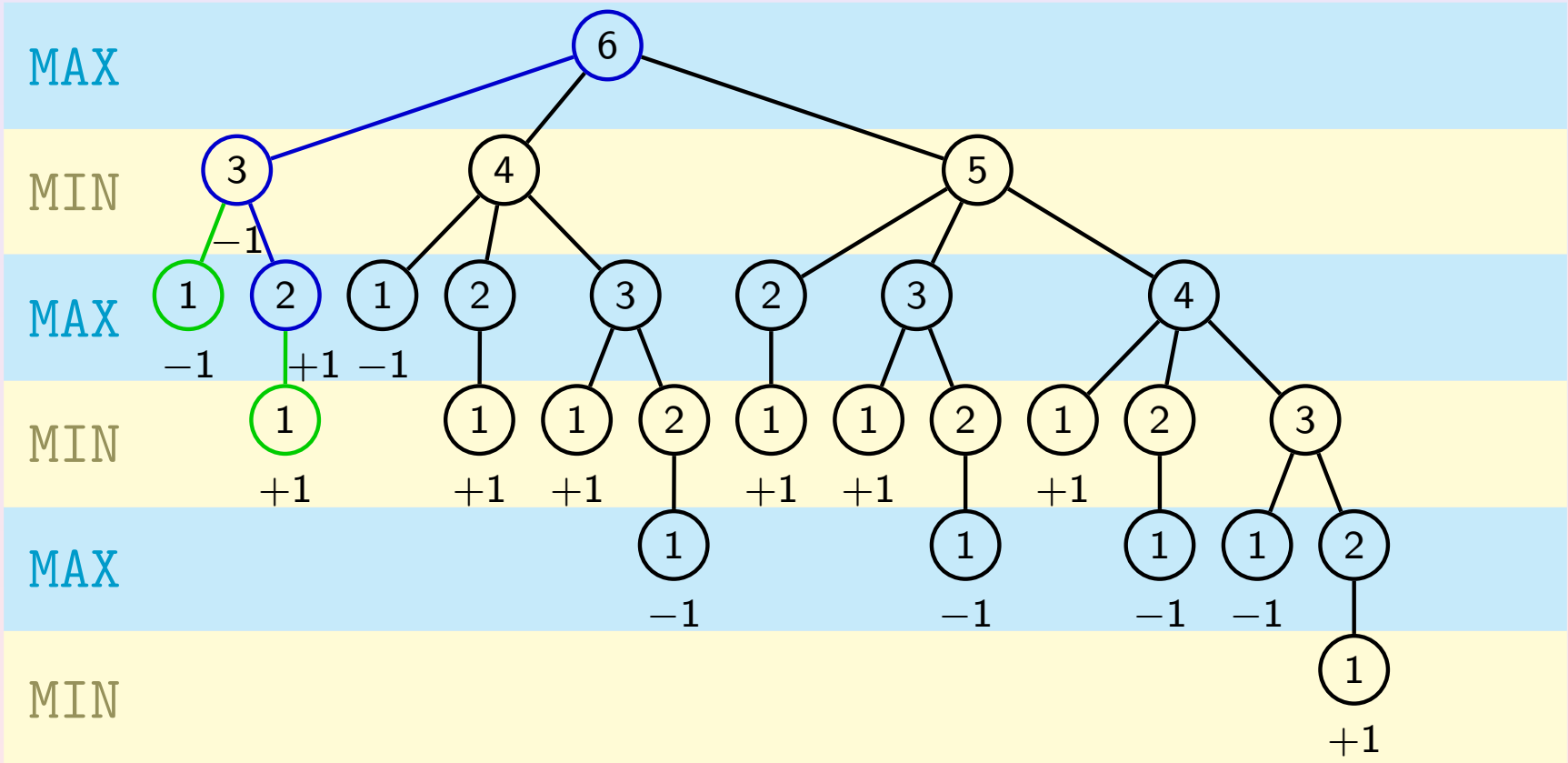
Ávore de Busca: Busca em profundidade



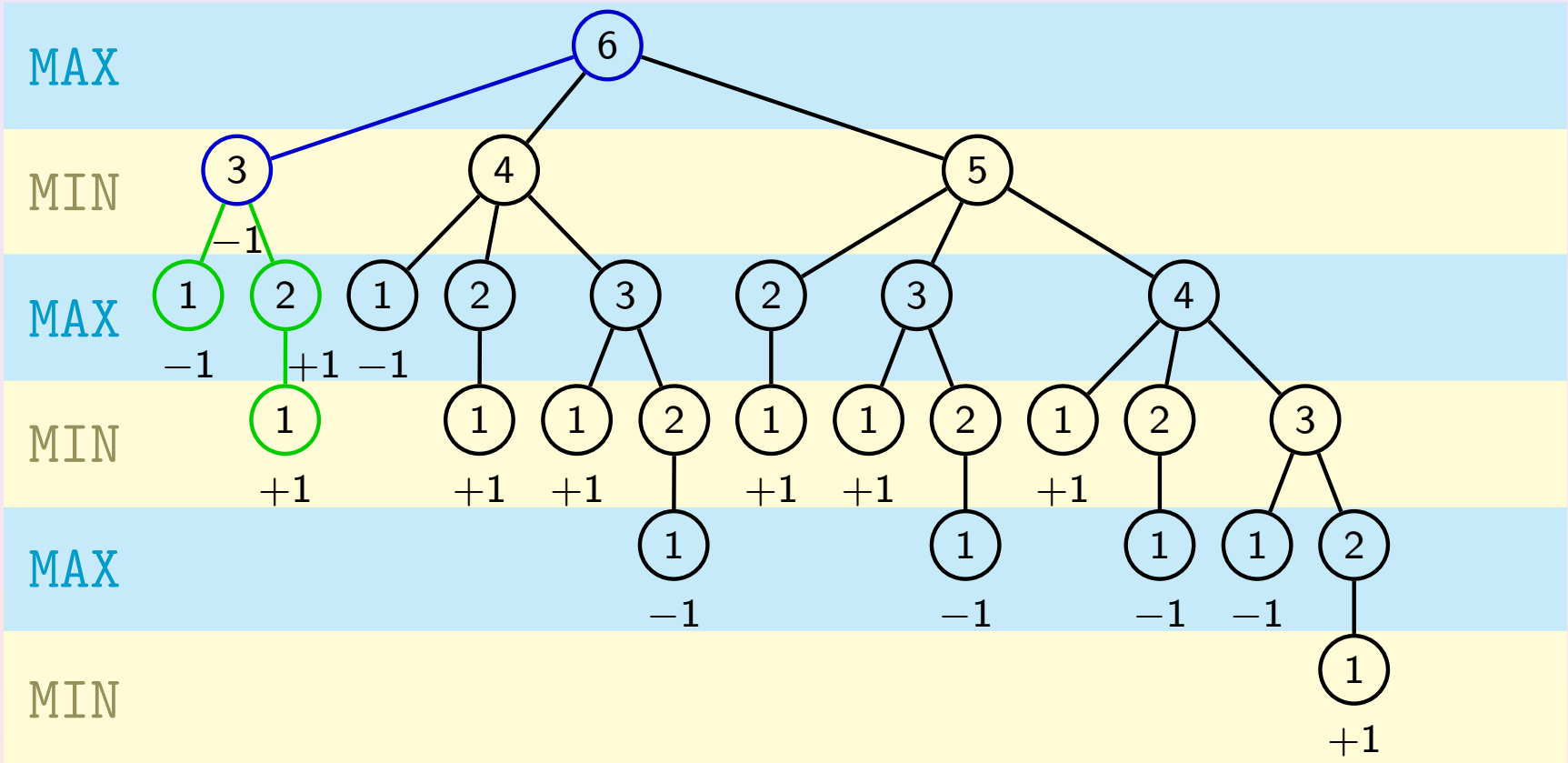
Ávore de Busca: Busca em profundidade



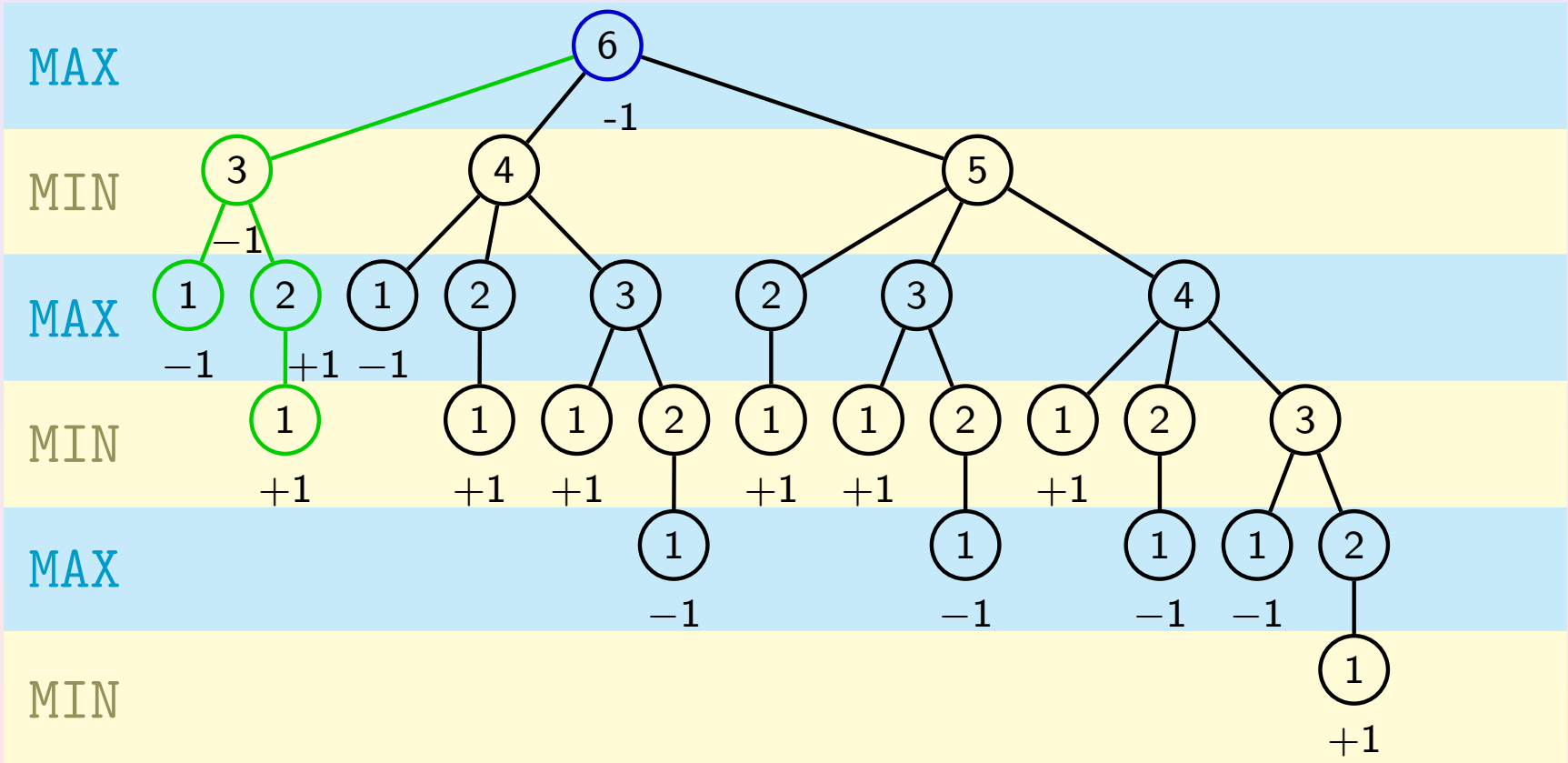
Ávore de Busca: Busca em profundidade



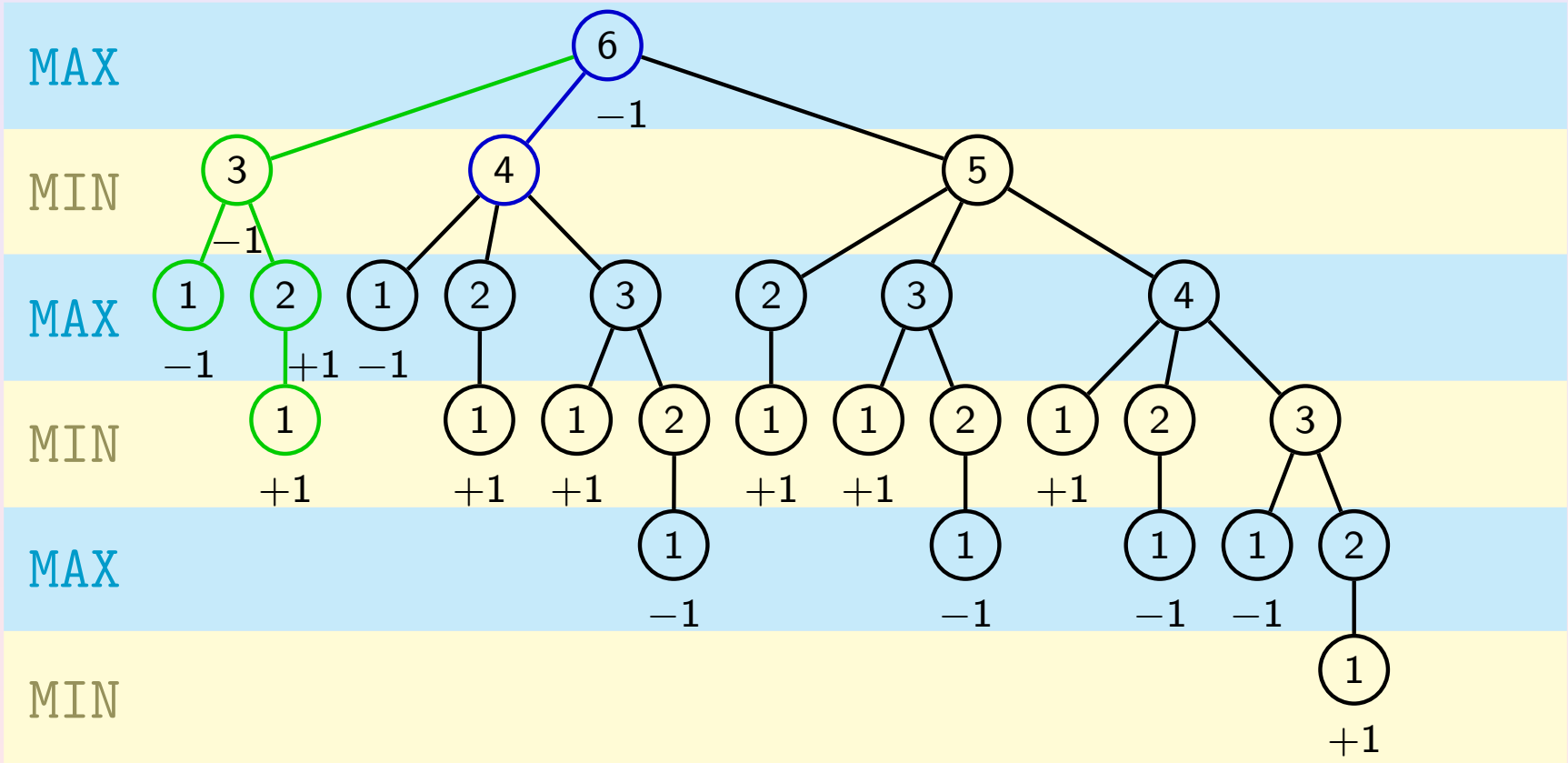
Ávore de Busca: Busca em profundidade



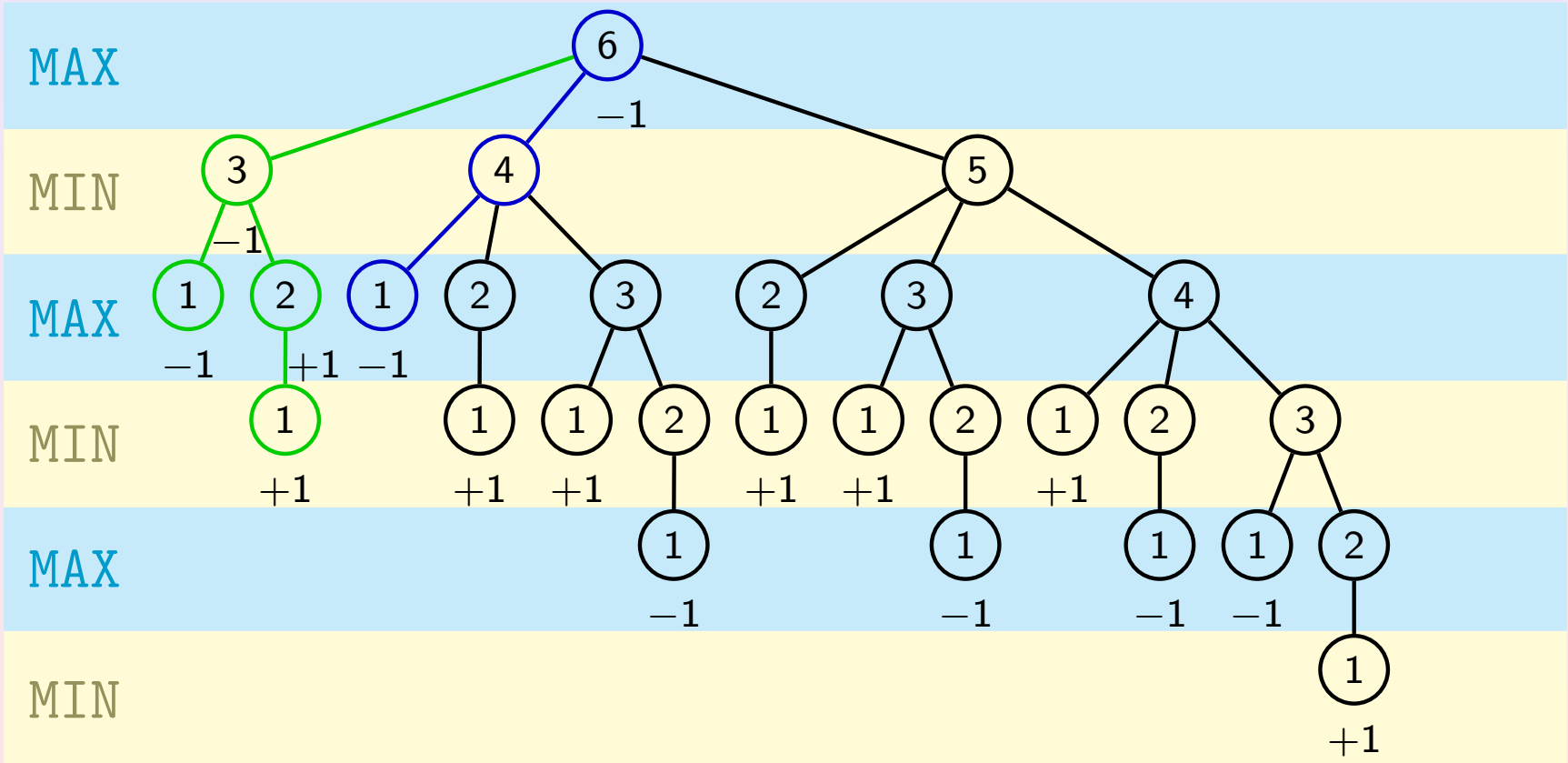
Ávore de Busca: Busca em profundidade



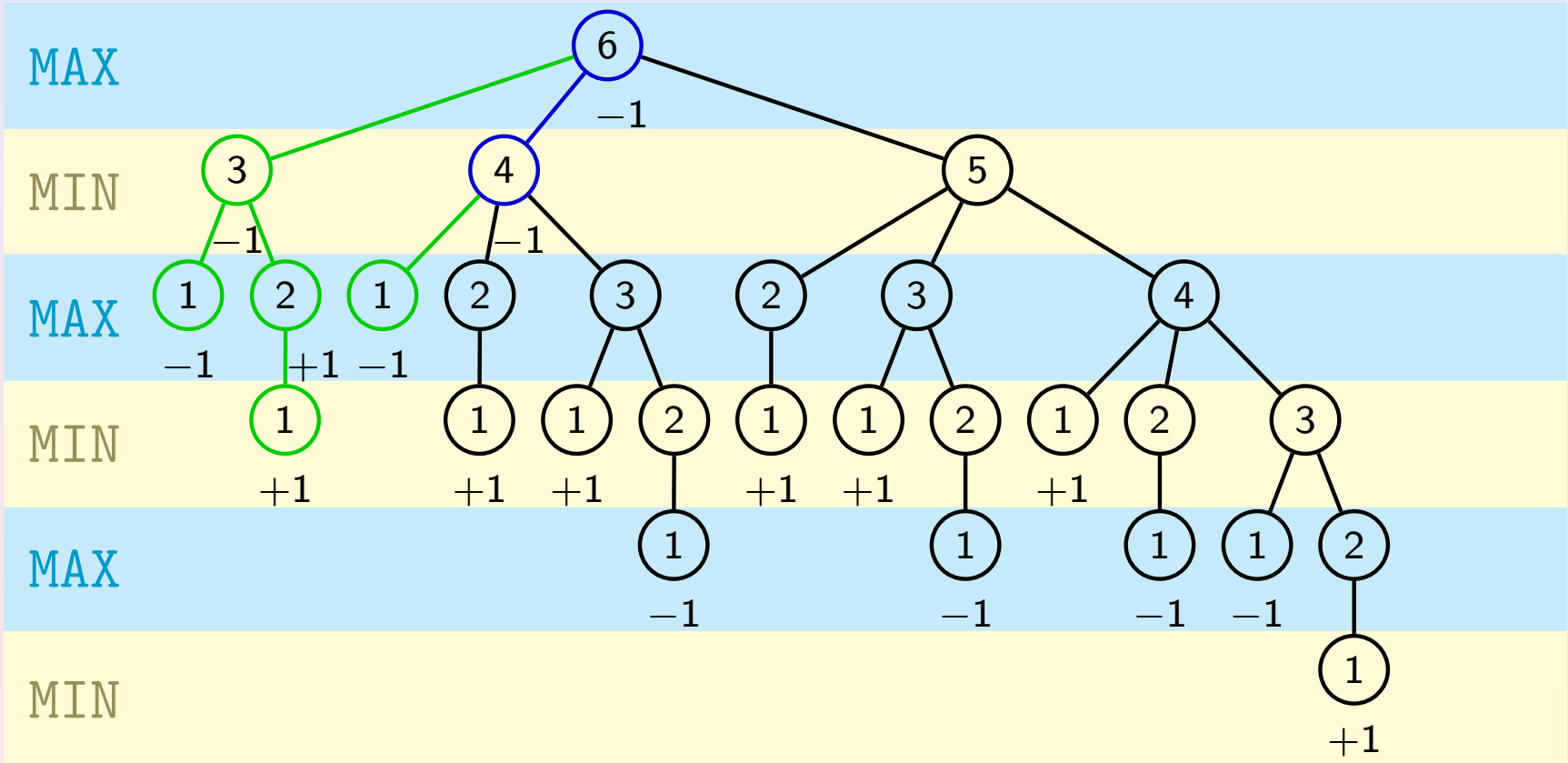
Ávore de Busca: Busca em profundidade



Ávore de Busca: Busca em profundidade



Ávore de Busca: Busca em profundidade



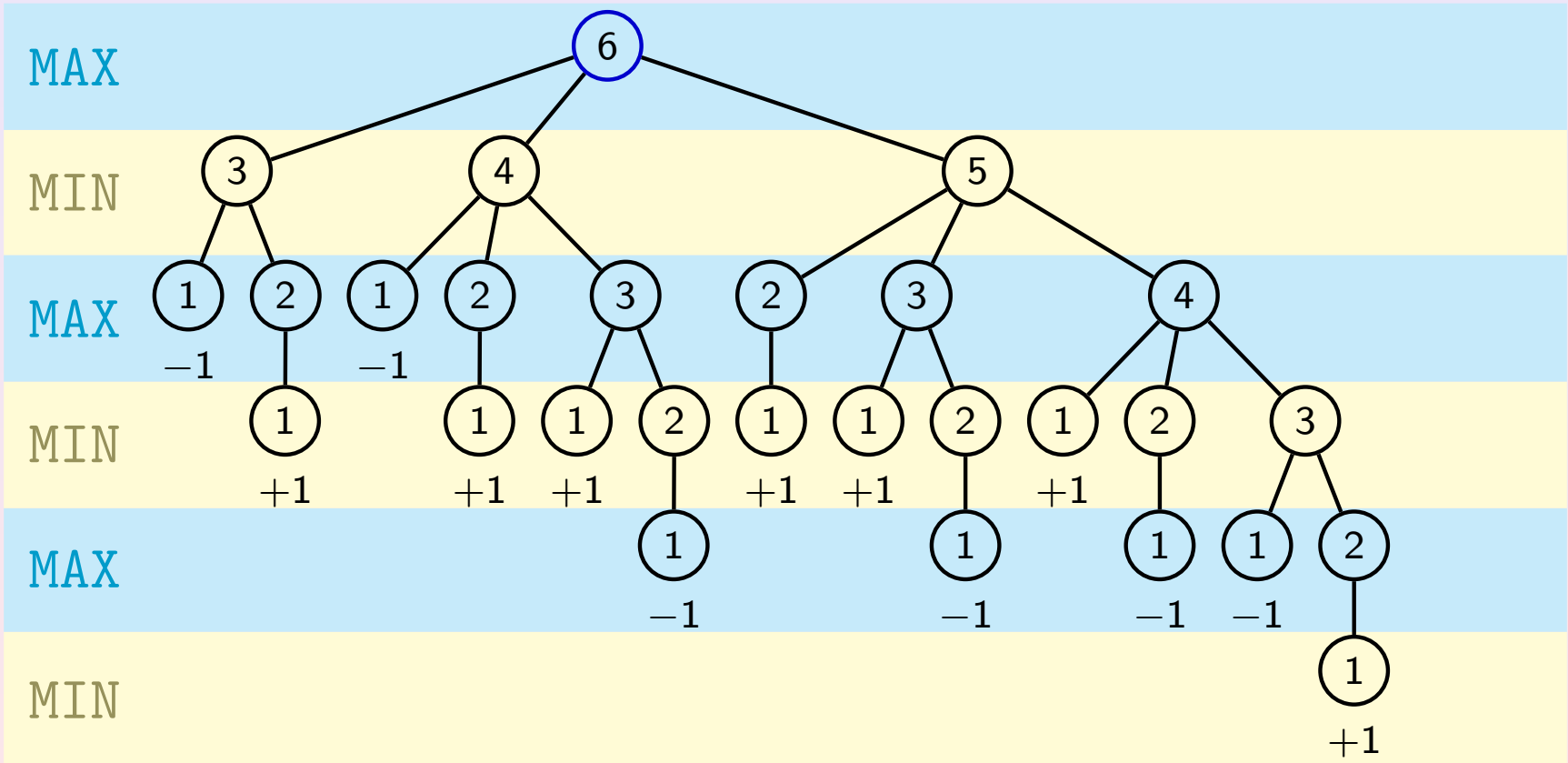
Árvore de Busca: Busca em profundidade

- Vamos analisar, vale a pena pesquisar os descendentes “2” e “3” do nó “4”?
  - O nó “4” está num nível min, e já possui o valor “-1”.
  - O nó “6” acima, num nível max, também já possui o valor “-1”.
  - Se o nó “4” encontrar nos descendentes um valor maior que “-1”, ele não vai alterar o valor, pois ele está em um nível MIN.
  - Se o nó “4” encontrar um valor menor que “-1”, não adianta alterar seu valor, pois o nó “6” ascendente dele, em um nível MAX, não irá considerar este nó.
- Podemos otimizar esta busca em profundidade fazendo uma poda nos ramos de busca.

Preenchendo a árvore de busca: Poda Alfa-Beta

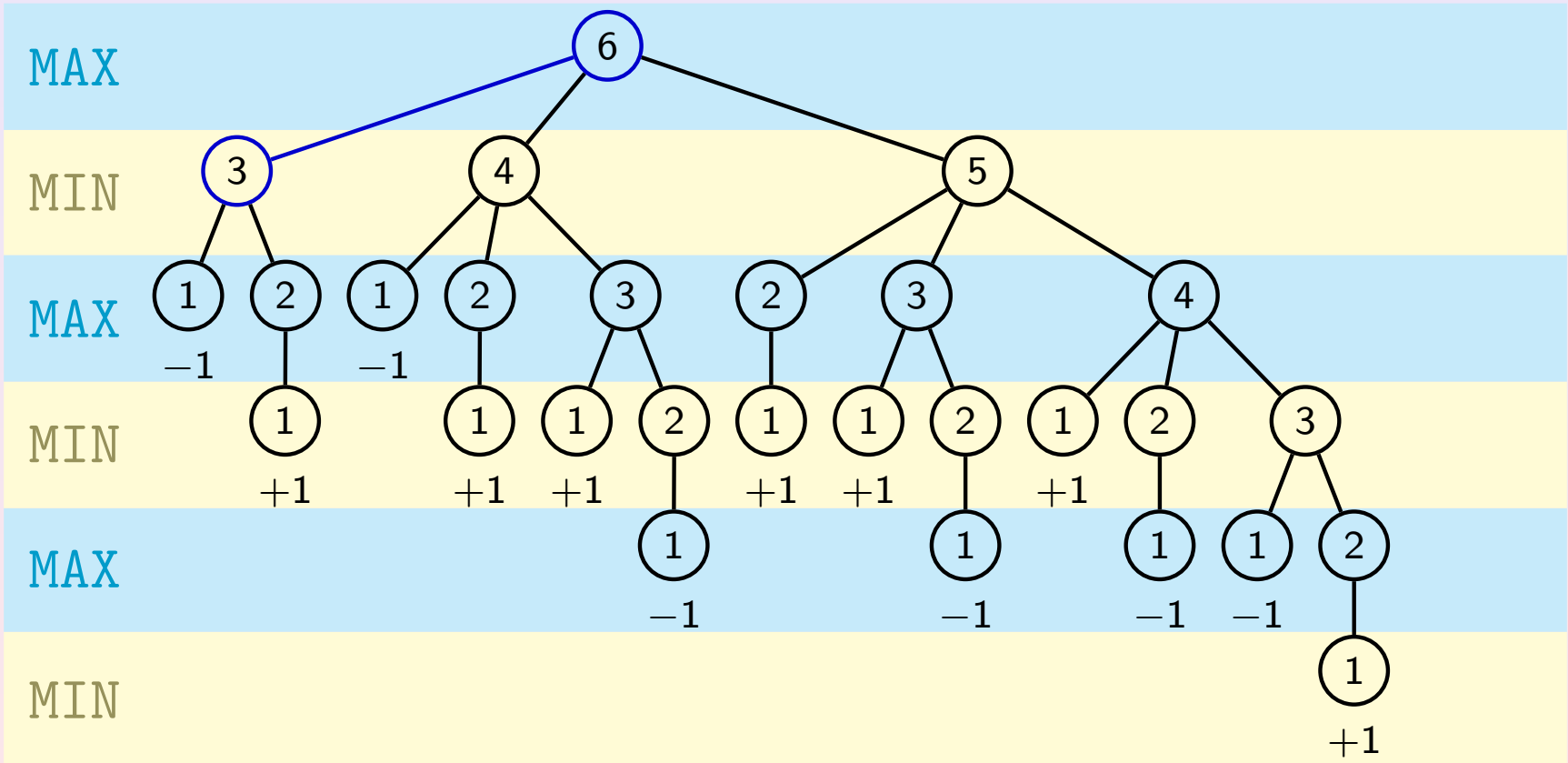
- Normalmente para preencher a árvore de busca, utiliza-se uma busca em profundidade.
- O custo de preencher a árvore toda é muito alto, logo a estratégia é podar a busca, quando observarmos que aquele ramo não será útil, ou não irá melhorar o resultado já obtido.
  - Alpha: para os MAX - é o máximo (melhor) valor encontrado até então nos descendentes dos nós MAX.
  - Beta: para os MIN - é o mínimo (melhor) valor encontrado até então nos descendentes dos nós MIN.
- Ao obter um valor:
  - Sendo um nó MAX: Se o valor alfa é maior ou igual ao beta ancestral, então não precisa mais visitar os descendentes.
  - Sendo um nó MIN: Se o valor beta é menor ou igual a um alfa ancestral, então não precisa mais visitar os descententes.

Ávore de Busca: Busca em profundidade



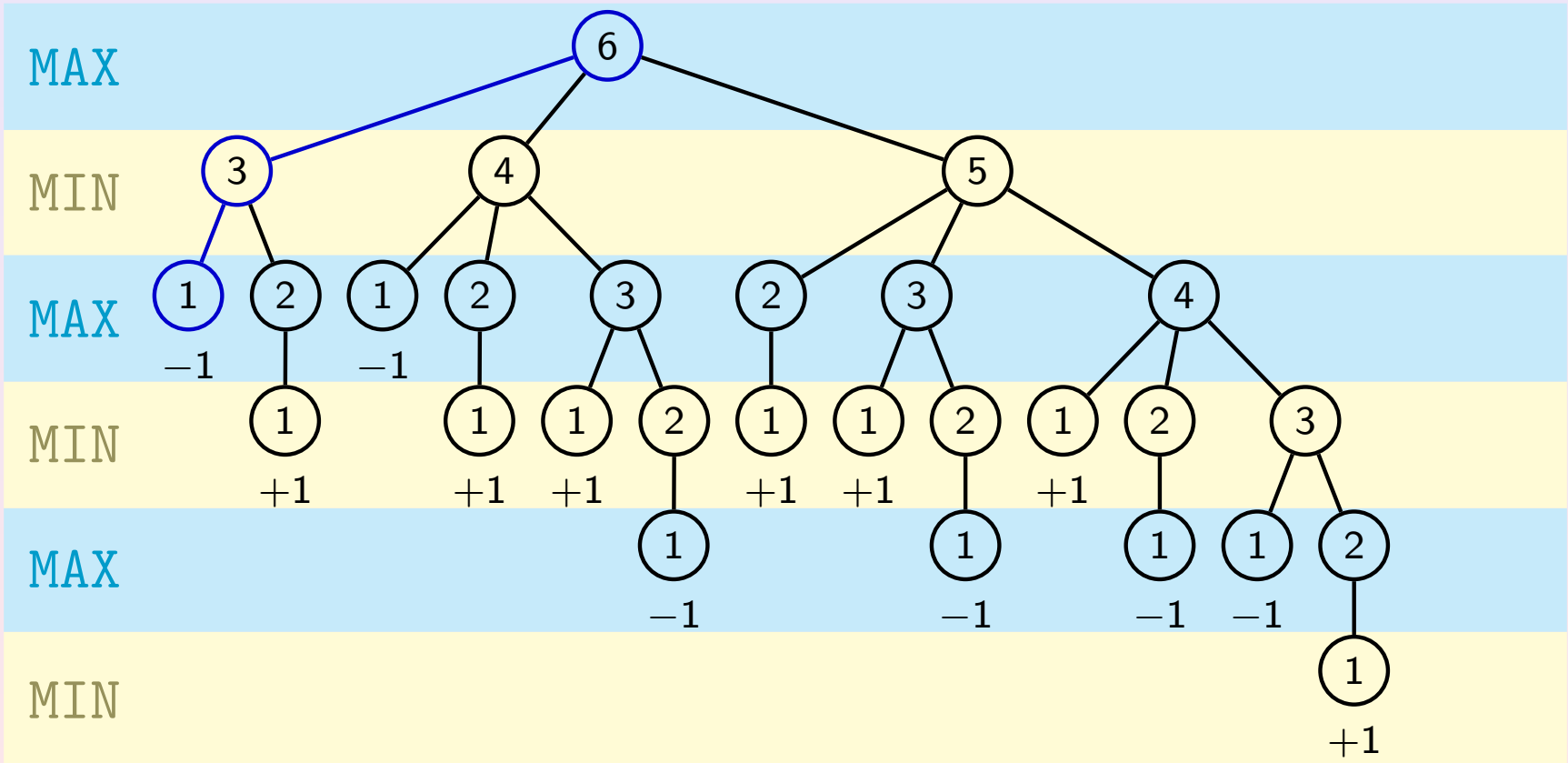
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



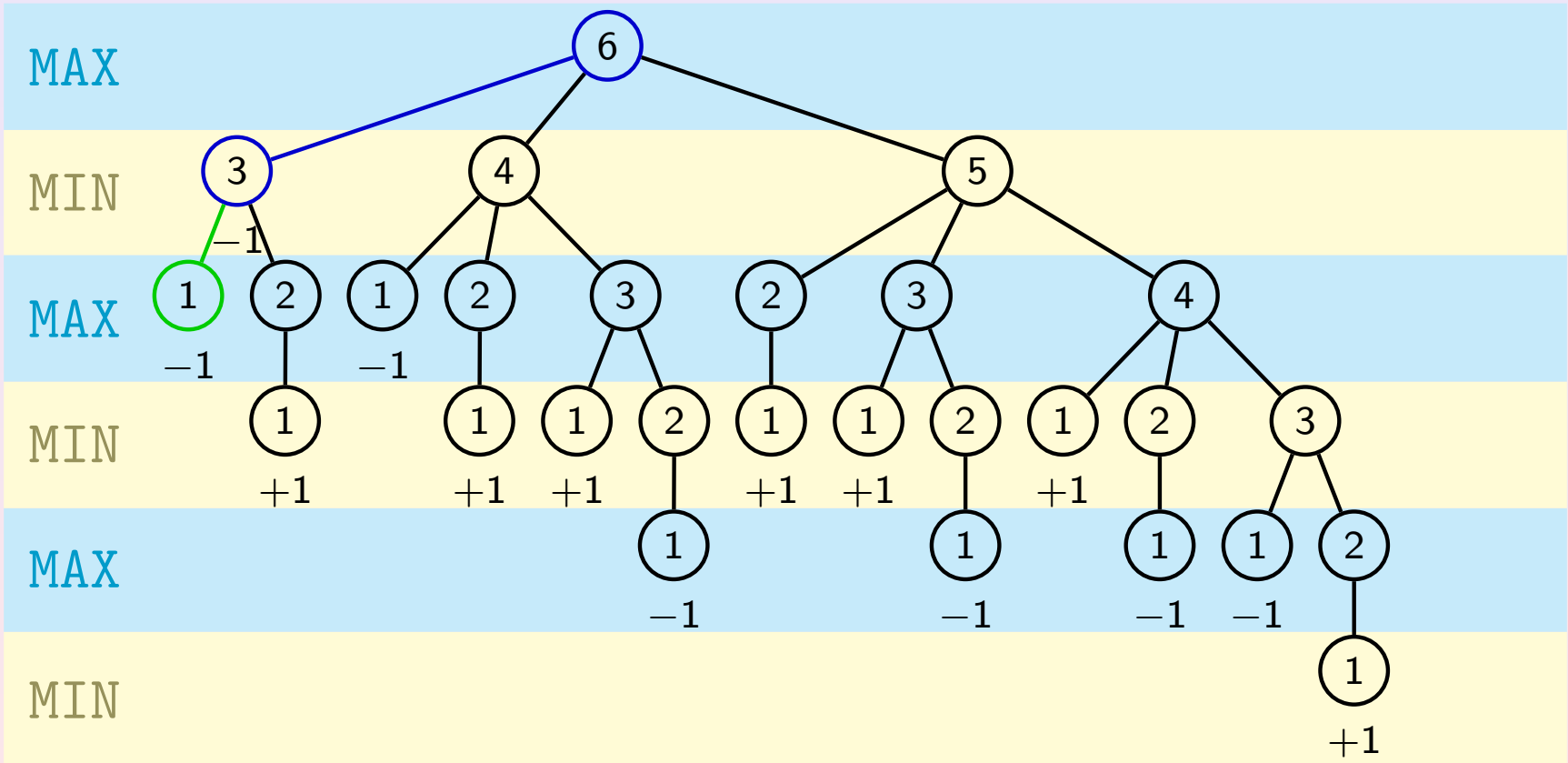
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



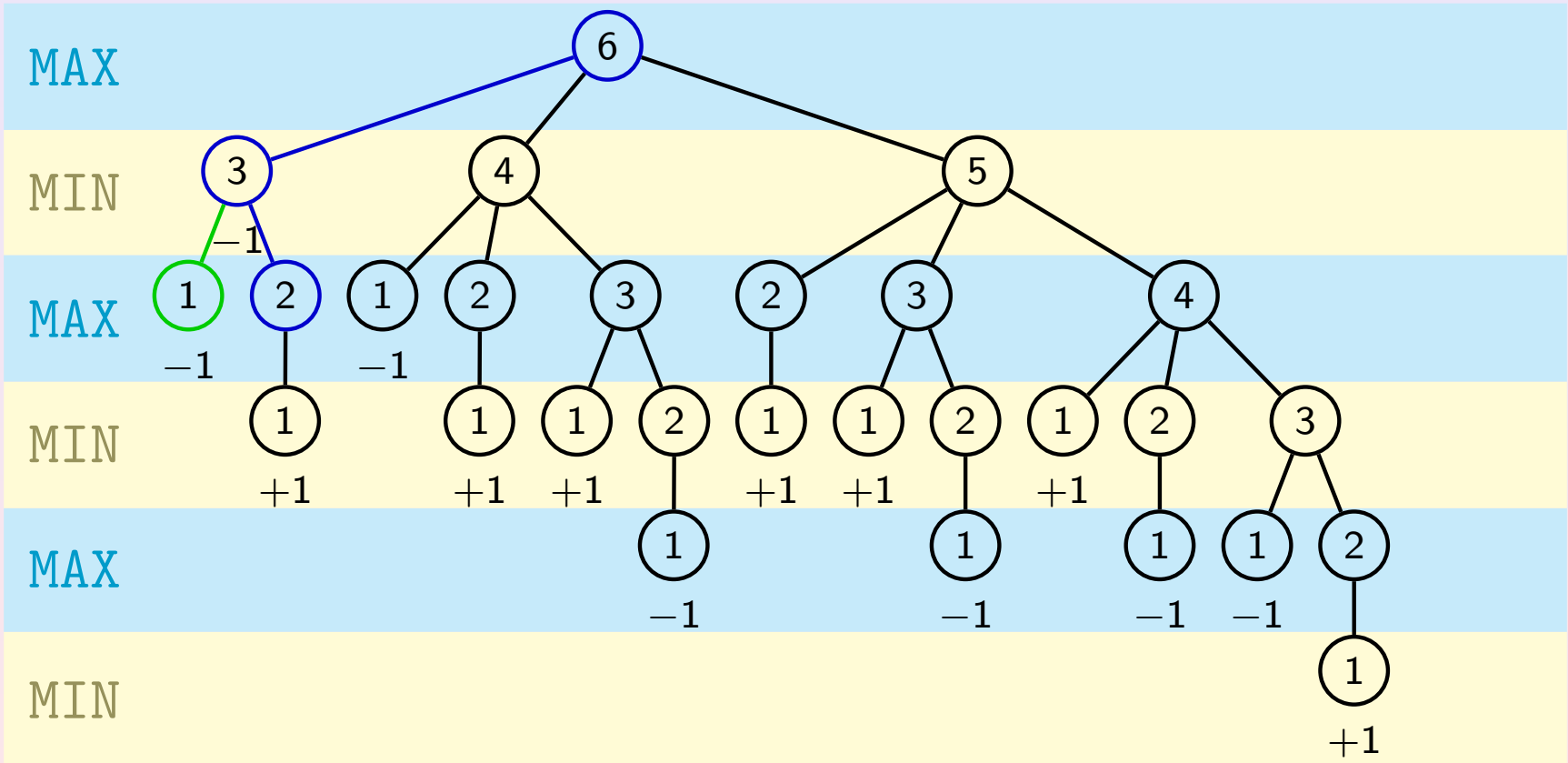
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



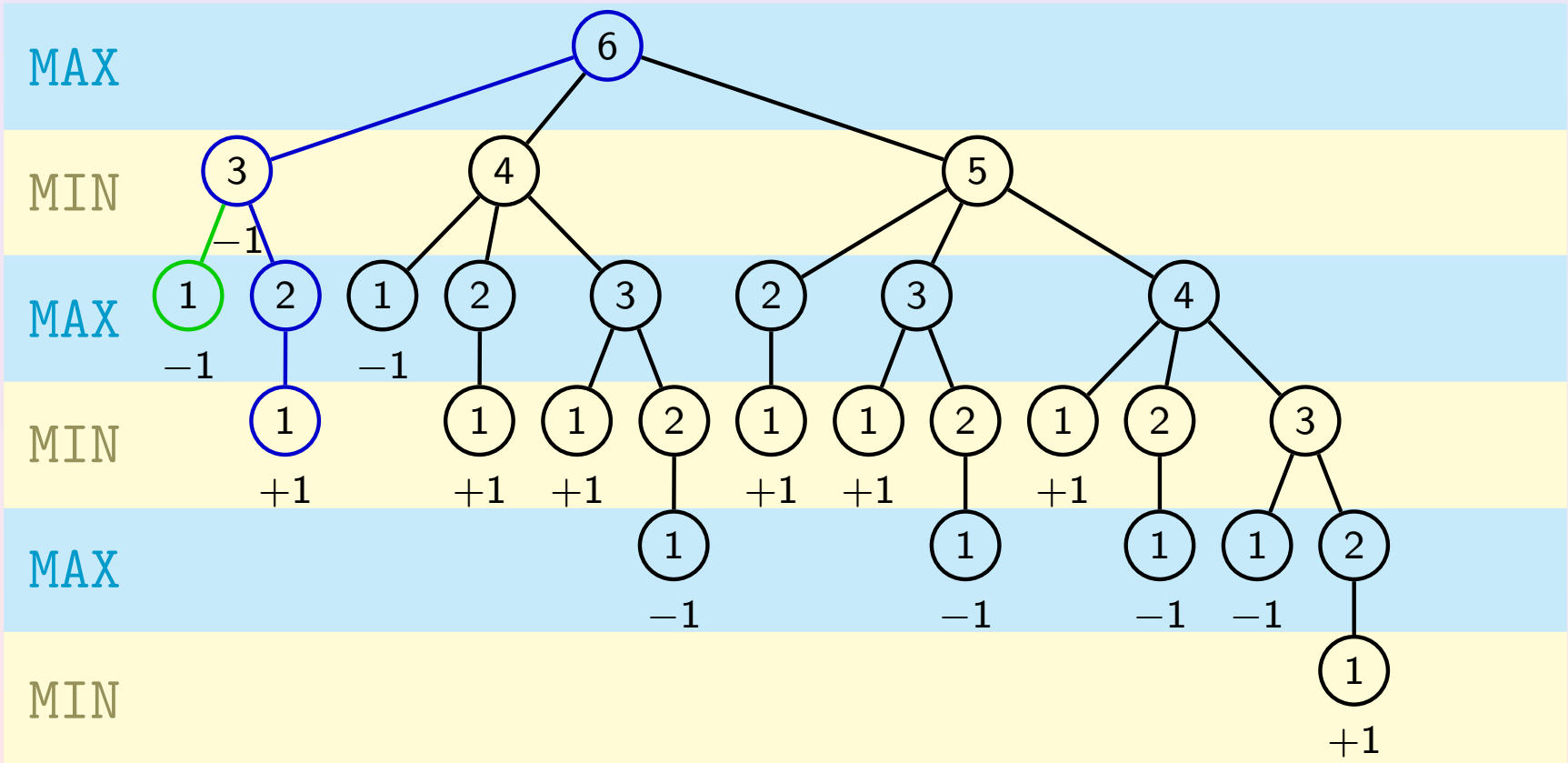
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



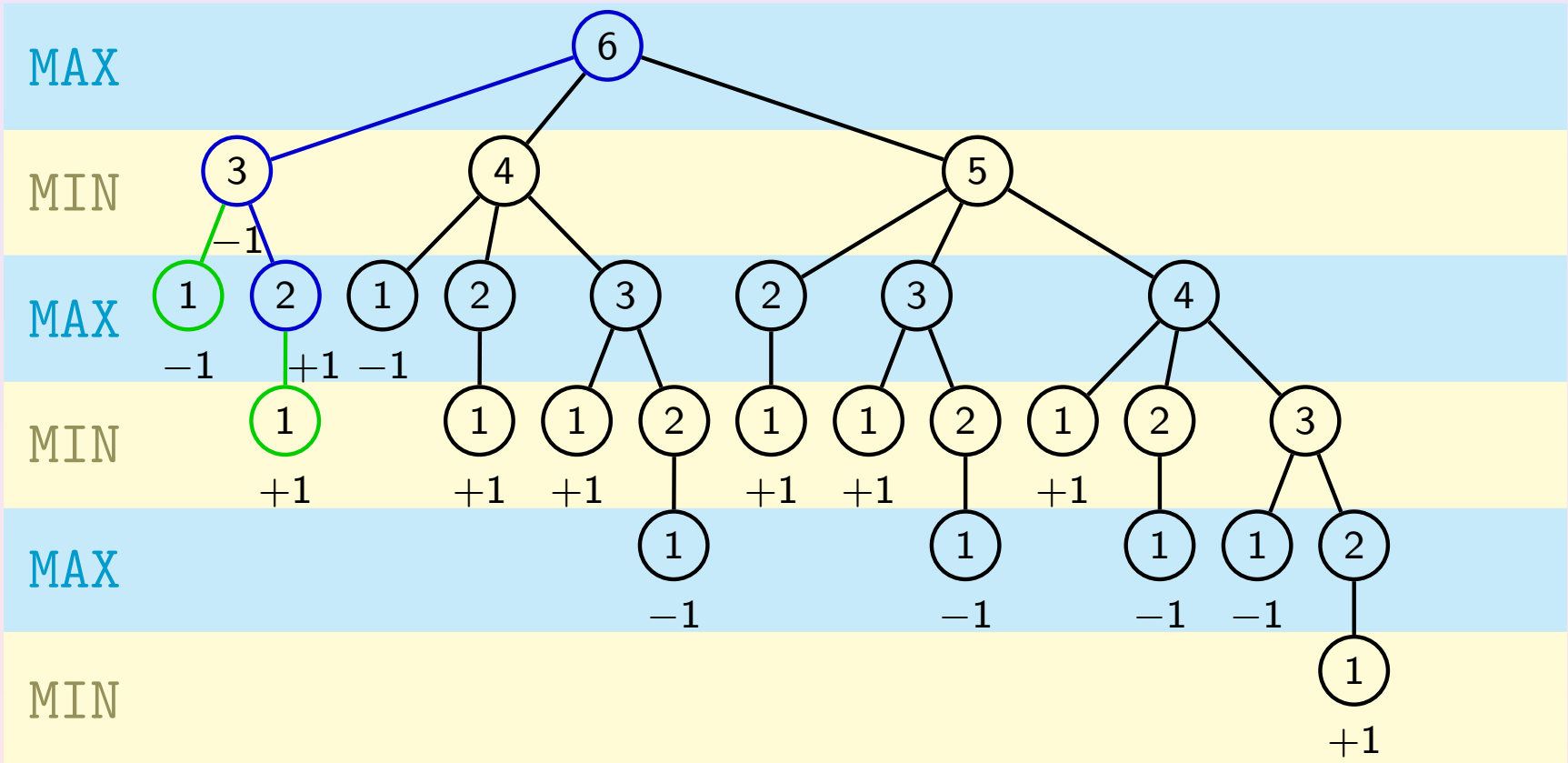
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



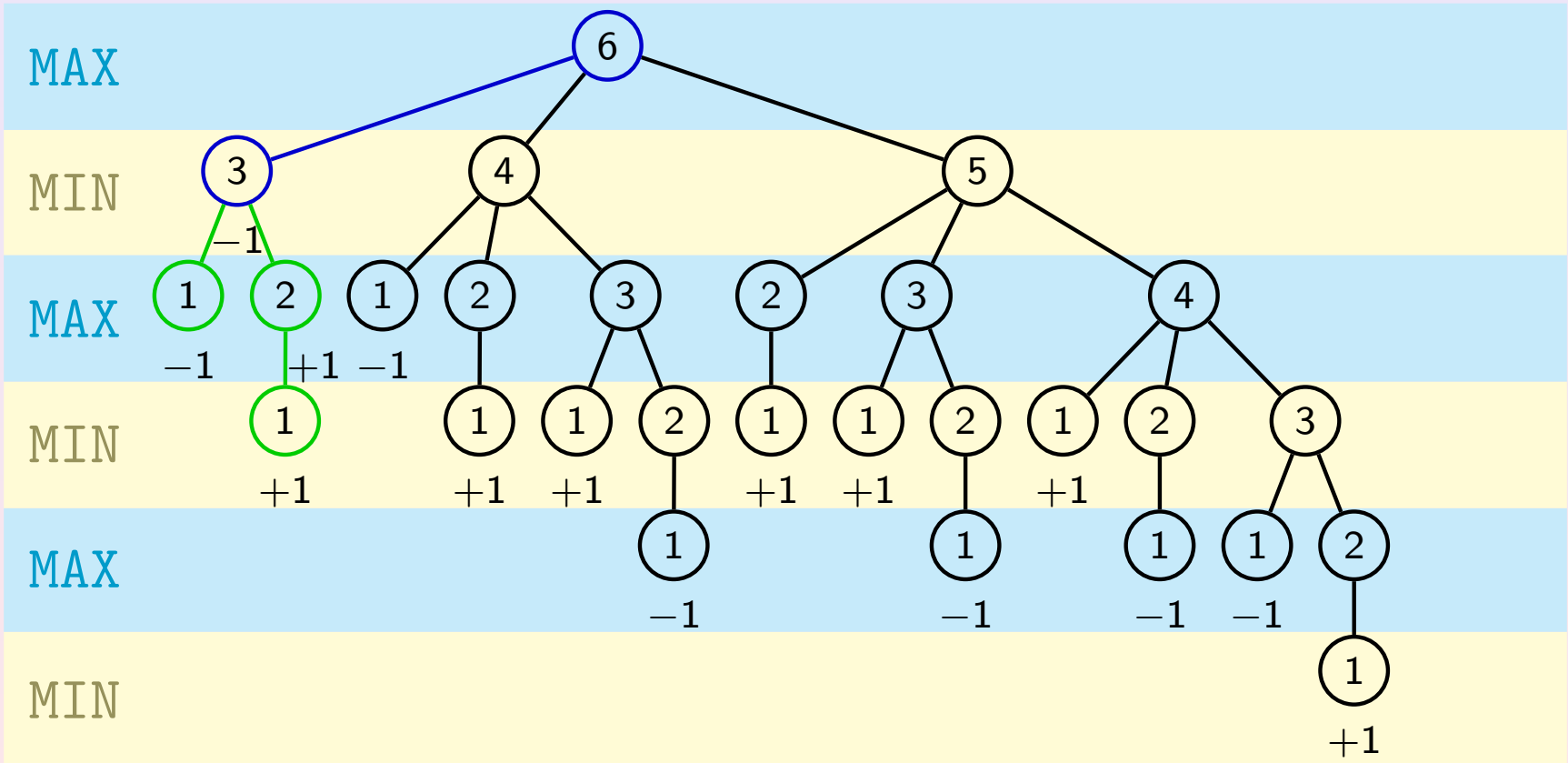
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



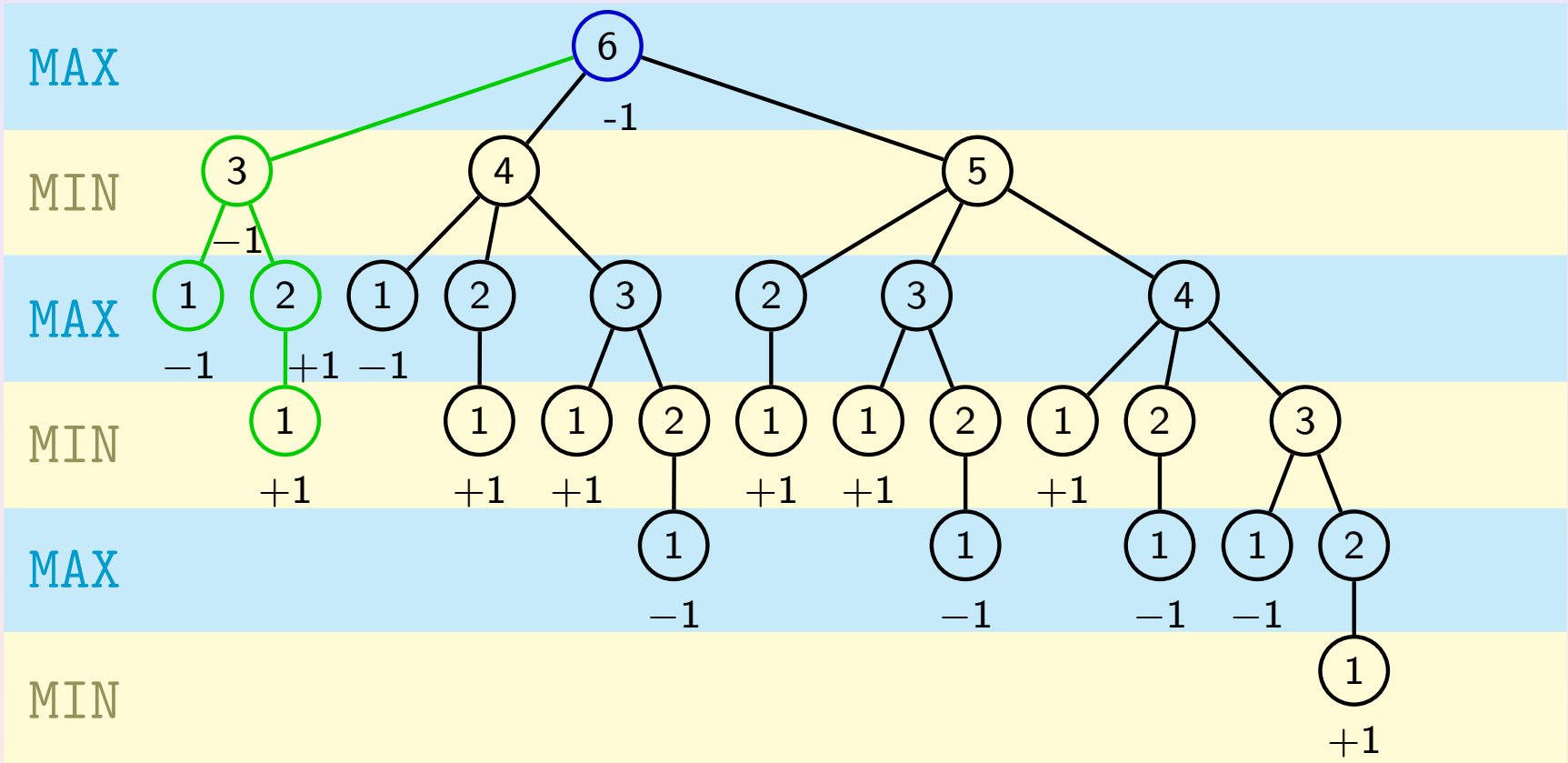
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



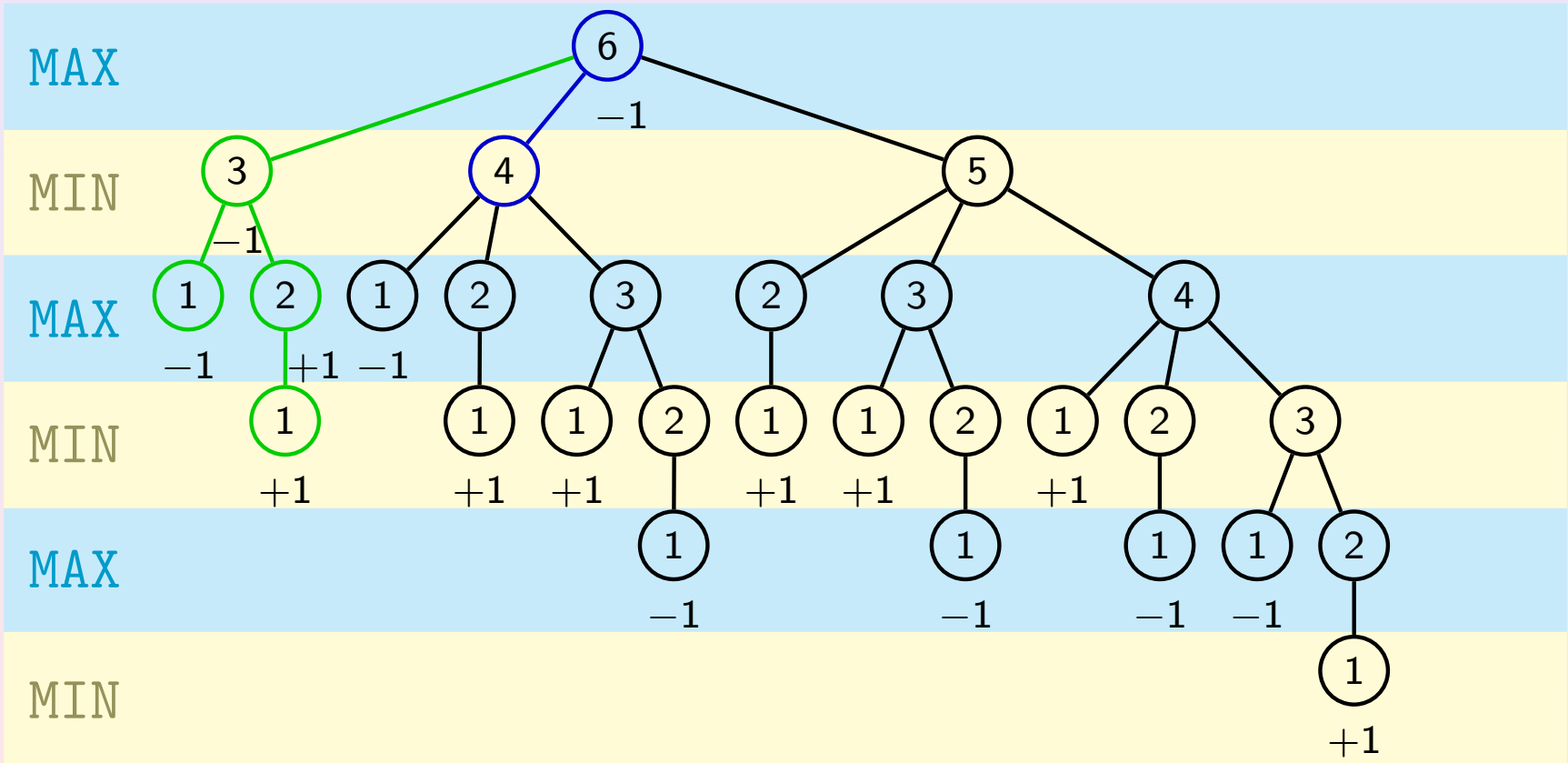
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



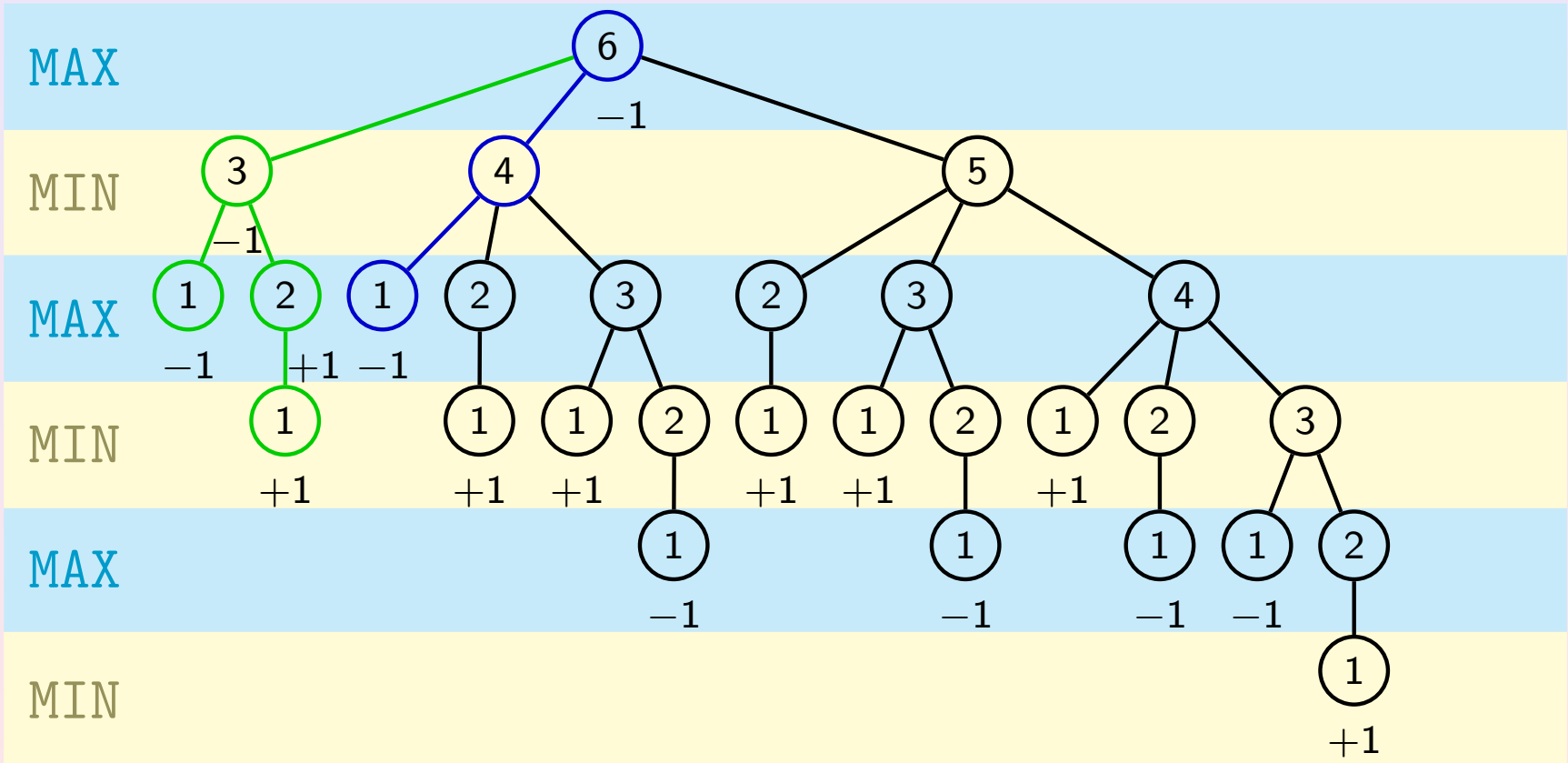
Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



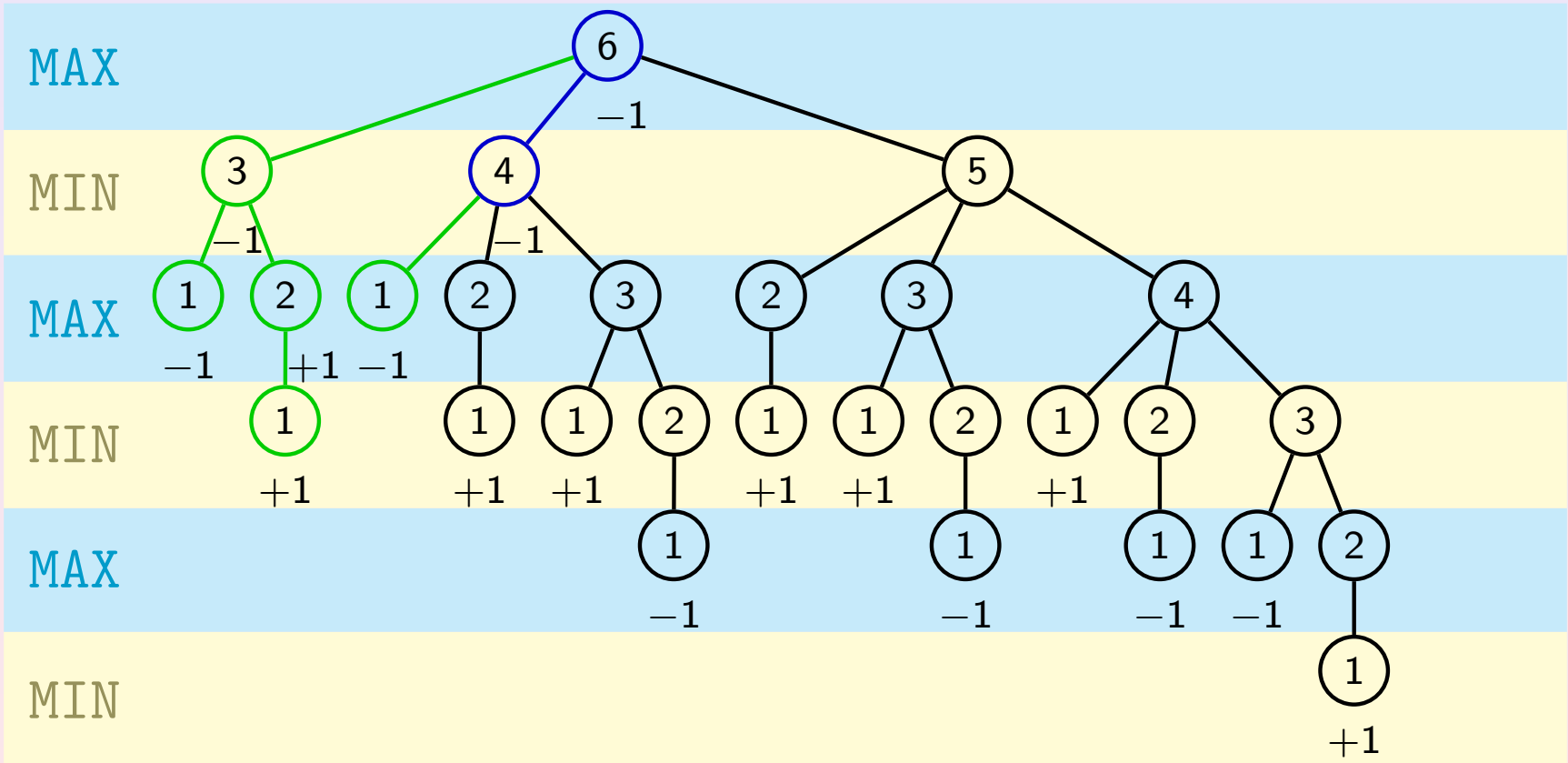
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



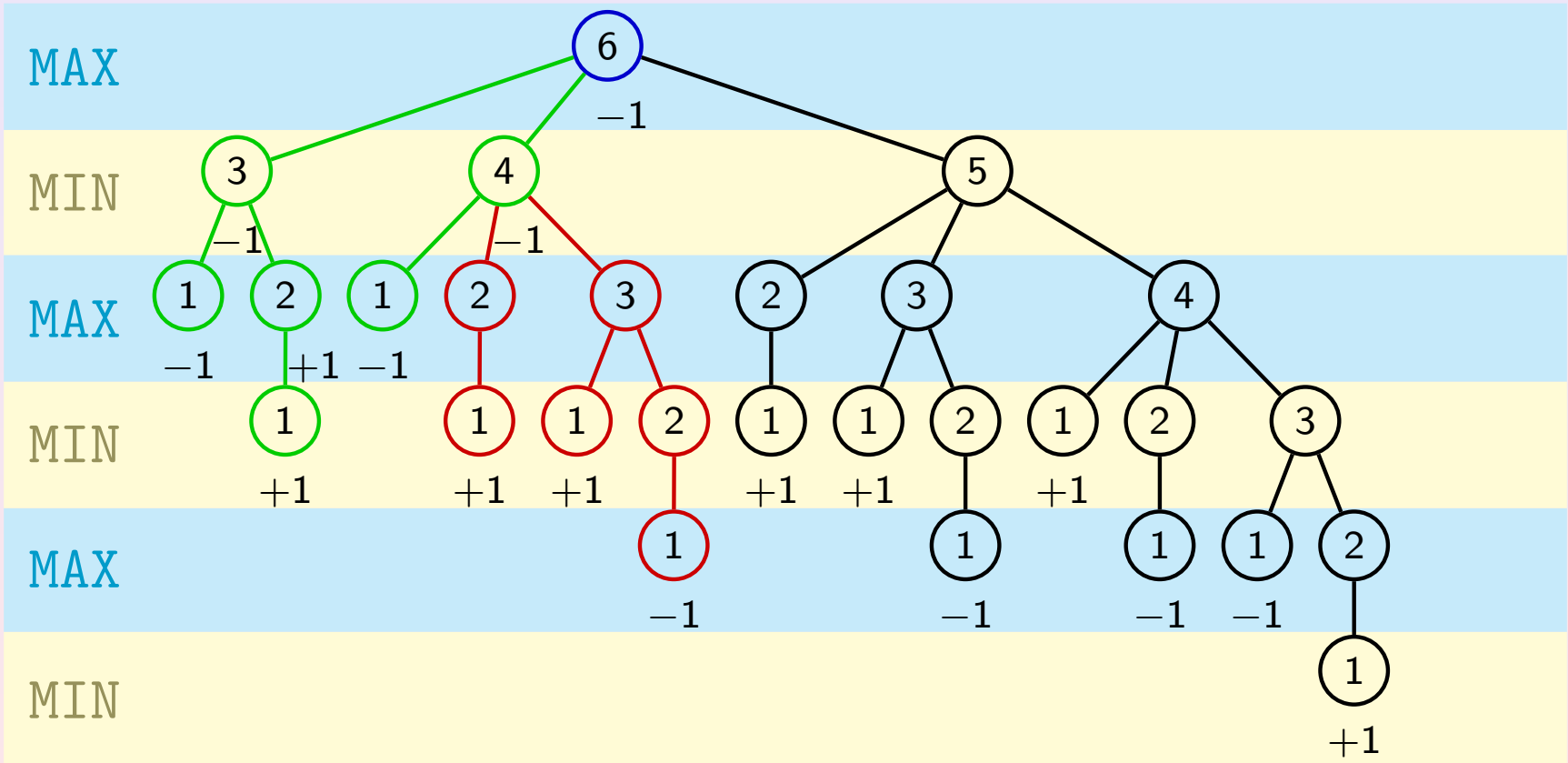
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



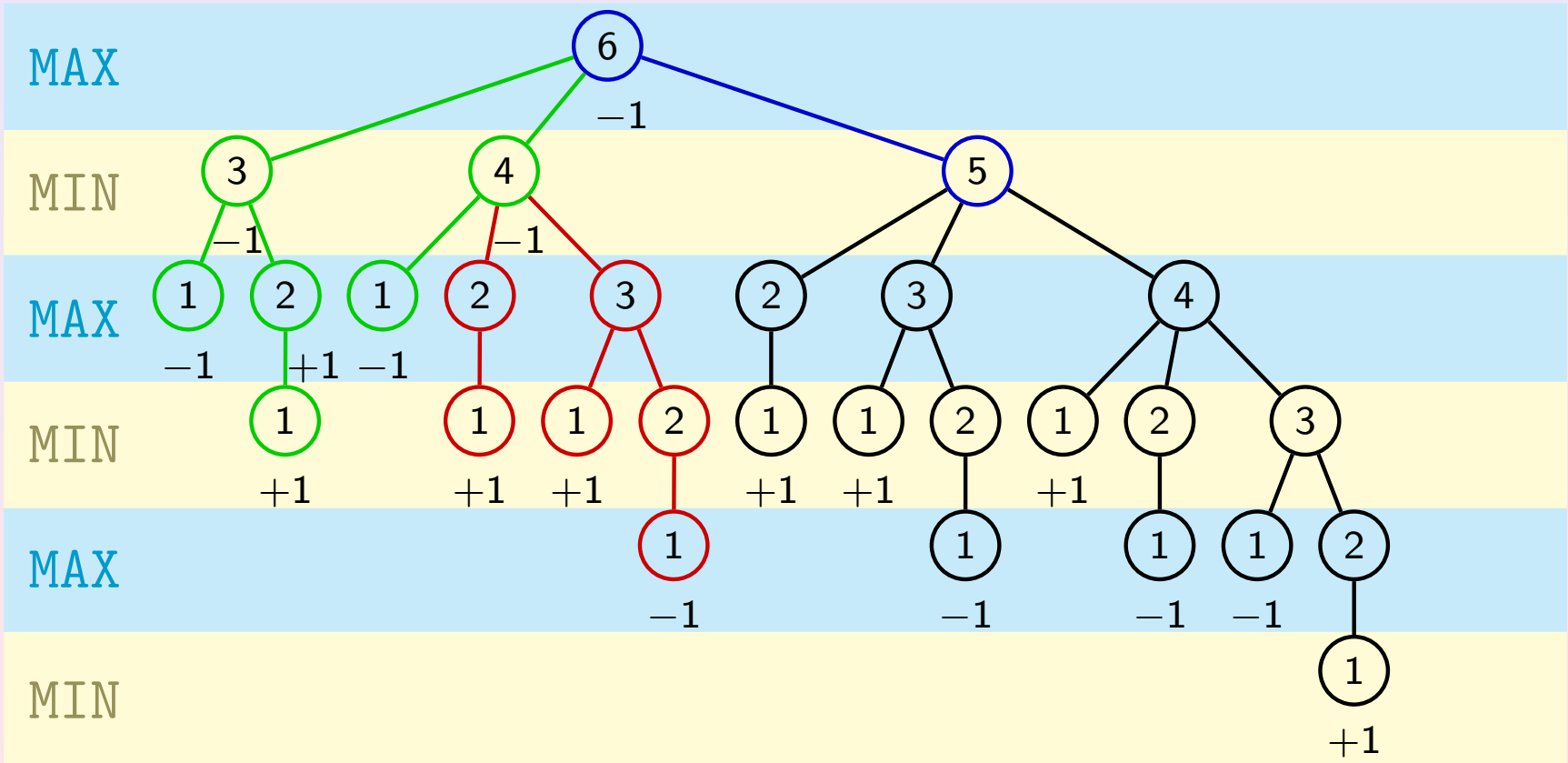
Beta: ? Alfa: -1 → PODA!

Ávore de Busca: Busca em profundidade



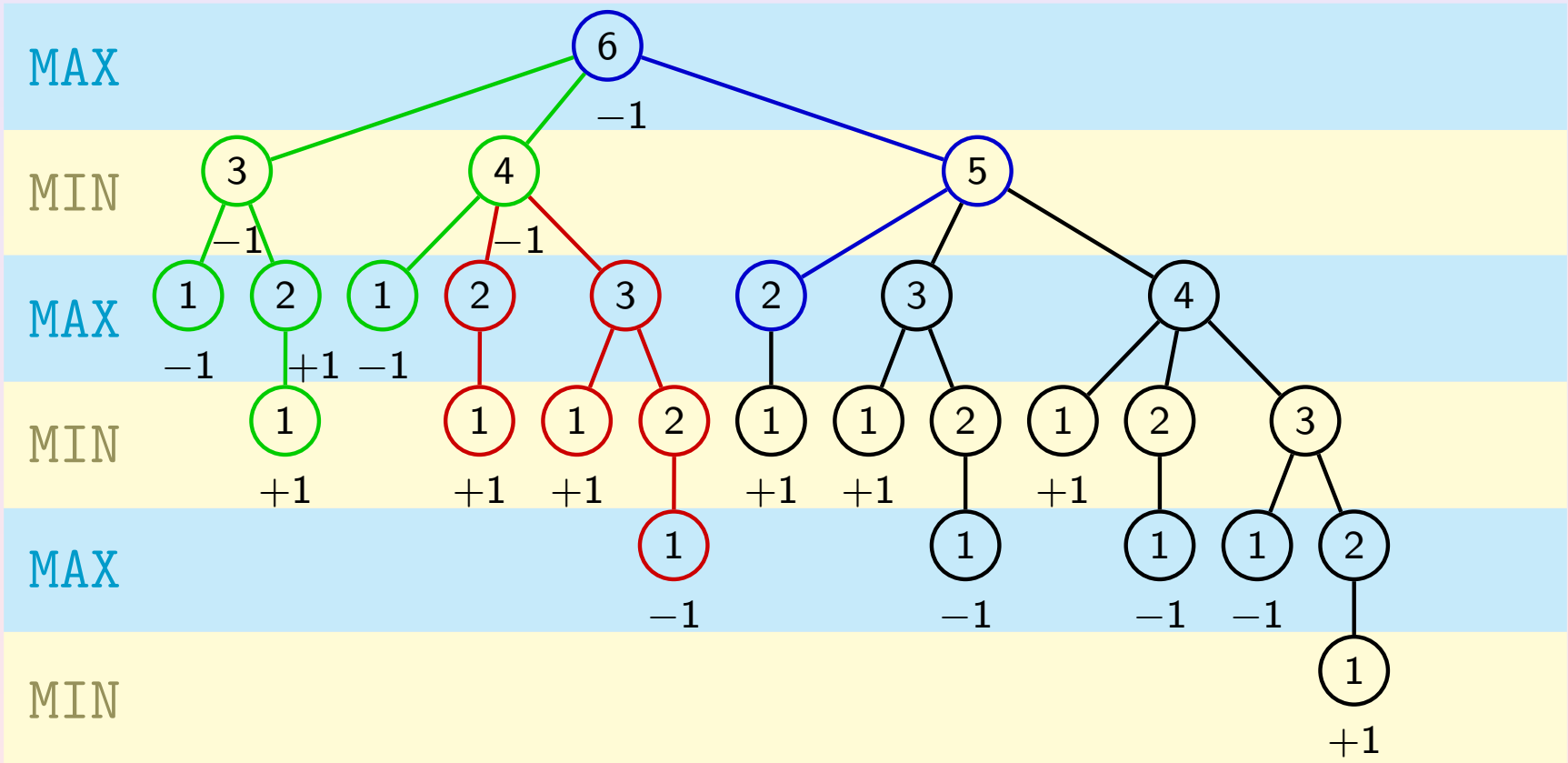
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



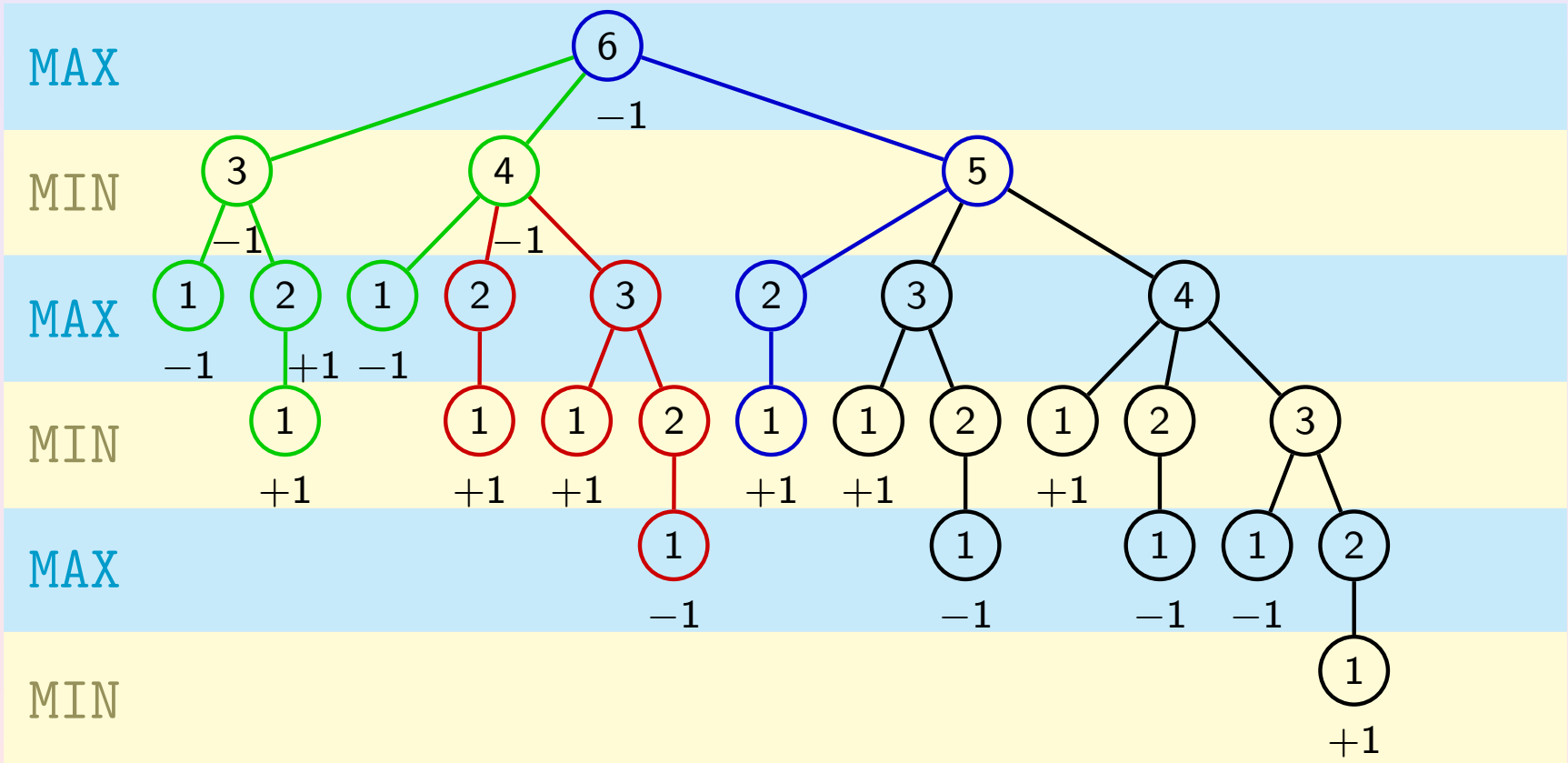
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



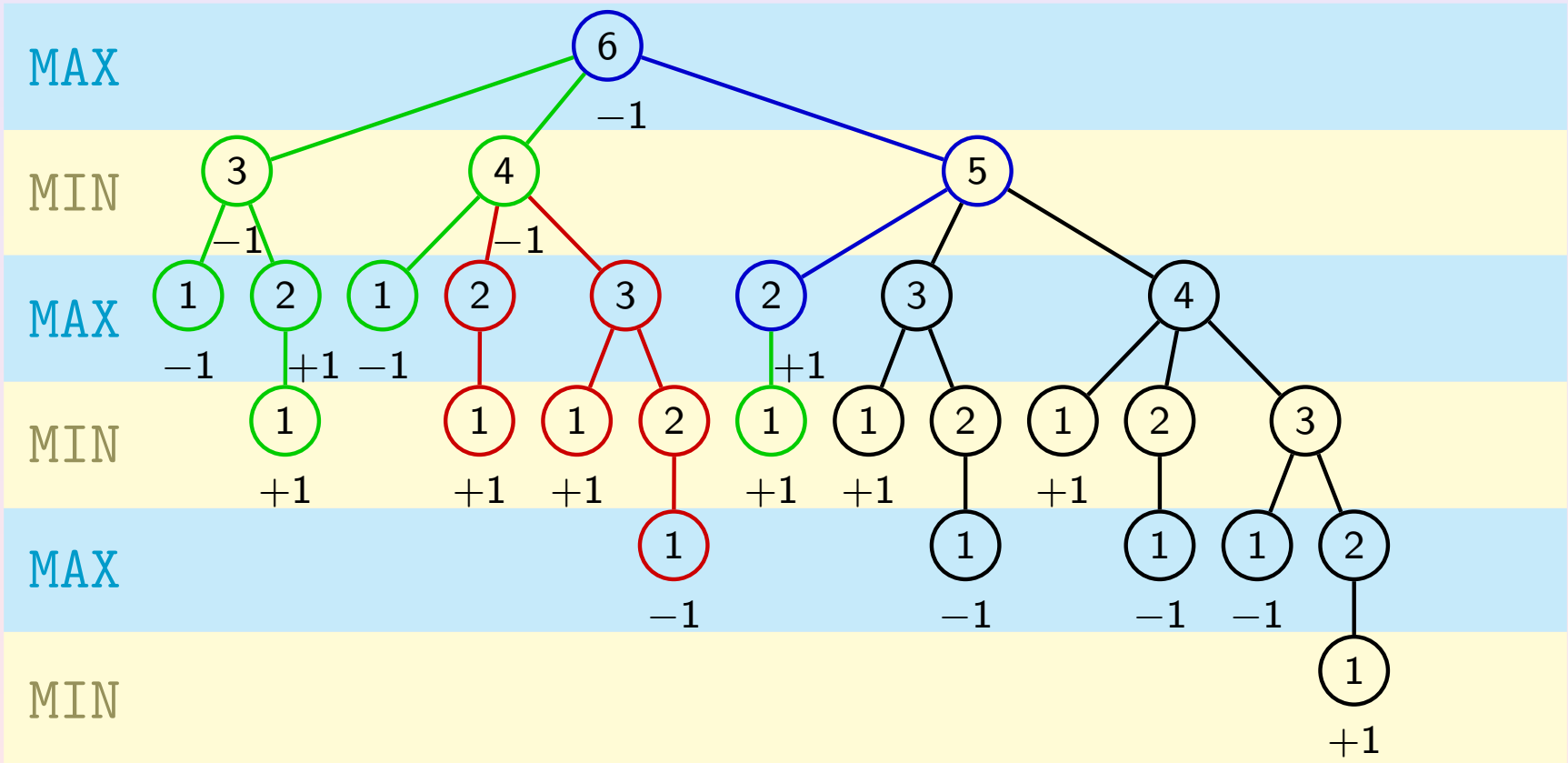
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



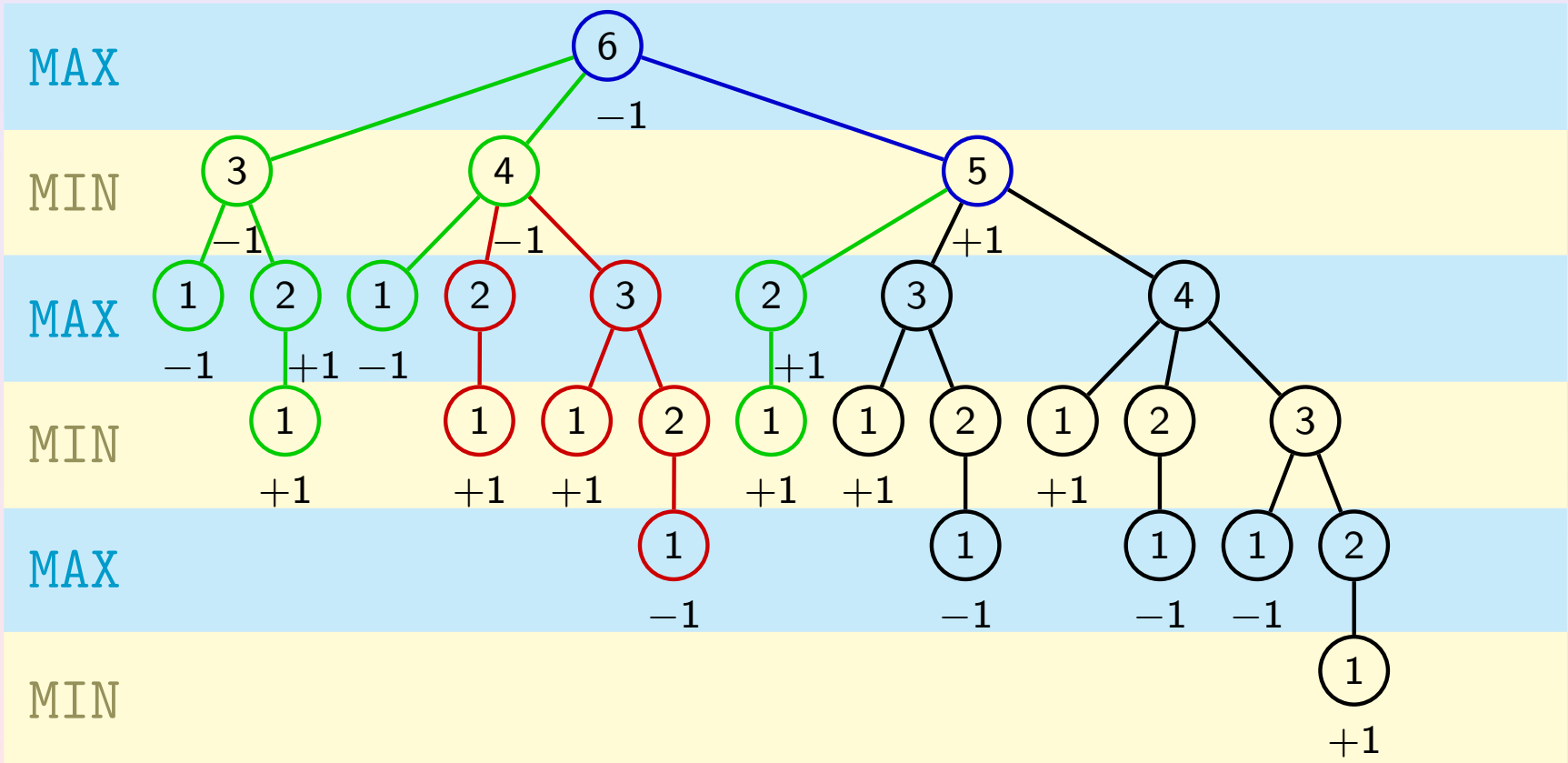
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



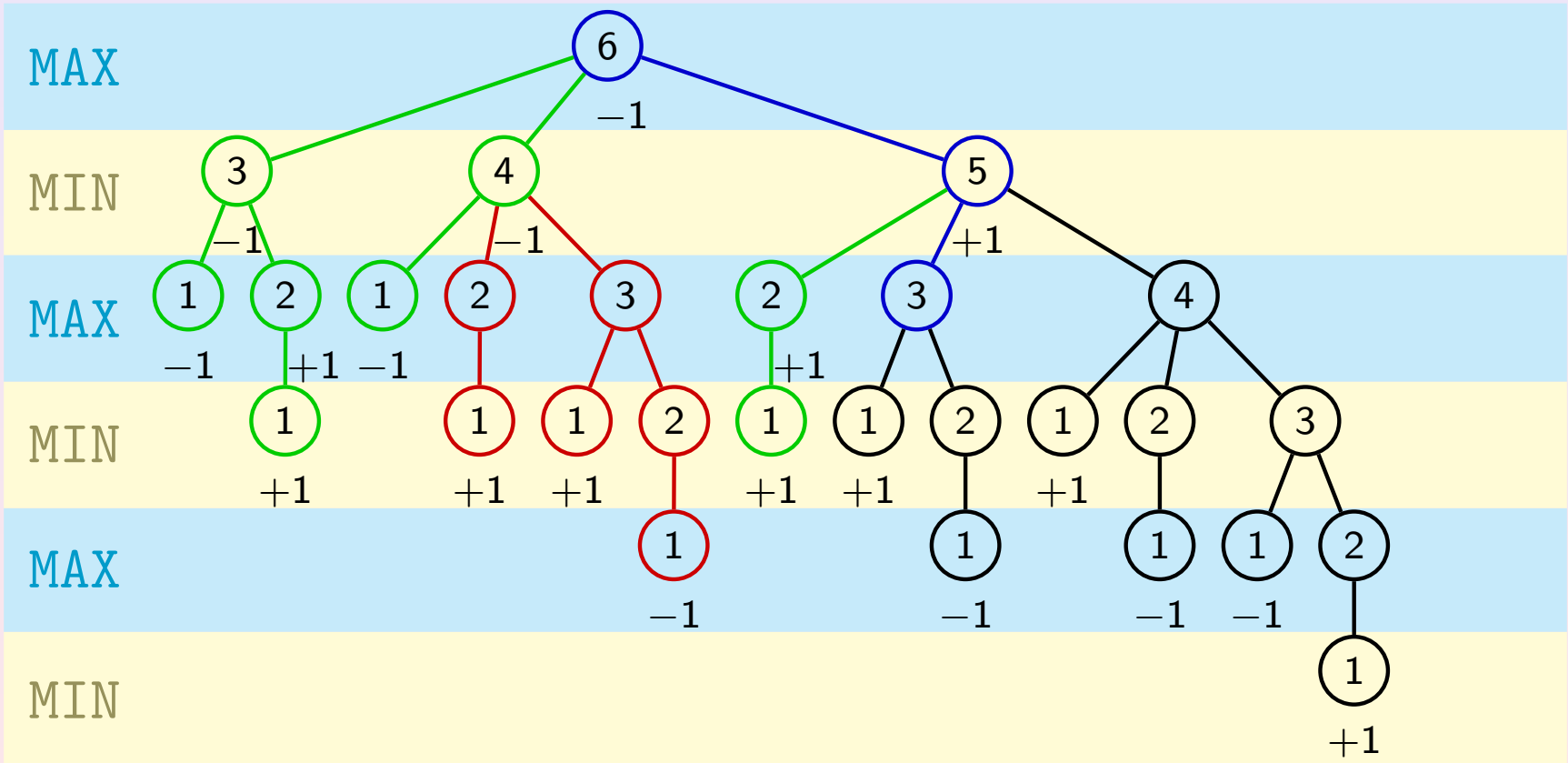
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



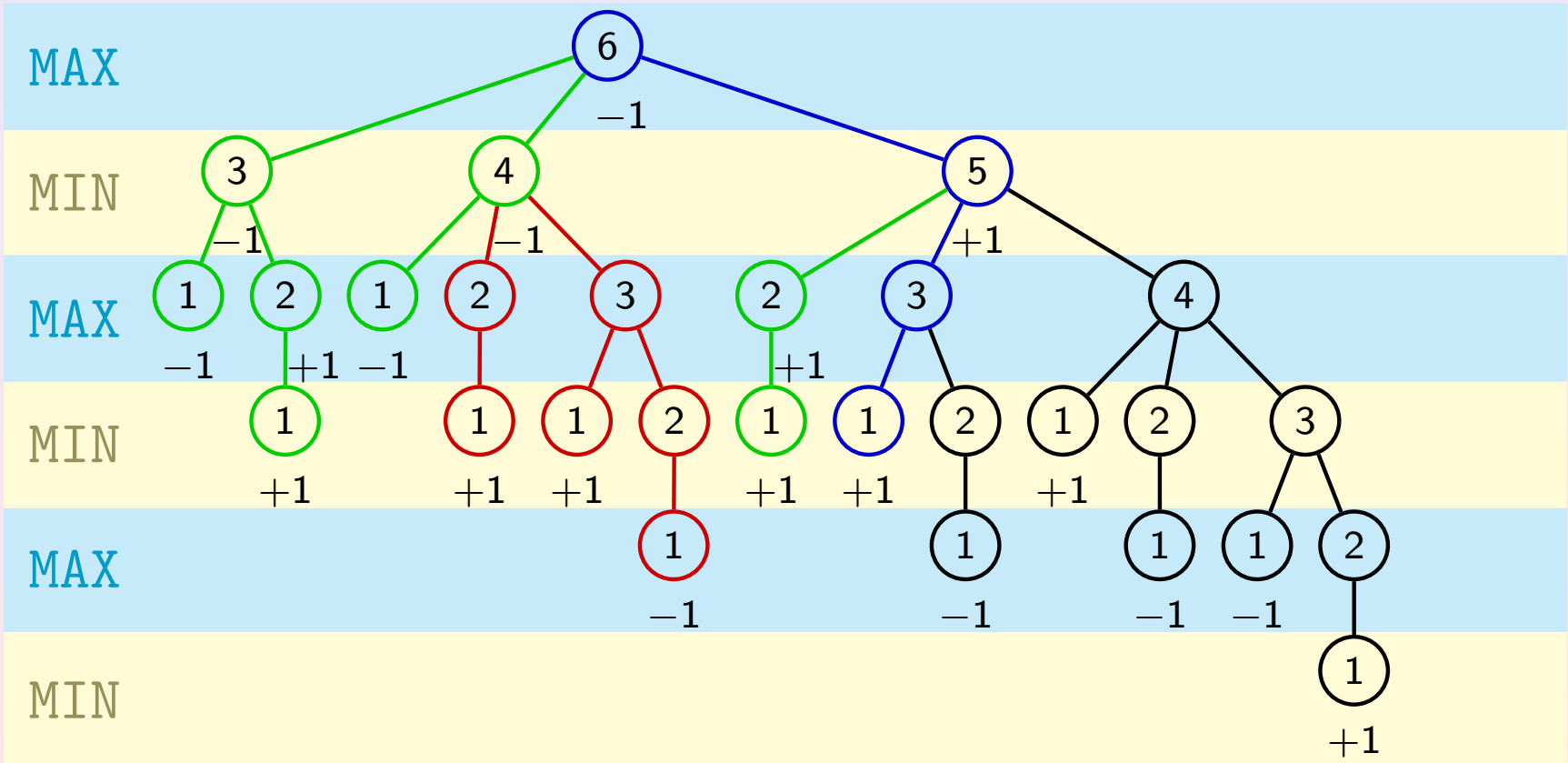
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



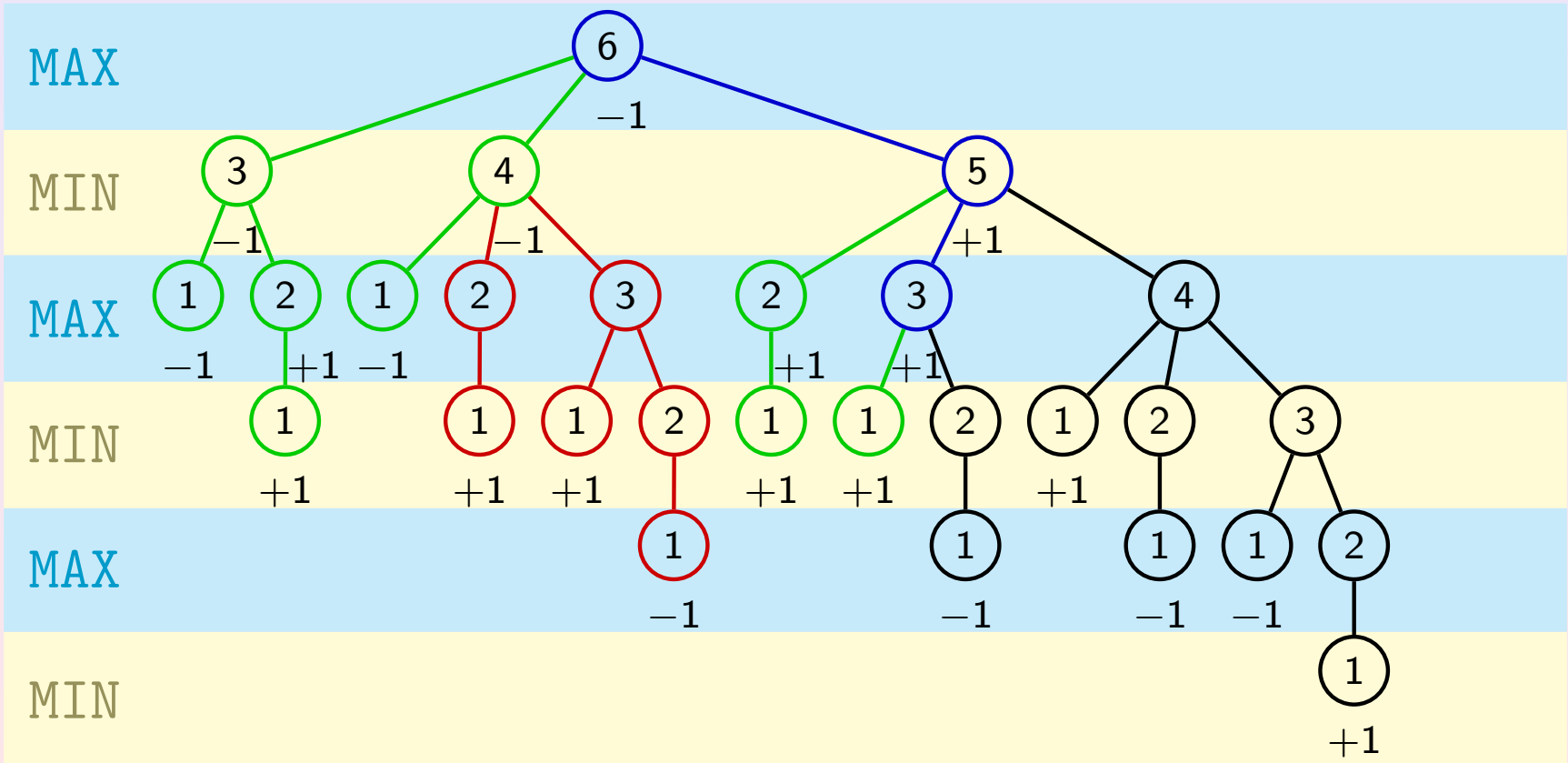
Beta: +1 Alfa: -1

Ávore de Busca: Busca em profundidade



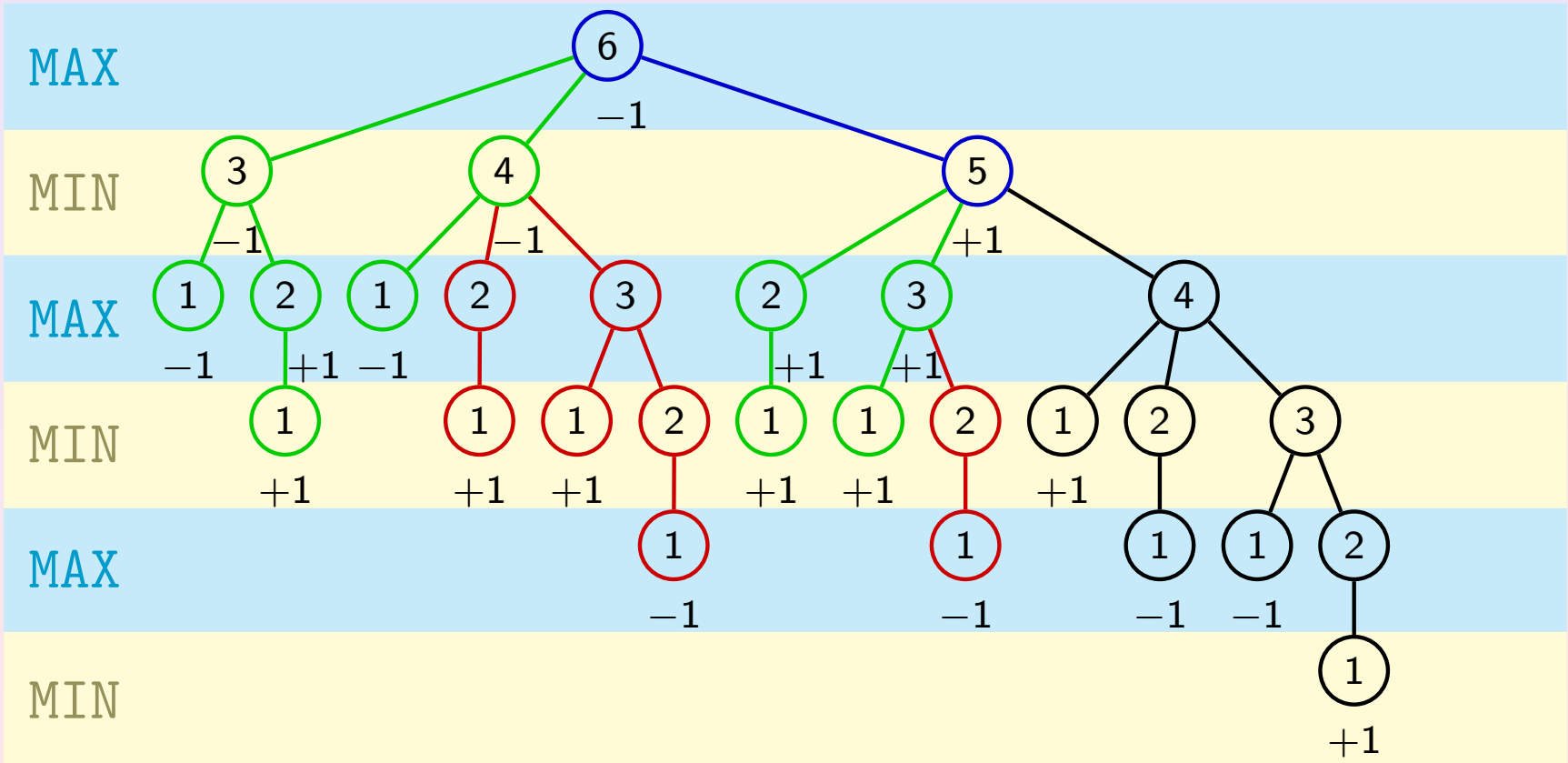
Beta: +1 Alfa: -1

Ávore de Busca: Busca em profundidade



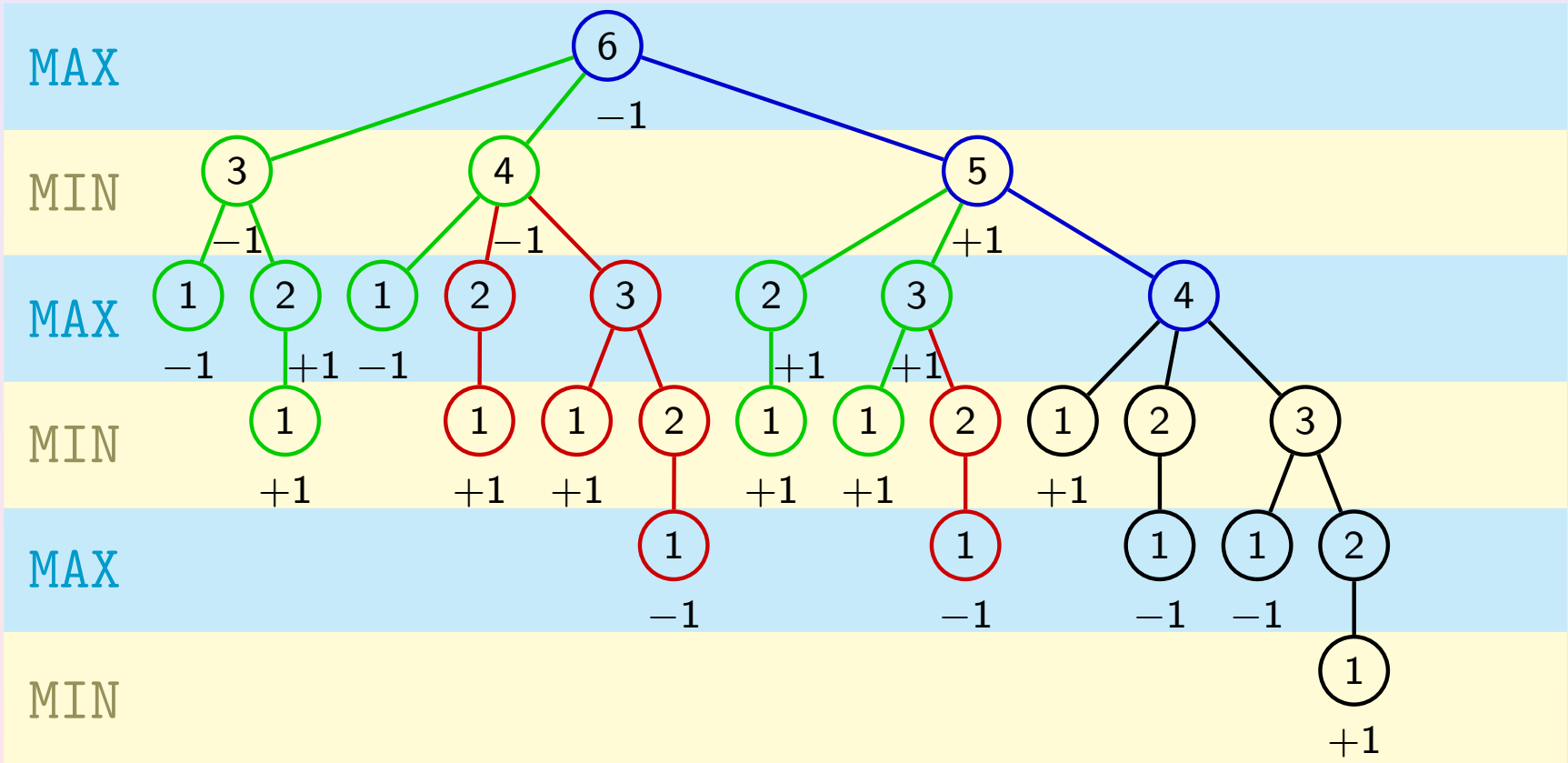
PODA ← Beta: +1 Alfa: -1

Ávore de Busca: Busca em profundidade



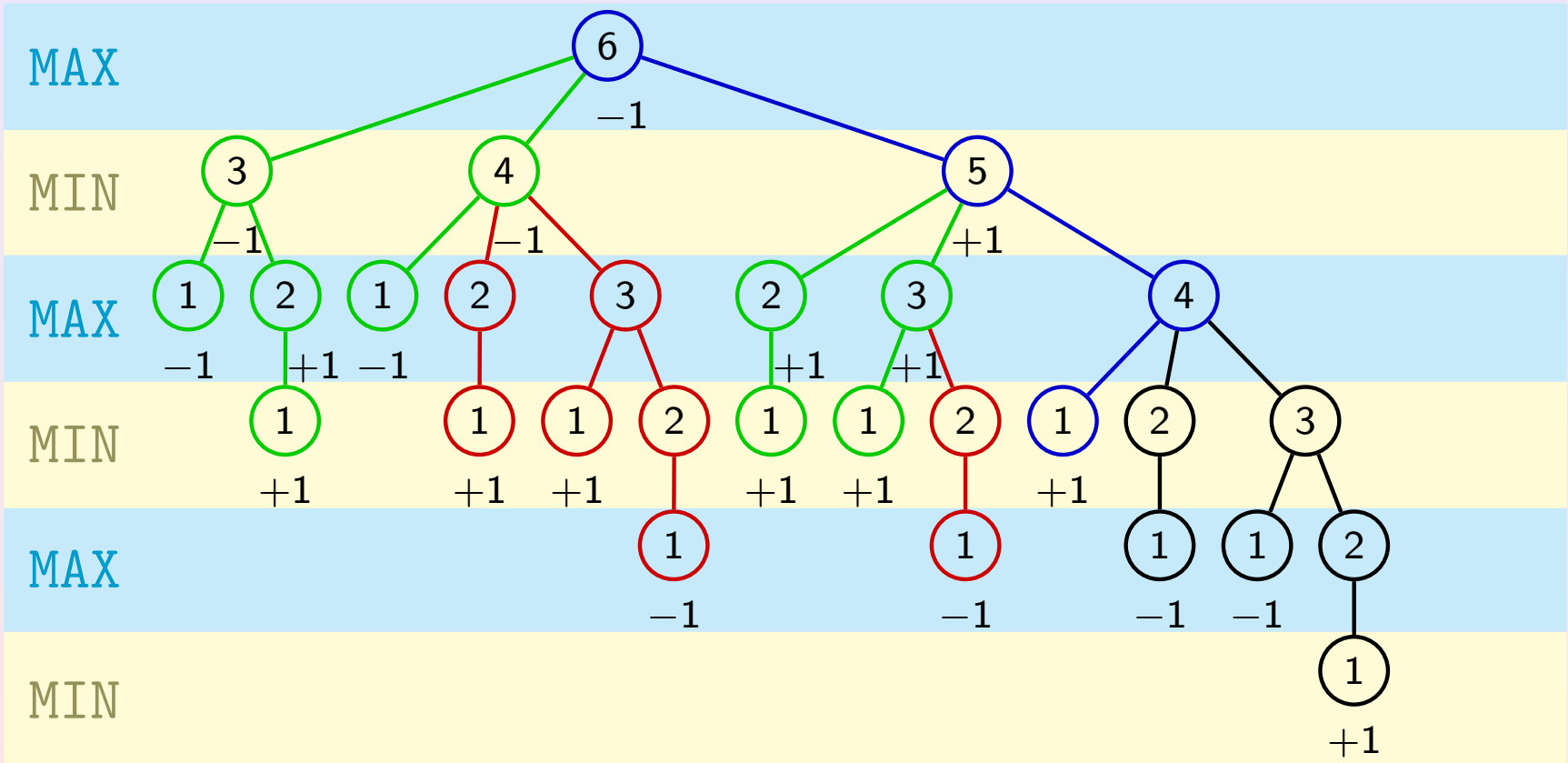
Beta: +1 Alfa: -1

Ávore de Busca: Busca em profundidade



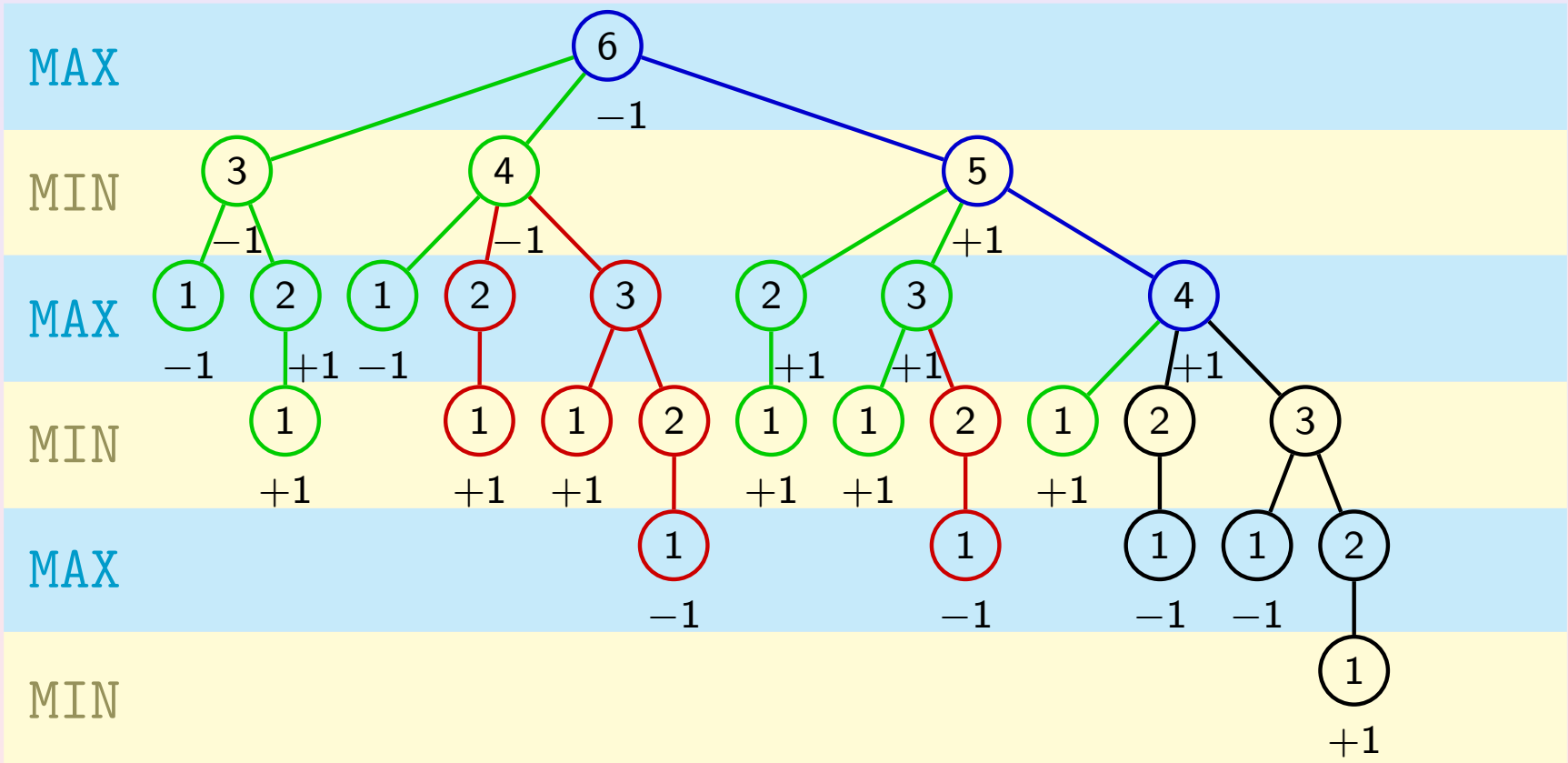
Beta: +1 Alfa: -1

Ávore de Busca: Busca em profundidade



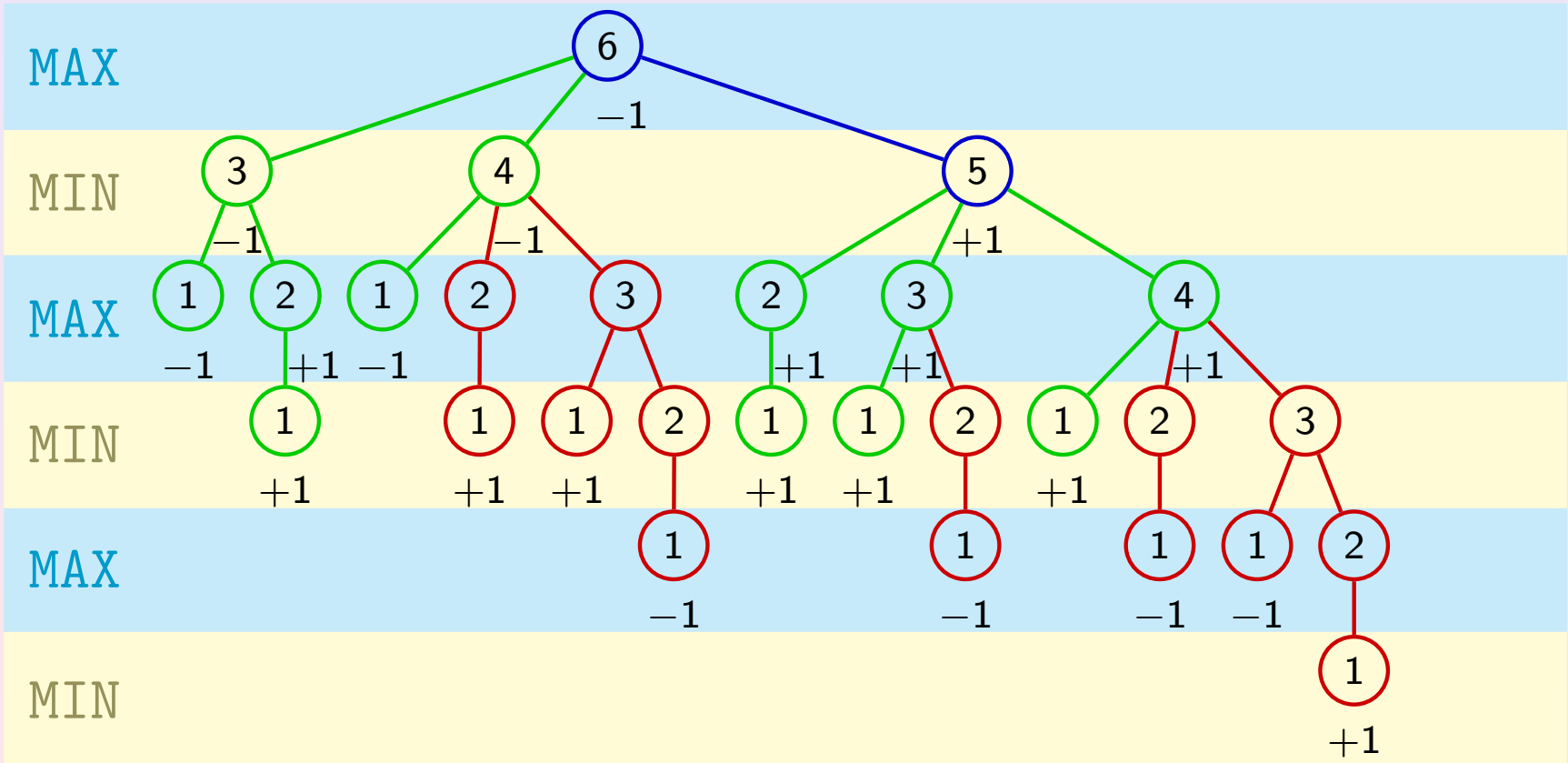
Beta: +1 Alfa: -1

Ávore de Busca: Busca em profundidade



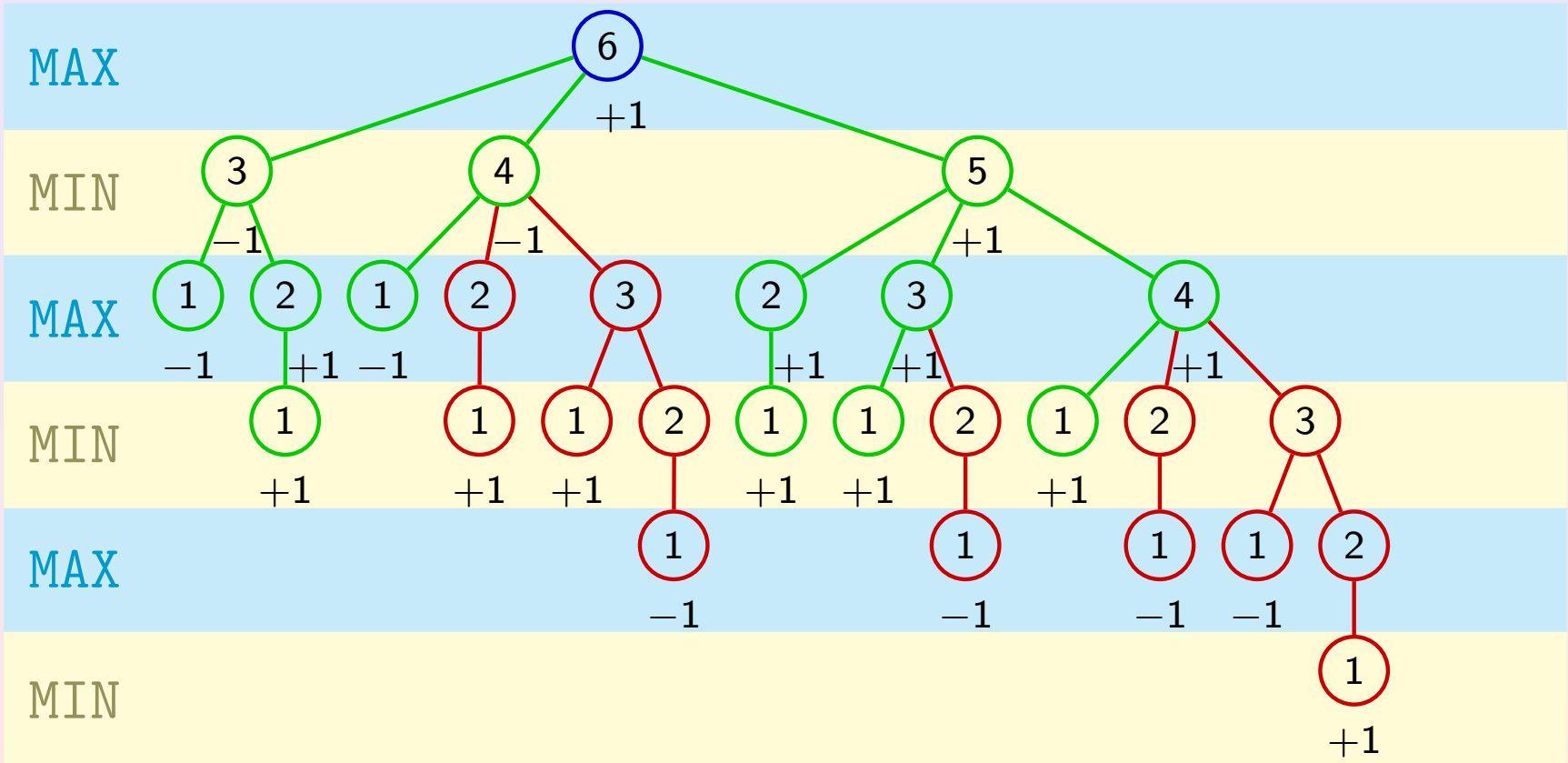
PODA ← Beta: +1 Alfa: -1

Ávore de Busca: Busca em profundidade



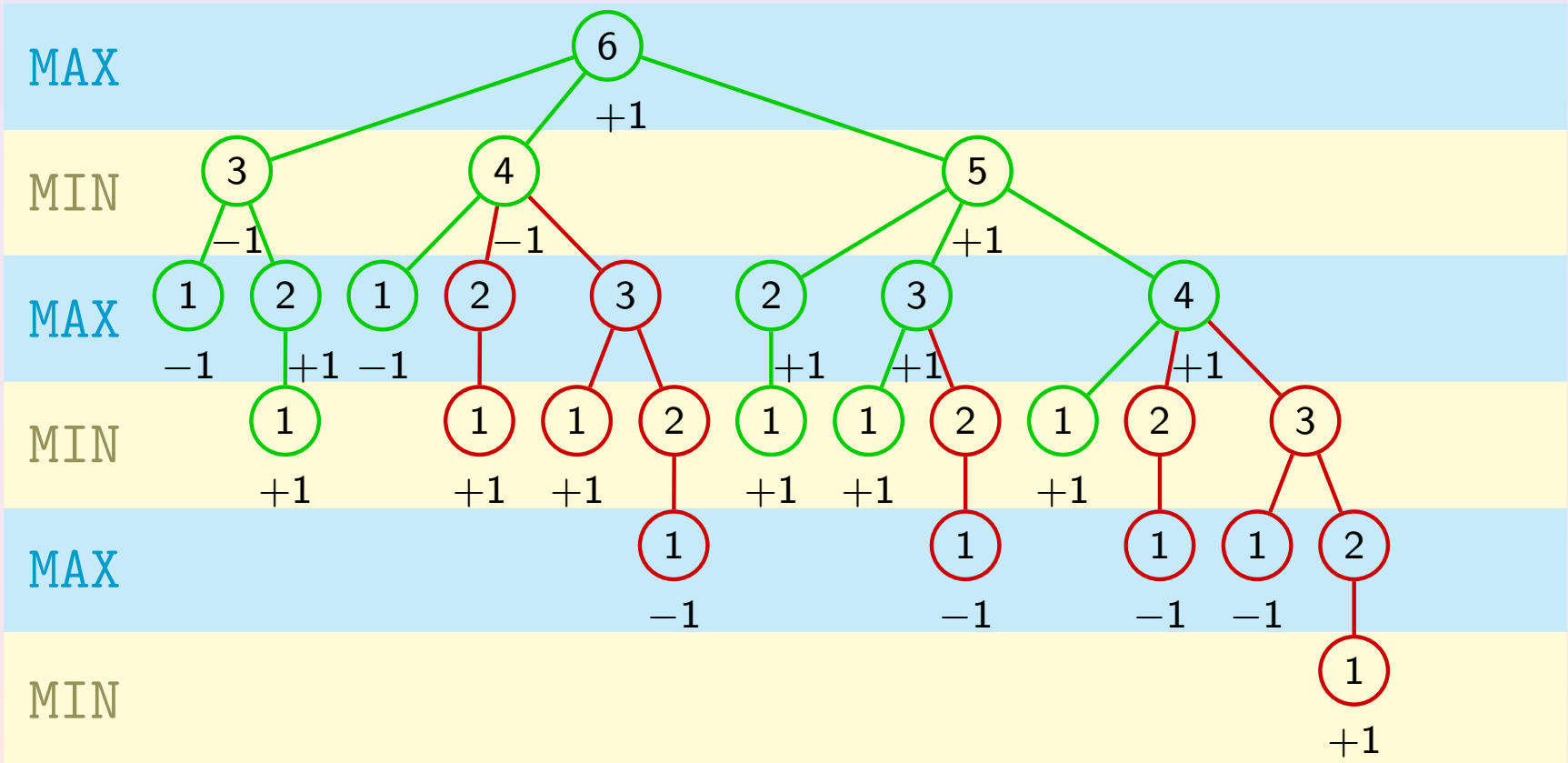
Beta: ? Alfa: -1

Ávore de Busca: Busca em profundidade



Beta: ? Alfa: ?

Ávore de Busca: Busca em profundidade



Beta: ? Alfa: ?

Árvore de Busca: Antecipação de Resultados

- Apesar da poda, muitas vezes ainda existem muitos níveis e nós para computar.
- Uma técnica é "Antecipar Resultados"
- Mas como calcular estes valores? Vamos usar uma "Função Heurística"
  - Em geral, definir uma forma de medir "vantagens"
  - Maior quantidade de peças pode ser uma vantagem (dama).
  - Além da quantidade, o tipo de peça (xadrez).
- Nem sempre o que achamos ser vantagens pode levar a uma estratégia vencedora.
- Este problema é chamado de problema do horizonte.

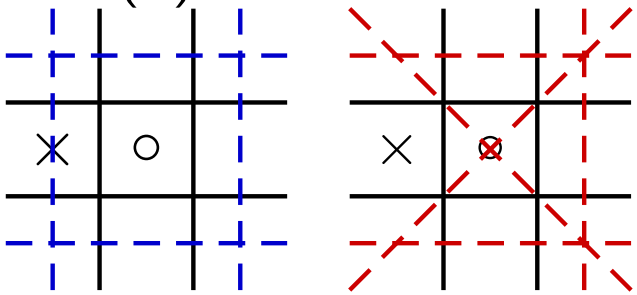
Árvore de Busca: Antecipação de Resultados

- Vamos pensar em uma estratégia para o jogo da velha.
- Uma função heurística para o jogo, pode ser definido por quantidades possíveis de vitórias em um determinado estado.
  - Sendo você o jogador $\times$ e o adversário $\circ$
  - $H(n) = \times(n) - \circ(n)$
  - $H(n)$ é a função heurística.
  - $\times(n)$ é o total de linhas vitoriosas para $\times$
  - $\circ(n)$ é o total de linhas vitoriosas para $\circ$

Árvore de Busca: Antecipação de Resultados

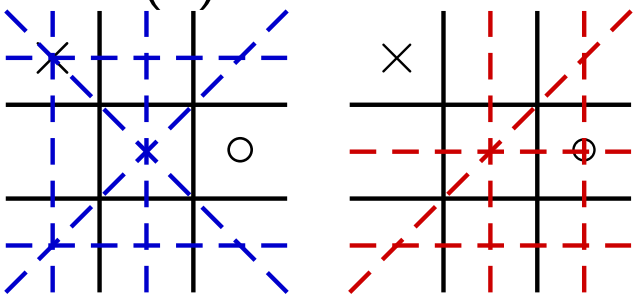
- Cálculo da função da função heurística

$$H(n) = 4 - 5 = -1$$



$$\times(n) = 4 \qquad \circ(n) = 5$$

$$H(n) = 6 - 5 = 1$$



$$\times(n) = 6 \qquad \circ(n) = 5$$

Árvore de Busca: Antecipação de Resultados