

# Bài tập Lý thuyết 4

## Thông tin sinh viên

### Bài 1. Trả lời câu hỏi

- a. Tìm kiếm đối kháng là gì? Nêu sự khác biệt so với các dạng tìm kiếm trước đây.
- b. Có những dạng nào của tìm kiếm đối kháng?
- c. Cây tìm kiếm là gì? Tại sao lại cần xác định cây tìm kiếm?
- d. Có những thành phần nào trong cây tìm kiếm, ý nghĩa của mỗi thành phần?
- e. Thuật toán nào để giải quyết tìm kiếm đối kháng? Mô tả ngắn gọn (các) thuật toán đó.
- f. Thách thức nào đặt ra cho tìm kiếm đối kháng? Hãy liệt kê ít nhất 3 thách thức.
- g. Nêu các giải pháp để giải quyết các thách thức.

### Bài 2. Tìm kiếm đối kháng

- a. Tính toán giá trị Minimax cho tất cả trạng thái của cây
- b. Tia nhánh  $\alpha - \beta$  (Chi tiết các bước)

### Câu 3. Tìm kiếm đối kháng

- a. Tính toán minimax
- b. Tia nhánh  $\alpha - \beta$  (Chi tiết từng bước)

### Câu 4. Thực hiện lại bài 2 và 3 với Min & Max ngược lại

Hình 2.

Hình 3.

### Bài 5. Tia nhánh các cây tìm kiếm

Hình a.

Hình b.

## Thông tin sinh viên

- MSSV: 22850034
- Họ và tên: Cao Hoài Việt
- Email: viet.ch2612@gmail.com

## Bài 1. Trả lời câu hỏi

- a. Tìm kiếm đối kháng là gì? Nêu sự khác biệt so với các dạng tìm kiếm trước đây.**

Tìm kiếm đối kháng là dạng tìm kiếm dùng trong các trò chơi đối kháng như Cờ vua, tic-tac-toe, cờ vây. Các dạng tìm kiếm trước đây ( $A^*$ , UCS, Greedy) là các dạng tìm kiếm đường đi, còn tìm kiếm đối kháng là tìm kiếm nước đi “tốt nhất”. Nước đi của người chơi sẽ phụ thuộc vào nước đi của đối thủ và ngược lại.

### b. Có những dạng nào của tìm kiếm đối kháng?

Theo em hiểu thì chỉ có một loại tìm kiếm đối kháng trên cây tìm kiếm, bao gồm tất cả các nước đi có thể có của cả 2 đối thủ.

### c. Cây tìm kiếm là gì? Tại sao lại cần xác định cây tìm kiếm?

Cây tìm kiếm dùng để biểu diễn khả năng của các nước đi trong các trò chơi đối kháng. Ví dụ như trò chơi Tic-tac-toe (Ca rô) thì gốc (*root*) của cây sẽ là trạng thái đầu, nhánh tiếp theo gồm 9 nhánh biểu hiện  $n = 9$  khả năng người chơi có thể đi trên bàn ca-rô, mỗi nhánh này sẽ tiếp tục có thêm  $n - 1$  nhánh thể hiện 8 khả năng của nước đi tiếp theo.

Cần xác định cây tìm kiếm để có thể áp dụng các giải thuật như minimax hay tỉa nhánh  $\alpha - \beta$  để tìm được nước đi tối ưu nhất.

### d. Có những thành phần nào trong cây tìm kiếm, ý nghĩa của mỗi thành phần?

Cây tìm kiếm gồm các nút và giá trị lợi ích nào đó. Giá trị tại các nút trong là các giá trị nhận được dựa trên giải thuật Minimax (Min hay max của các nút con). Gốc (root) sẽ tượng trưng cho người chơi đầu tiên (MAX) và người chơi tiếp theo sẽ là (MIN).

### e. Thuật toán nào để giải quyết tìm kiếm đối kháng? Mô tả ngắn gọn (các) thuật toán đó.

- Thuật toán Minimax: Hai đối thủ MIN và MAX, MAX sẽ luôn tìm cách tối đa ưu thế của mình và MIN tìm mọi cách để đưa MAX vào thế khó nhất. Thuật toán sẽ gán giá trị cho các nút, truyền ngược các giá trị từ các nút lá về gốc theo quy tắc.
  - Nếu ở đỉnh MAX, gán giá trị cho đỉnh này bằng giá trị lớn nhất trong các giá trị của các giá trị các con của nó.
  - Nếu ở đỉnh MIN thì ngược lại, gán bằng giá trị nhỏ nhất.

- b. Cắt tỉa  $\alpha - \beta$ : Đây là một phiên bản của thuật toán MINIMAX giúp giảm số nút được đánh giá bởi thuật toán bằng cách loại bỏ các nhánh ít có khả năng dẫn đến kết quả thuận lợi. Nó hoạt động bằng cách duy trì hai giá trị  $\alpha$  và  $\beta$ . Thuật toán bắt đầu từ gốc rồi DFS với  $\alpha = -\infty$ ,  $\beta = \infty$ , tới mỗi nút Min sẽ cập nhật  $\beta$ , max sẽ cập nhật  $\alpha$  và cập nhật ngược lại nút cha. Nếu nút nào có giá trị  $\alpha > \beta$  là không hợp lệ, sẽ không cần phải duyệt tiếp các nhánh khác của nút này nữa.
- c. Thuật toán Monte Carlo Tree Search (MCTS) là phương pháp tìm kiếm ngẫu nhiên trên cây tìm kiếm. Khác với cắt tỉa  $\alpha - \beta$  thì MCTS sẽ tập trung vào việc tạo ra các nước đi ngẫu nhiên và đánh giá chúng để tìm ra nước đi tốt nhất. MCTS sẽ tốn thời gian tính toán nhiều hơn so với Minimax hay cắt tỉa  $\alpha - \beta$ .
- d. Deep Reinforcement Learning: Là một phương pháp sử dụng neural networks để học các cách tối ưu cho một trò chơi cụ thể. Thuật toán này sẽ sử dụng cách tự chơi với chính nó để điều chỉnh cách chơi dựa trên kết quả của những ván chơi đó.

#### **f. Thách thức nào đặt ra cho tìm kiếm đối kháng? Hãy liệt kê ít nhất 3 thách thức.**

- Không đoán trước được phản ứng của đối thủ. Nếu đối thủ là con người thì không phải lúc nào đối thủ cũng chọn nước đi tối ưu nhất còn thuật toán tìm kiếm đối kháng thì giả định là đối thủ luôn chọn được nước đi tối ưu.
- Giới hạn về thời gian đối với những trò chơi yêu cầu về thời gian (Cờ vua tính giờ) thì sẽ phải cân bằng giữa thời gian và chất lượng nước đi.
- Khó có thể tìm được nước đi tối ưu nhất trong tất cả các nước đi với những trò chơi có miền giá trị cực lớn. Ví dụ đối với trò chơi Cờ vua, hệ số phân nhánh  $b \approx 35$  và hệ số độ sâu của cây biểu diễn  $b \approx 100$  nên chi phí quá cao, không thể tìm được chính xác tới nước đi tối ưu nhất.

#### **g. Nêu các giải pháp để giải quyết các thách thức.**

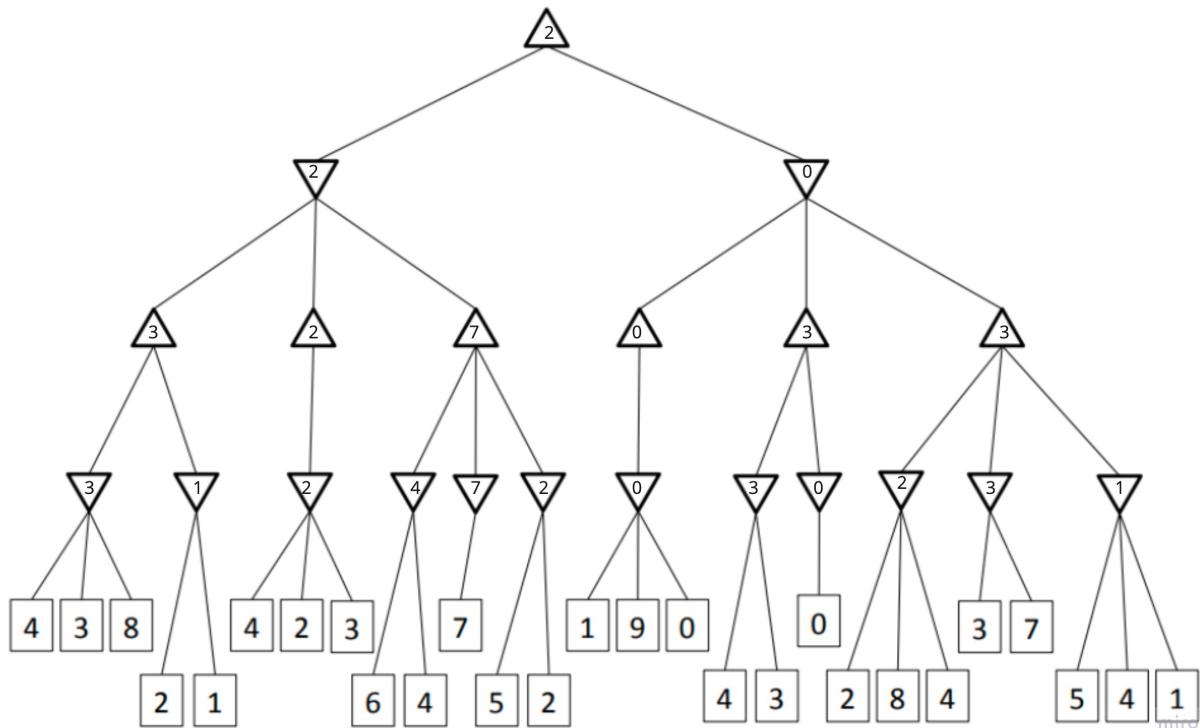
- Sử dụng siêu máy tính để có thể tính toán được nhiều bước đi với độ sâu lớn.
- Dừng ở độ sâu hợp lý. Ví dụ với trò chơi cờ vua, chúng ta không thể tìm kiếm tới độ sâu 100 được nên tuỳ vào khả năng tính toán của máy tính, chúng ta có thể tìm kiếm tới độ sâu phù hợp. Ví dụ, phần mềm Stockfish 15.1 [https://en.wikipedia.org/wiki/Stockfish\\_\(chess\)](https://en.wikipedia.org/wiki/Stockfish_(chess)) cho môn cờ vua có Elo > 3500+ và

thắng 13 lần các giải đấu cờ vua cho phần mềm máy tính thường tìm kiếm tới độ sâu **20 → 24**.

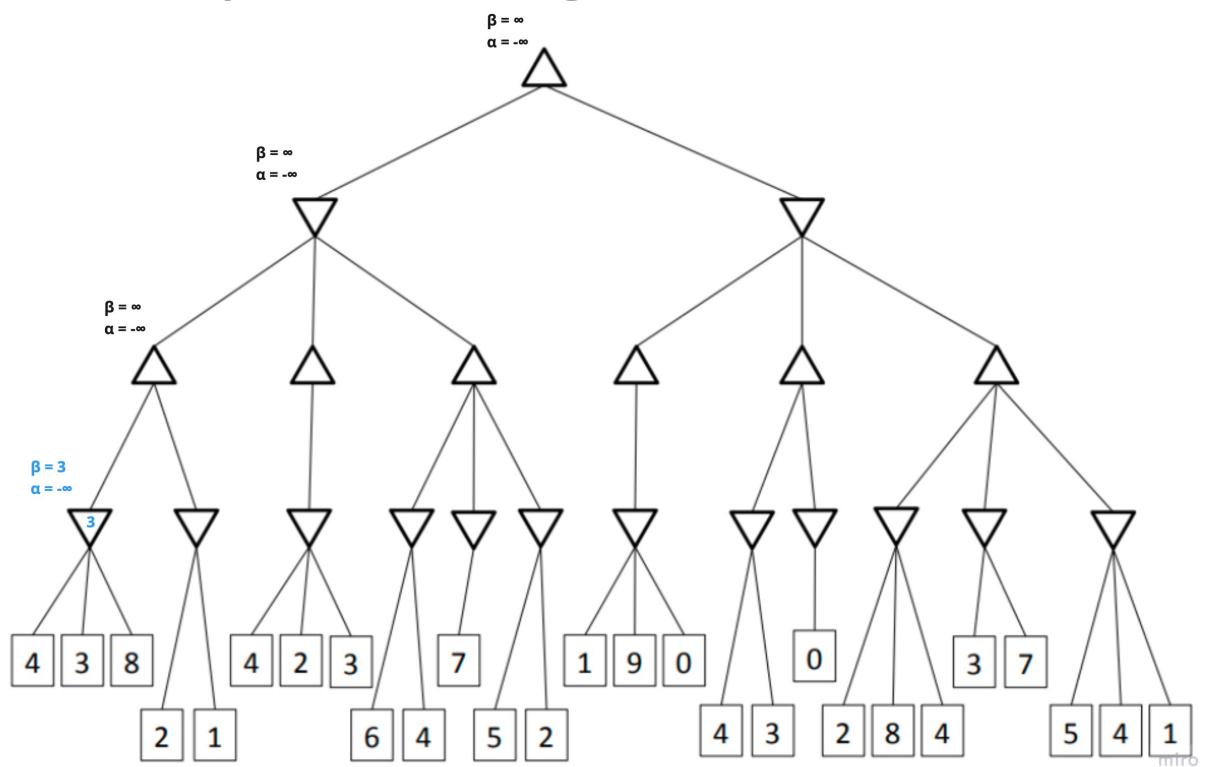
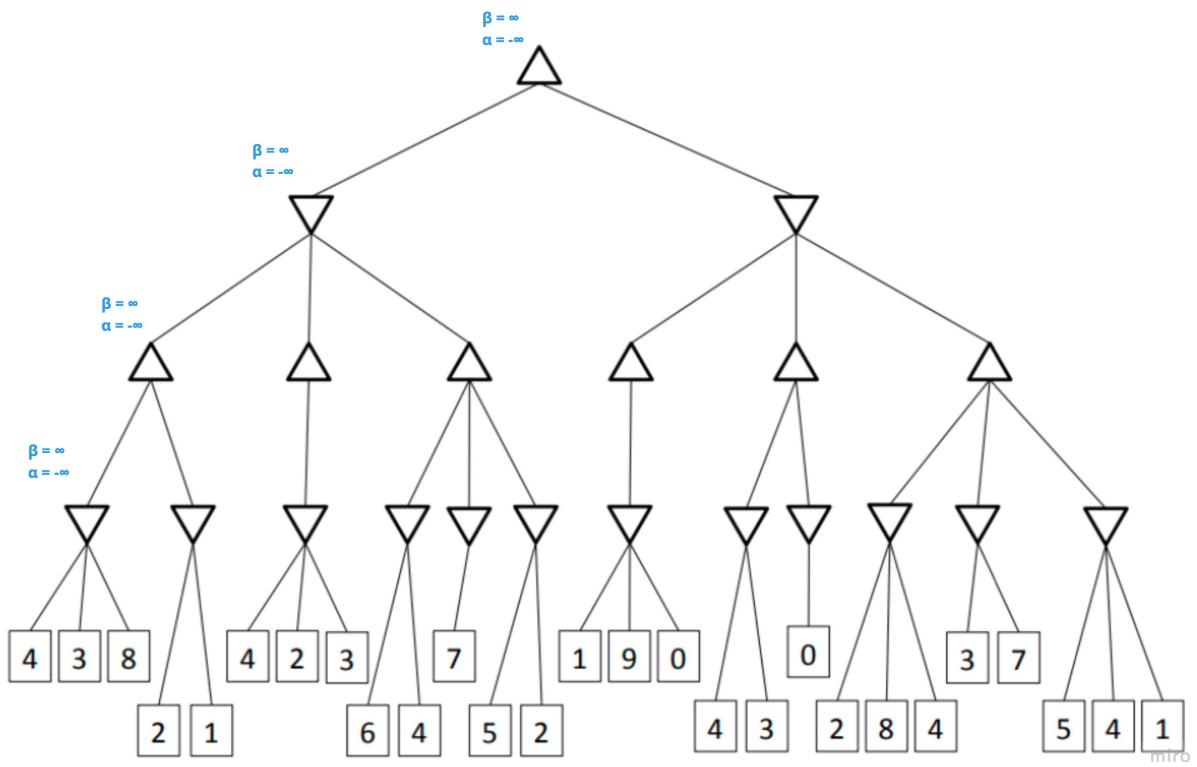
- Sử dụng trí tuệ nhân tạo (AI): Khác với phần mềm chạy trên một máy tính, sử dụng CPU để tính toán các nước đi có thể thì AI sẽ học tập từ các nước đi để tìm ra cách đi tốt nhất. Hiệu quả của từng bước đi sẽ dựa trên tập dữ liệu mẫu, thời gian train model, . . . Ví dụ: AI **AlphaZero** cho trò chơi Cờ vua được tạo bởi công ty DeepMind. Khi AlphaZero được train trong vòng 24 giờ thì đã đánh bại được Nhà vô địch máy tính cờ vua Stockfish.

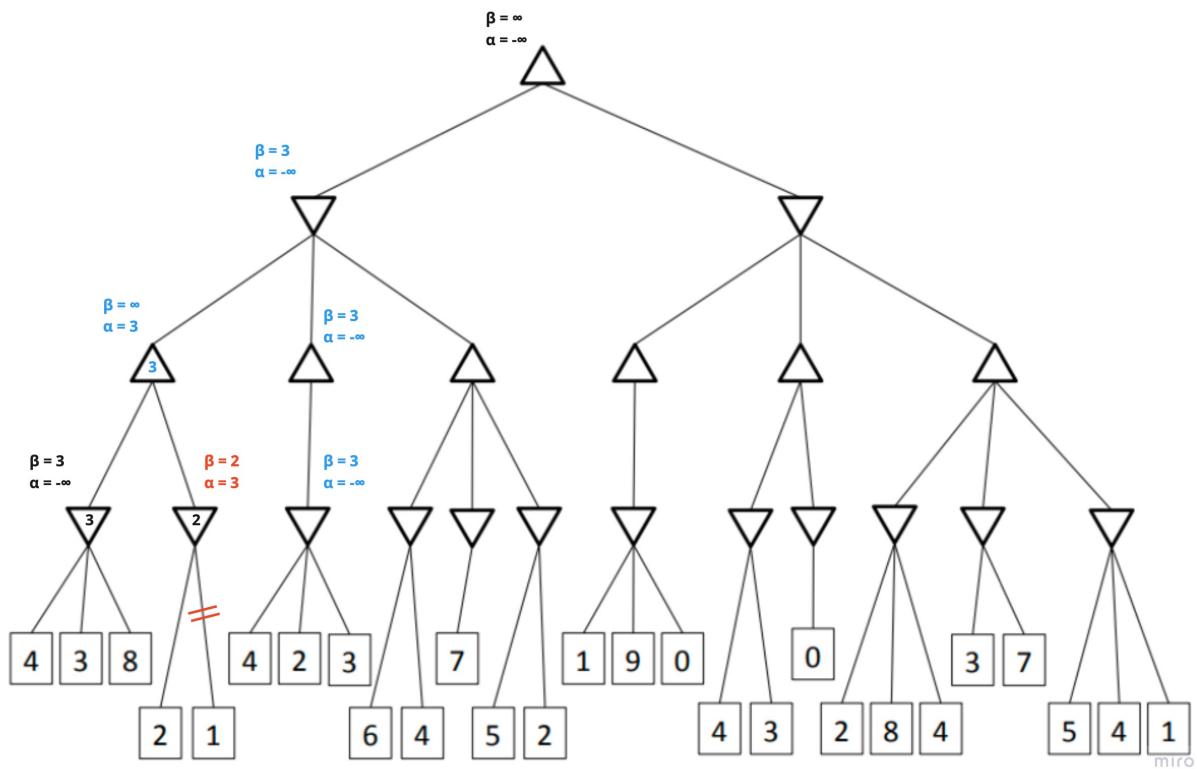
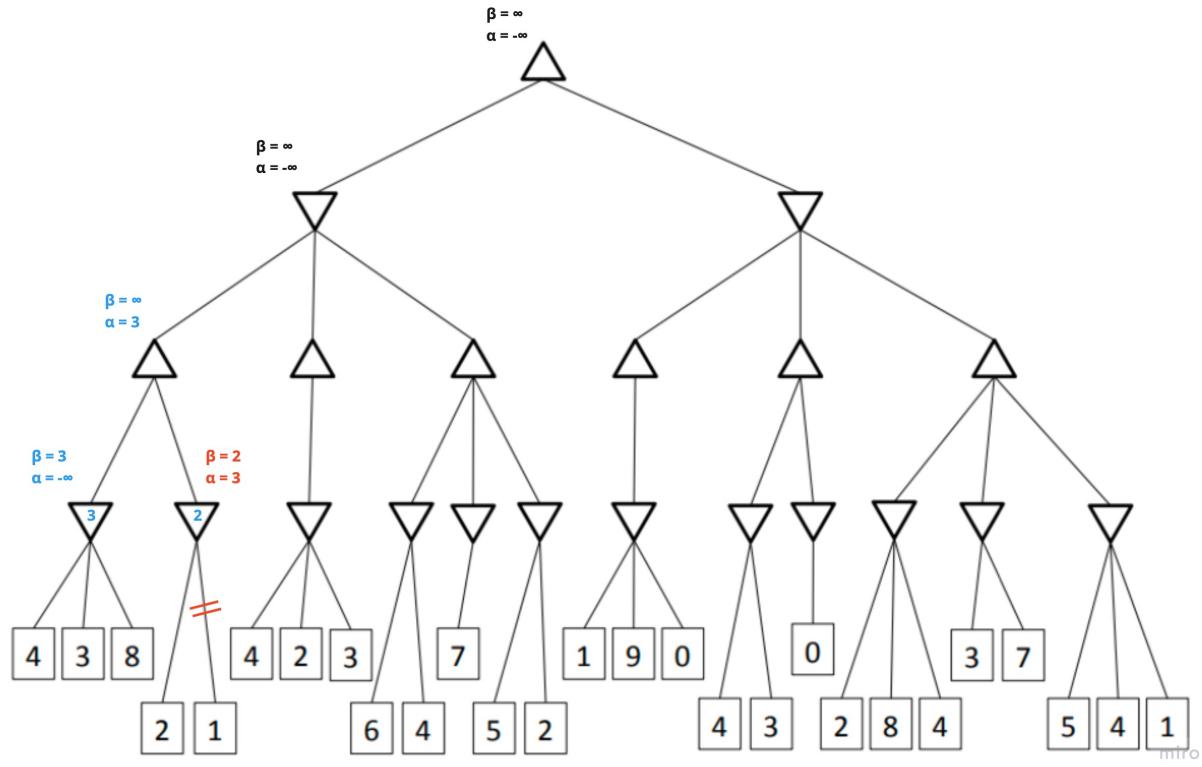
## Bài 2. Tìm kiếm đối kháng

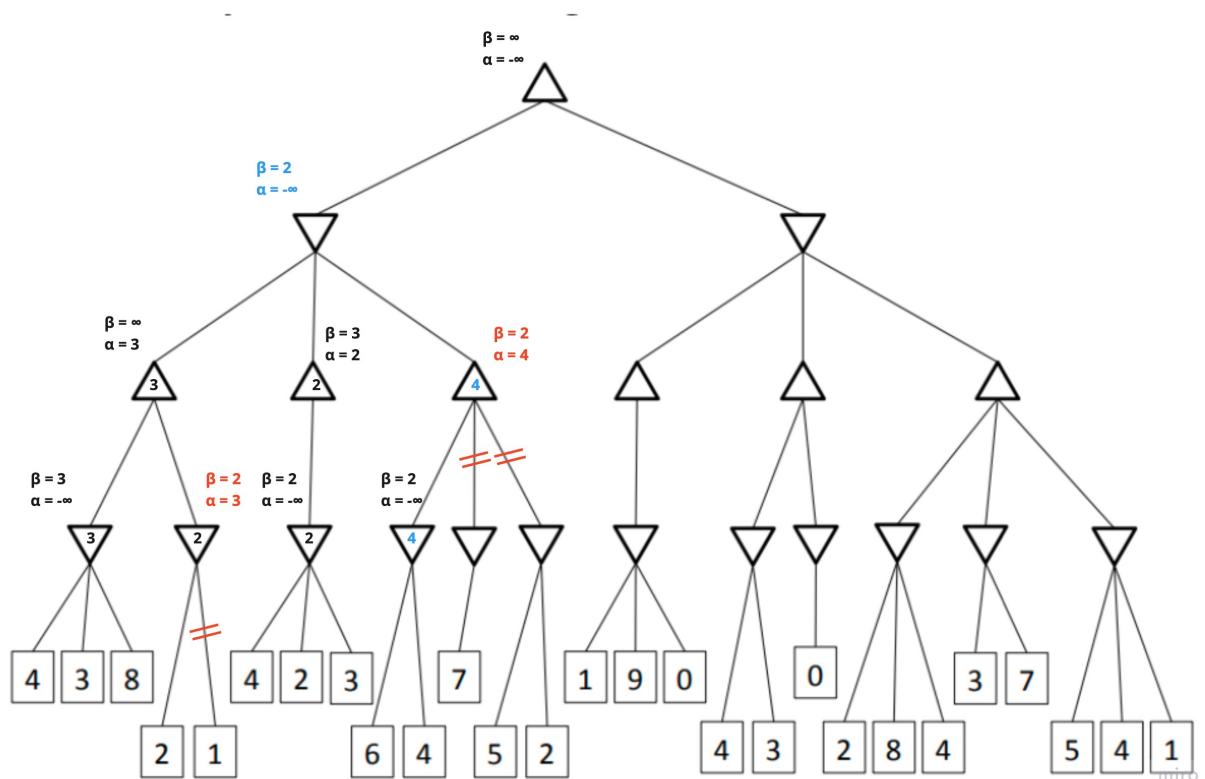
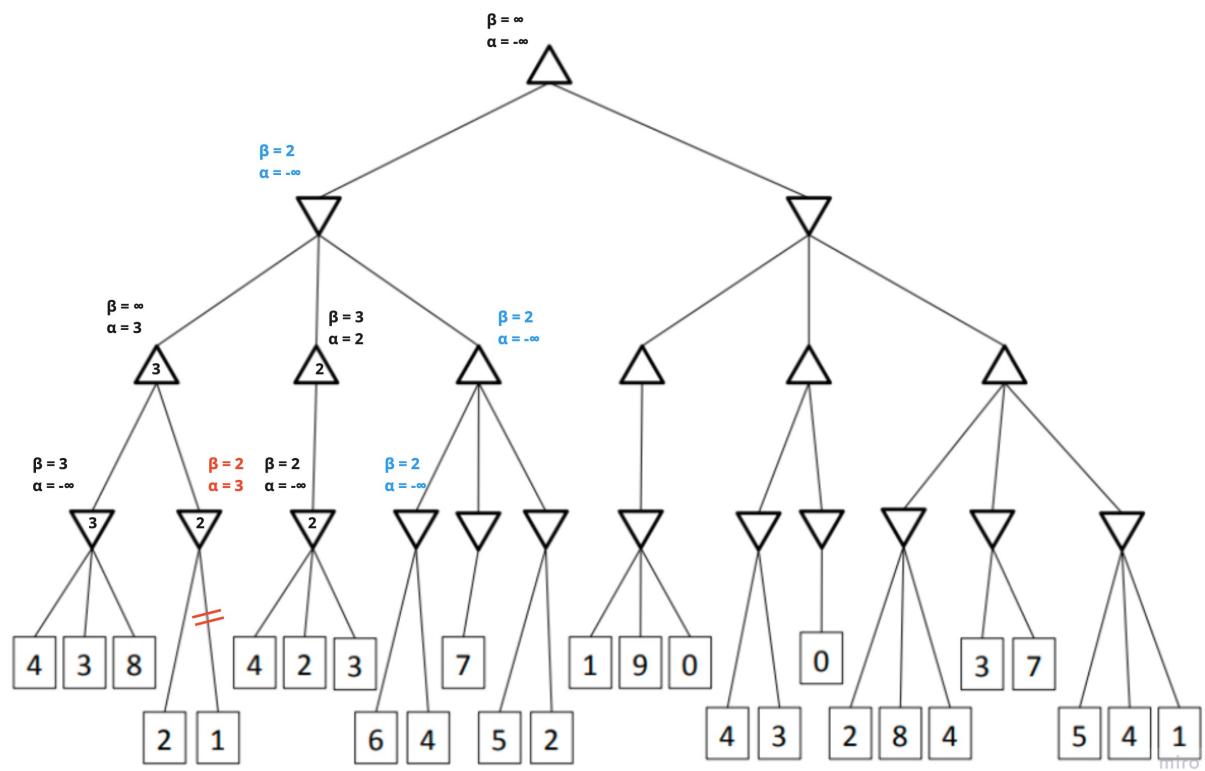
### a. Tính toán giá trị Minimax cho tất cả trạng thái của cây

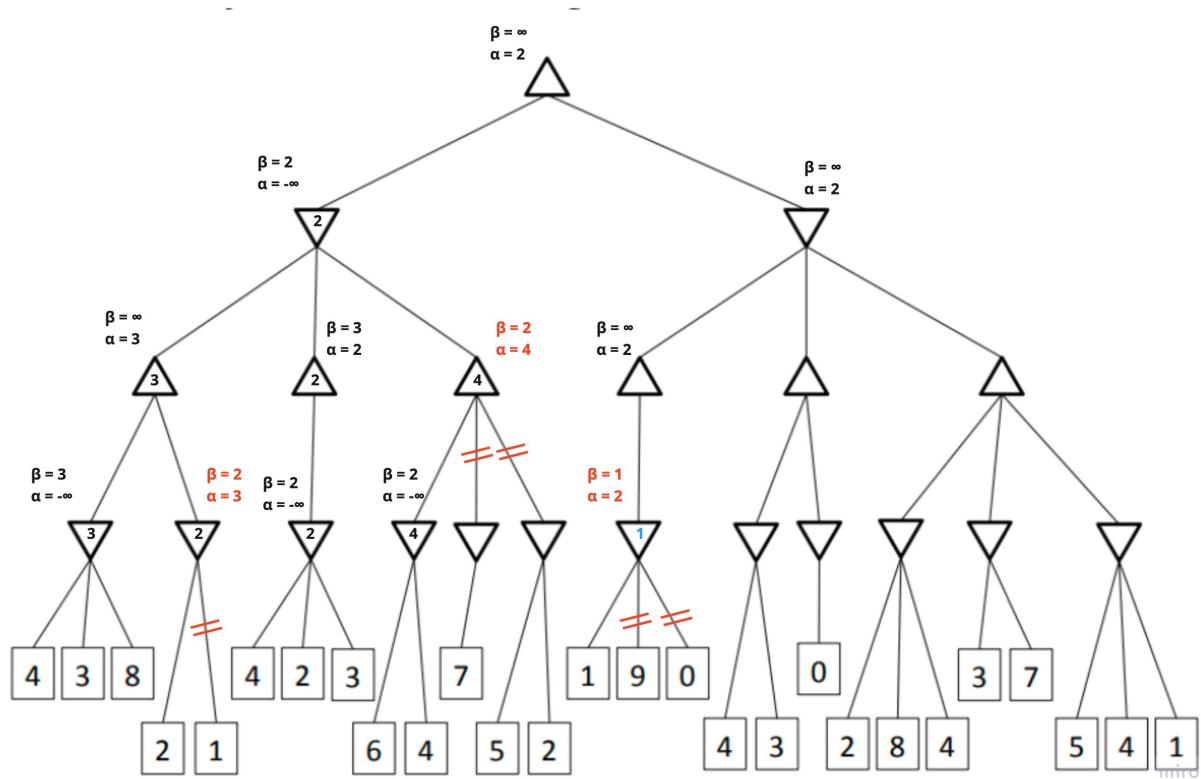
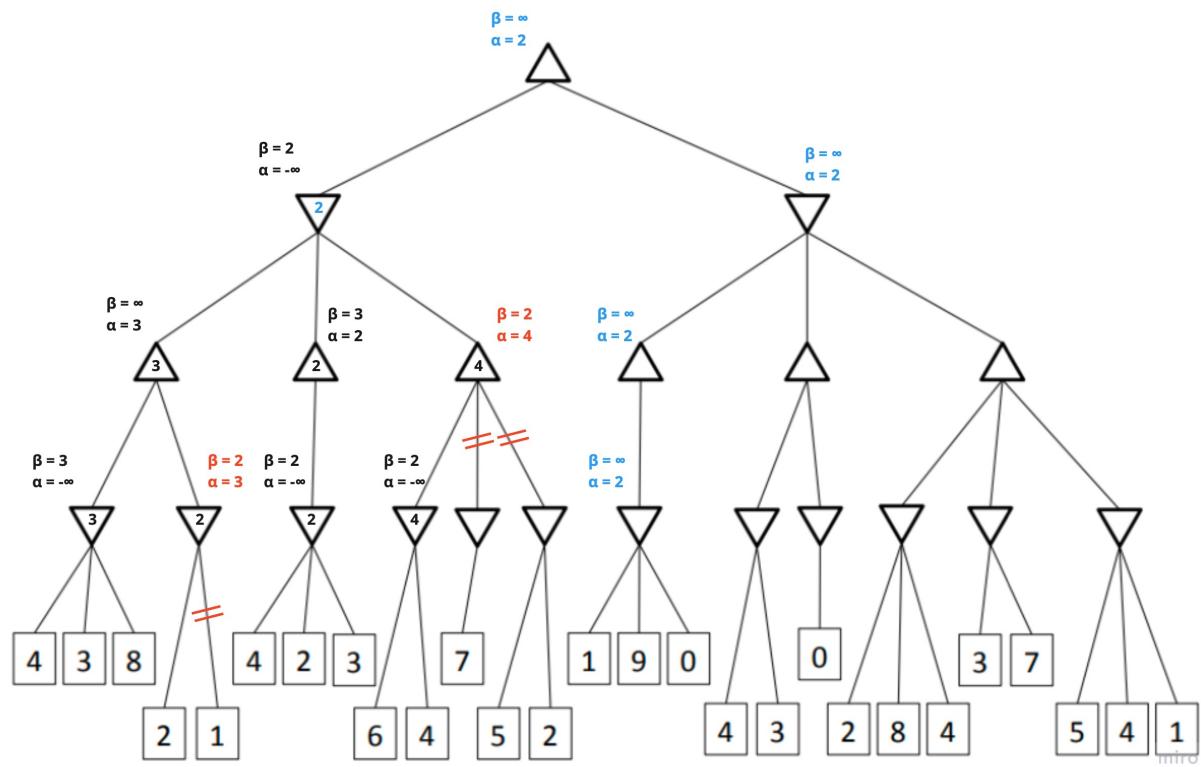


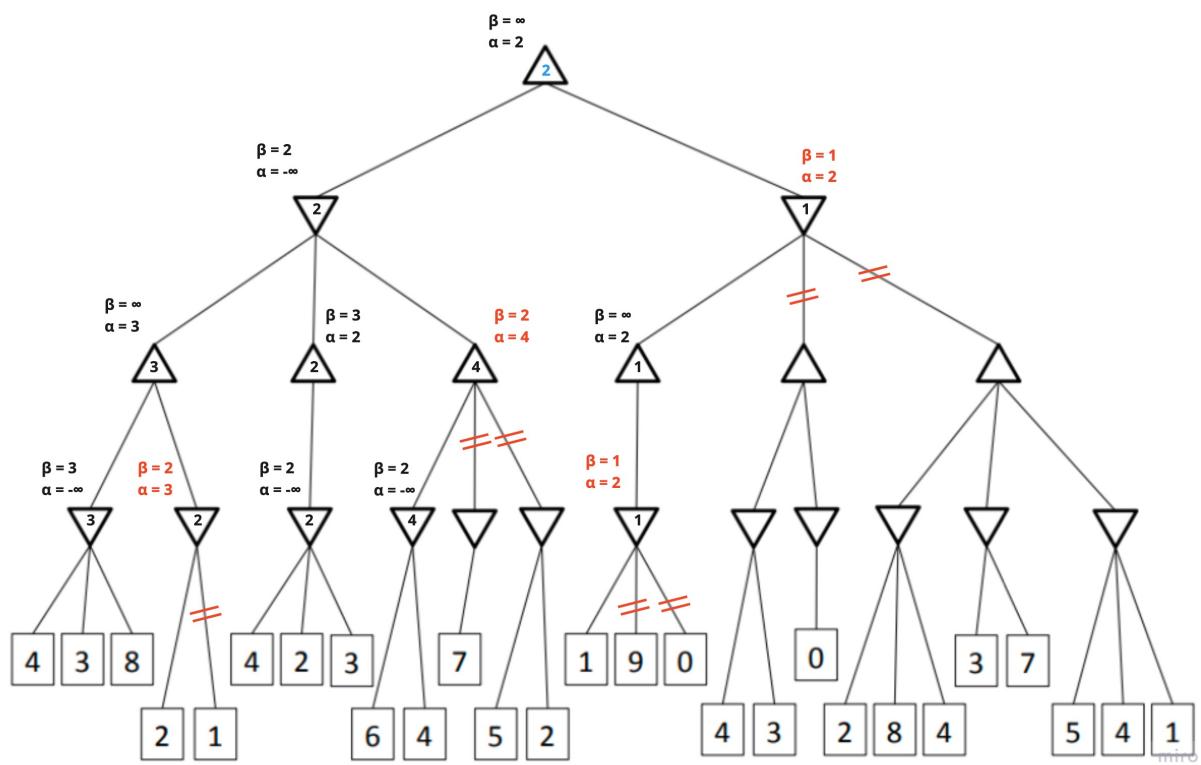
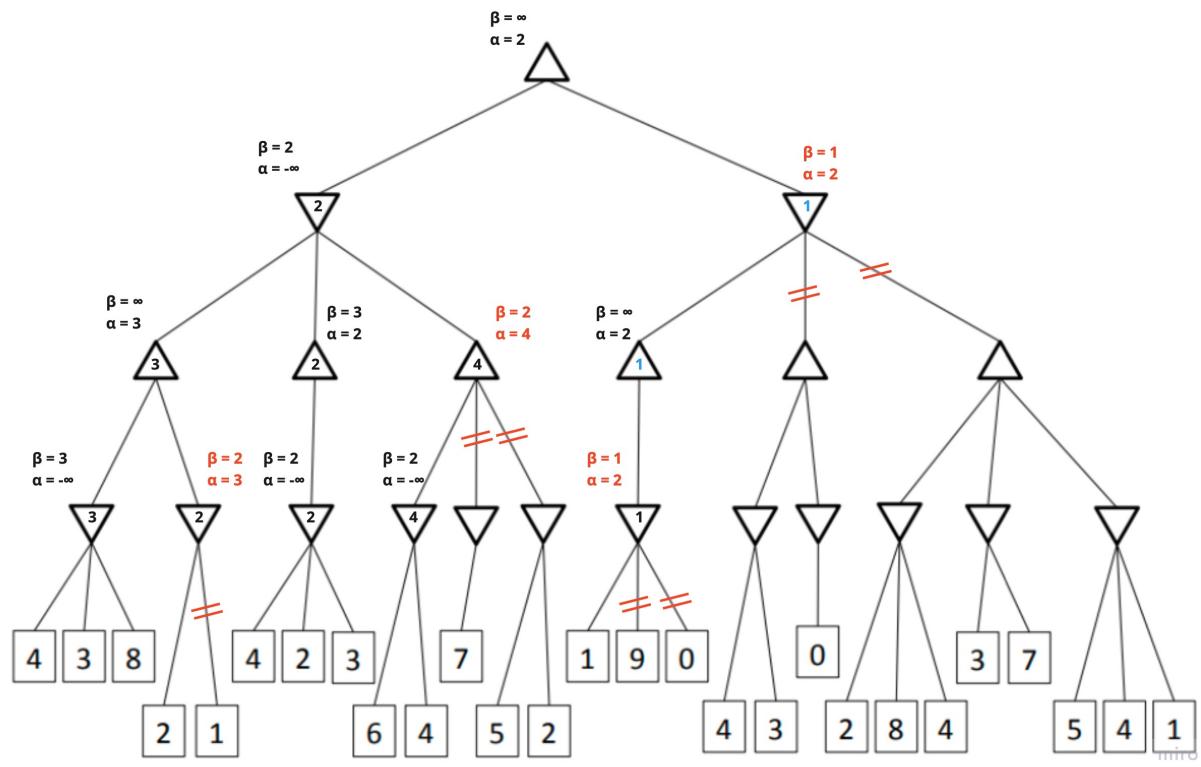
### b. Tia nhánh $\alpha - \beta$ (Chi tiết các bước)





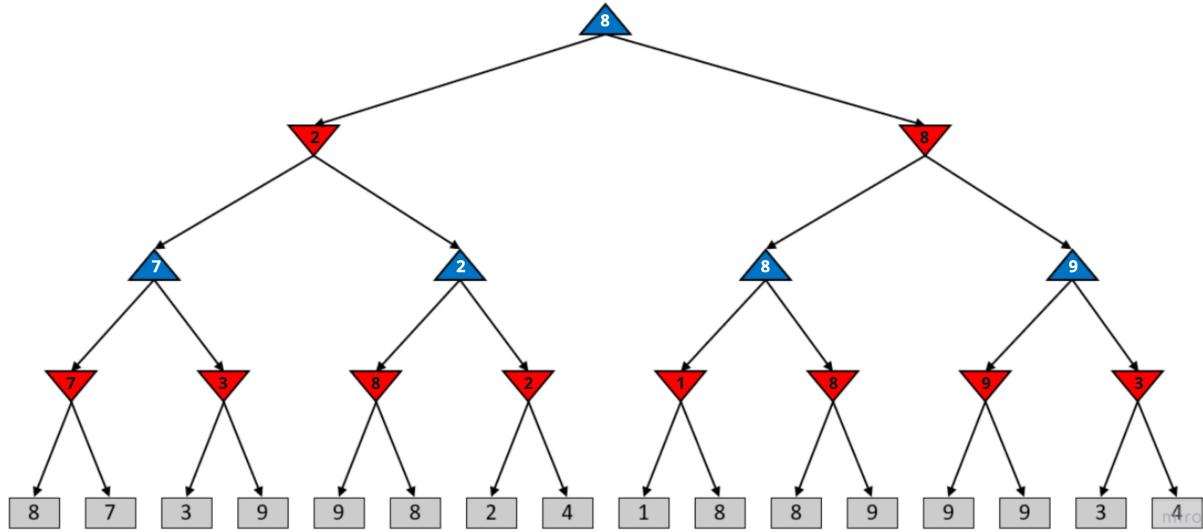




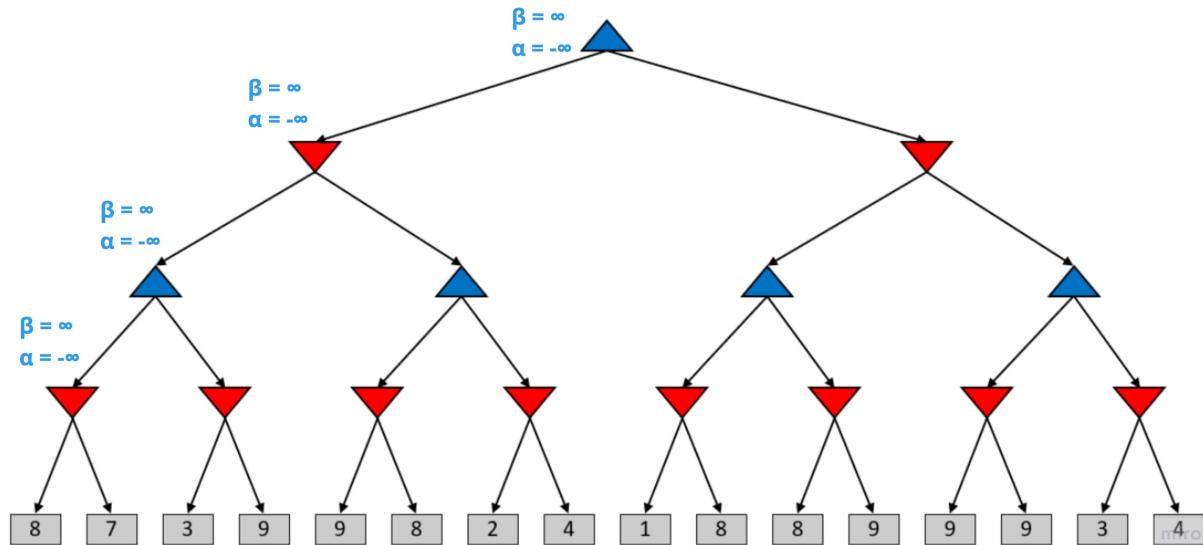


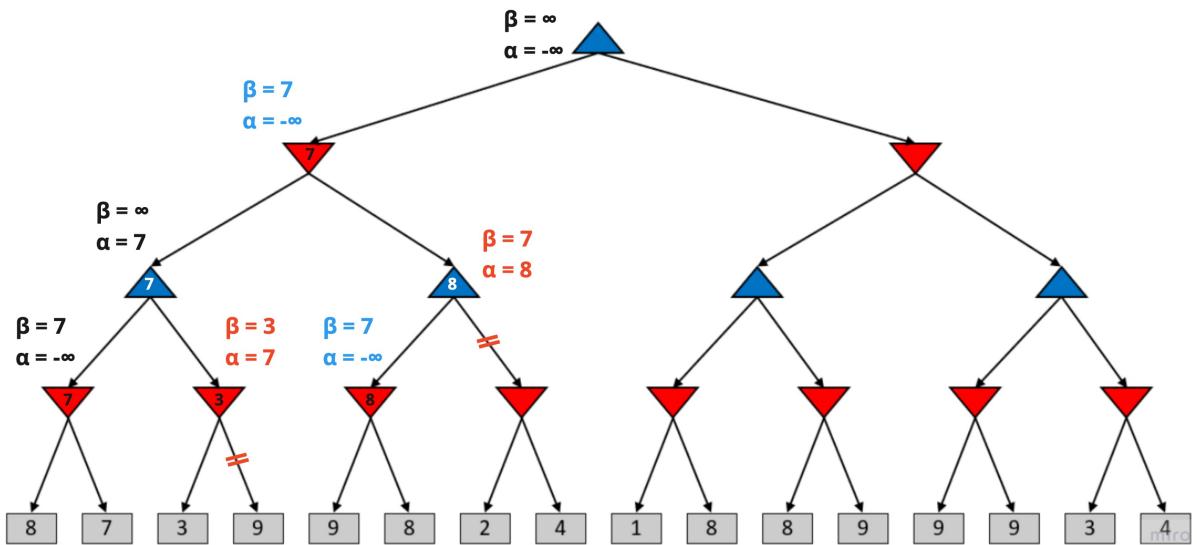
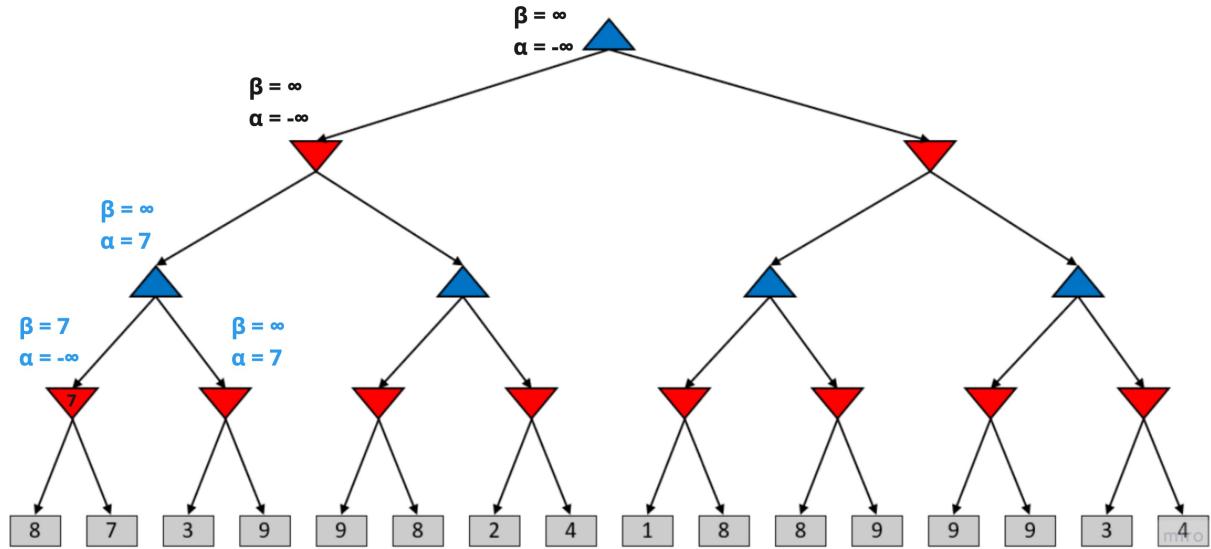
# Câu 3. Tìm kiếm đối kháng

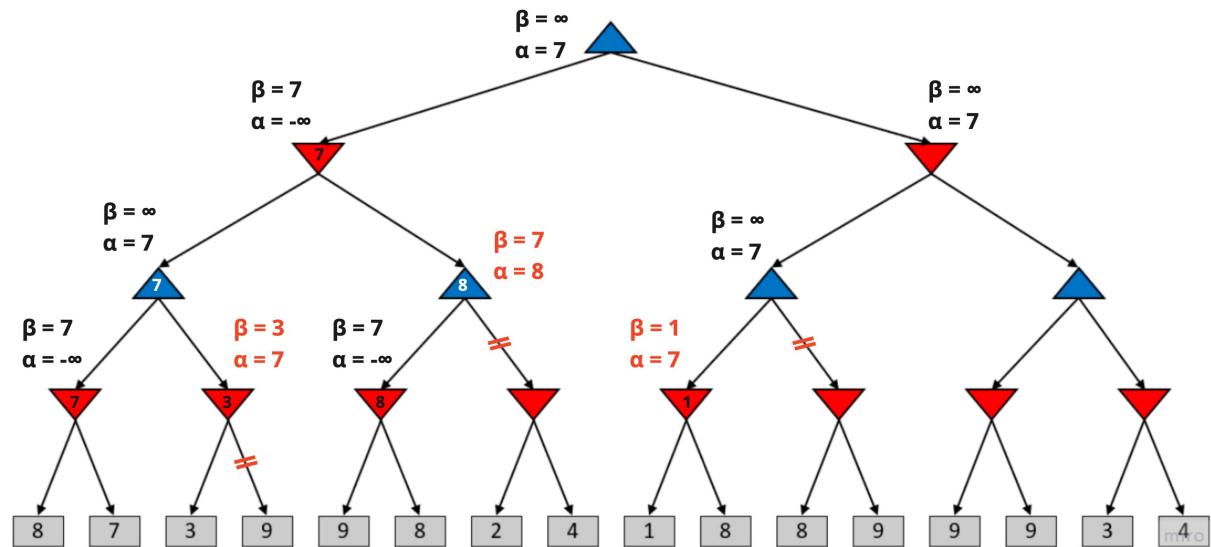
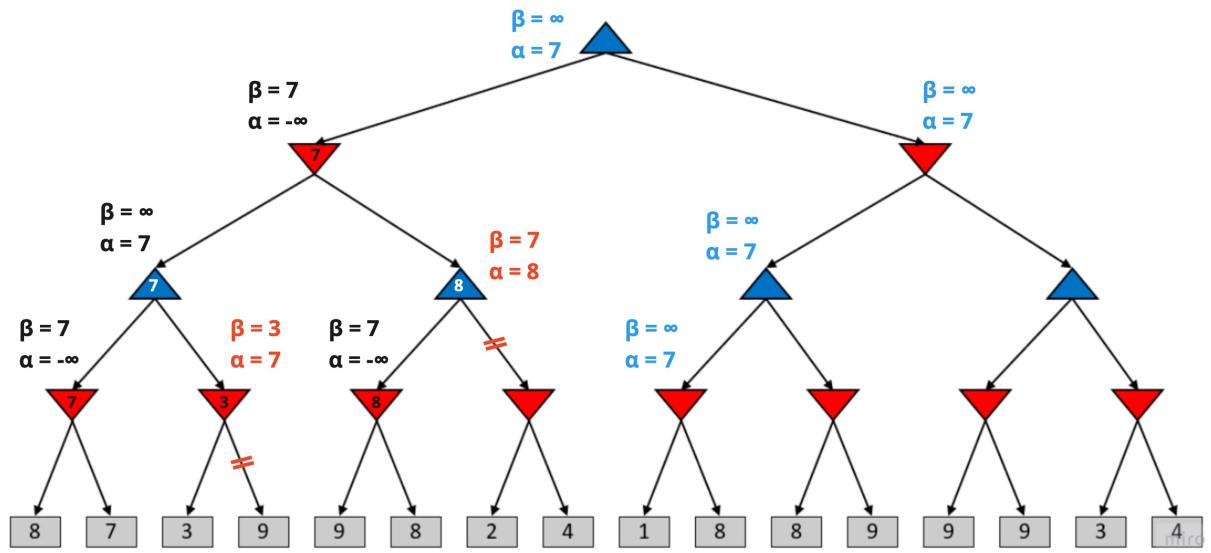
## a. Tính toán minimax

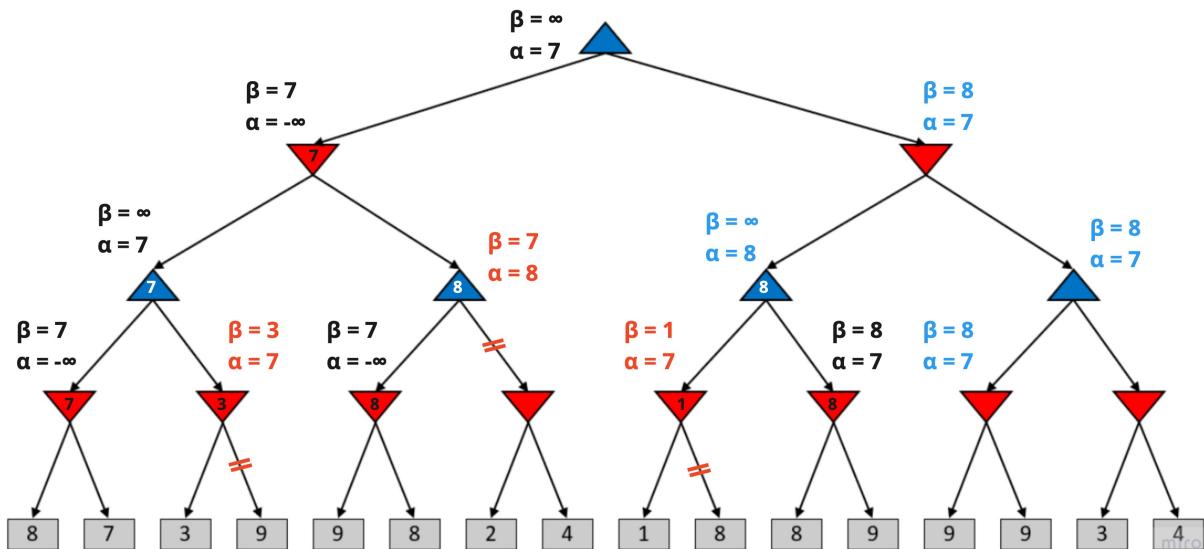
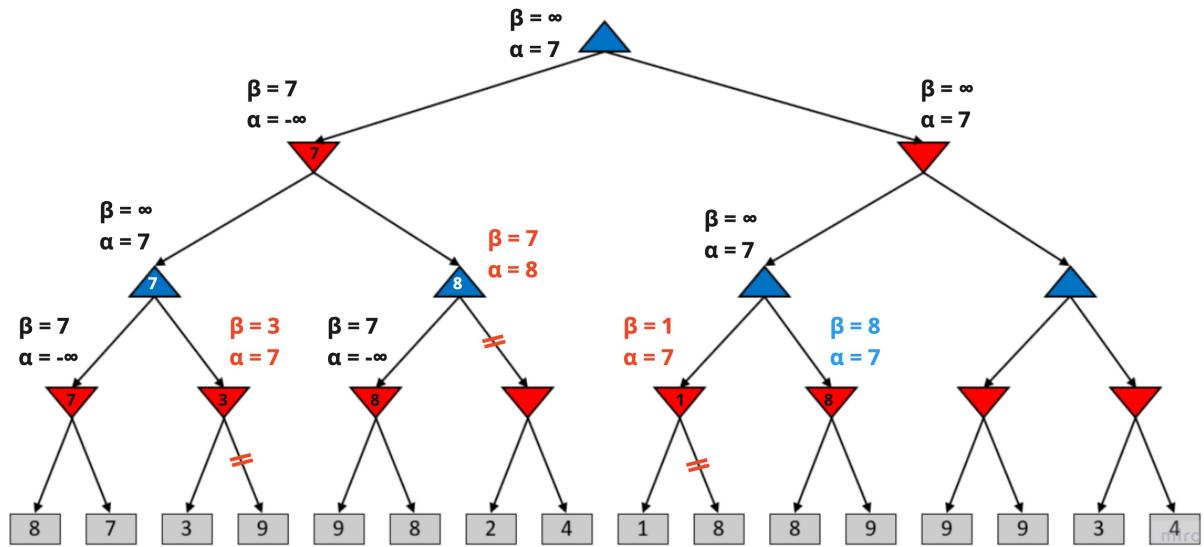


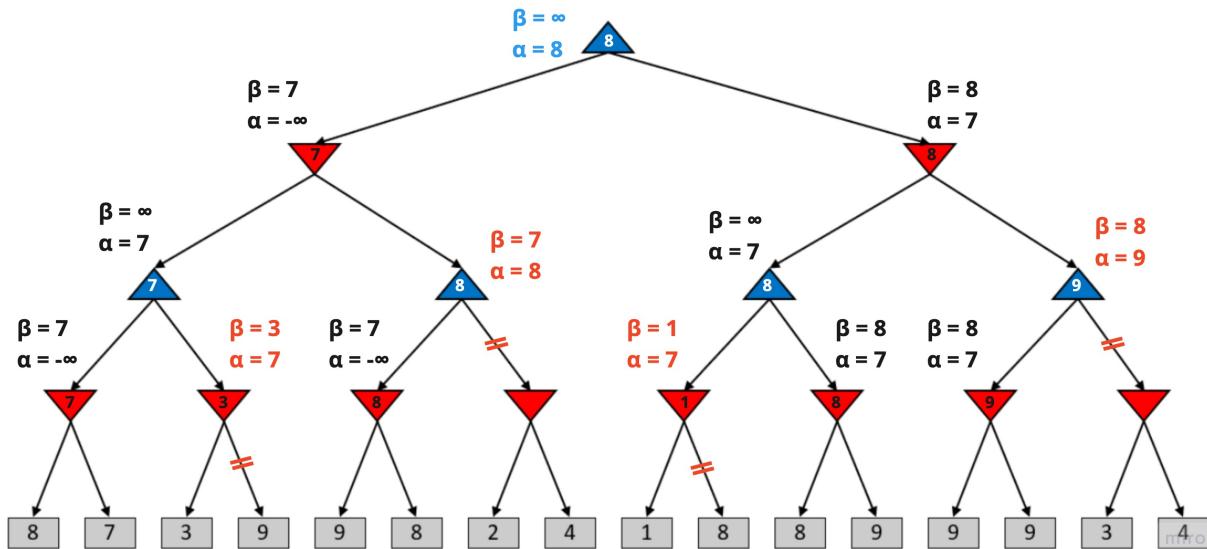
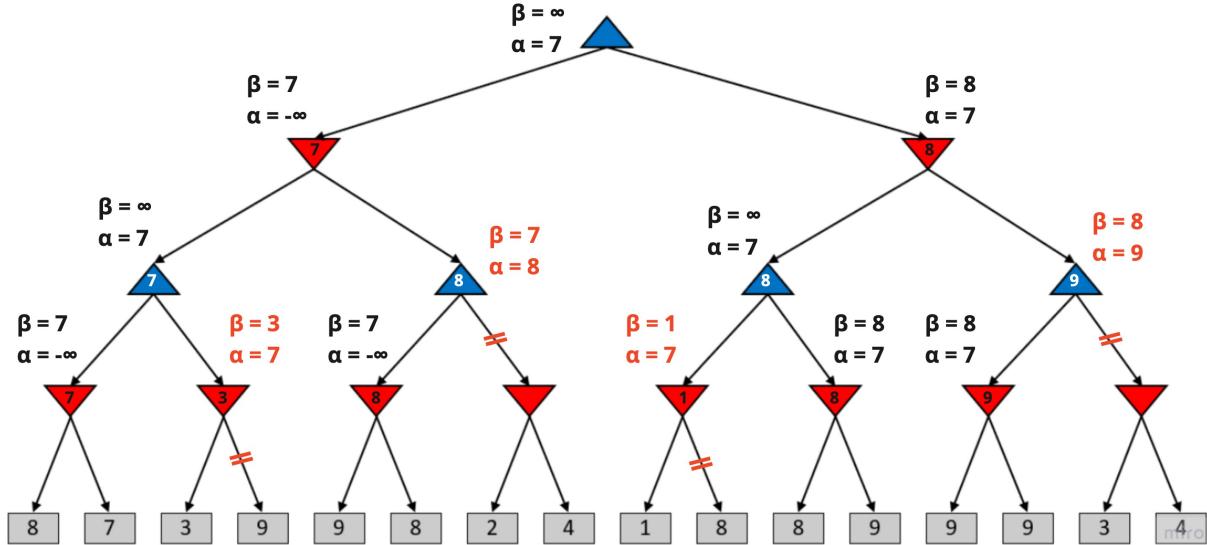
## b. Tỉa nhánh $\alpha - \beta$ (Chi tiết từng bước)





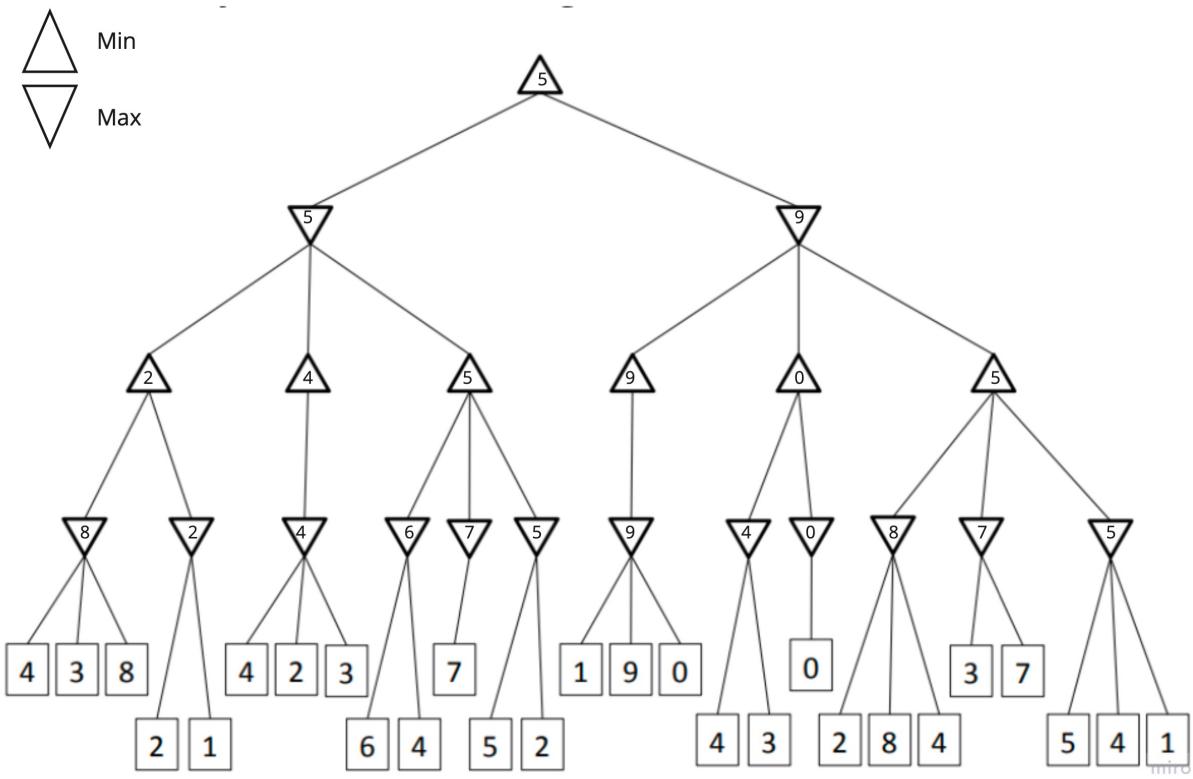




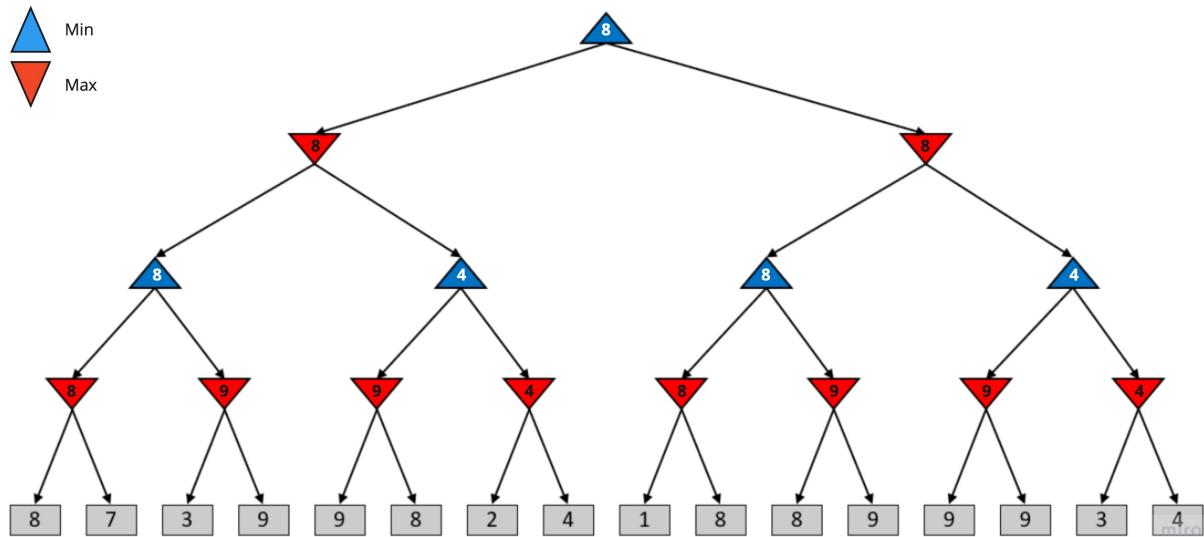


## Câu 4. Thực hiện lại bài 2 và 3 với Min & Max ngược lại

Hình 2.

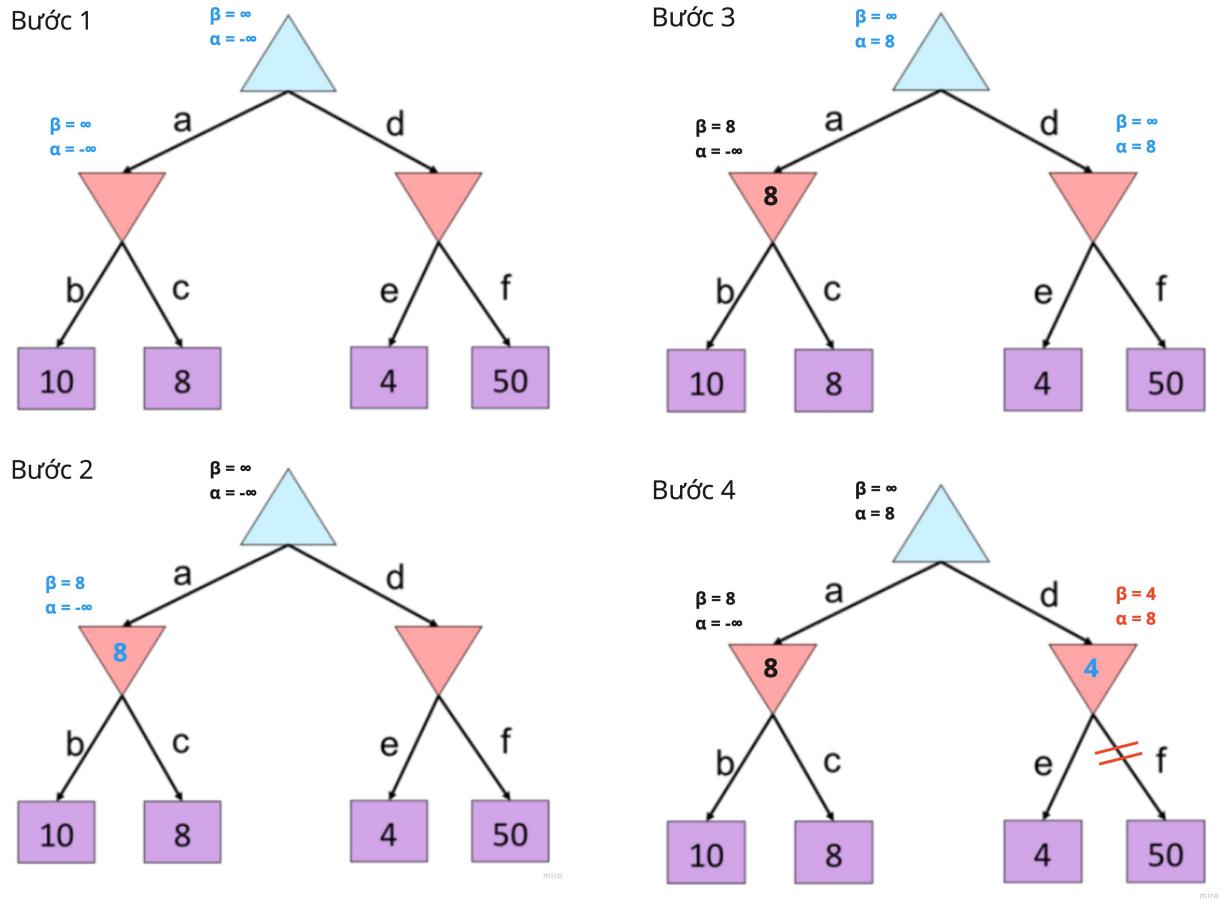


**Hình 3.**

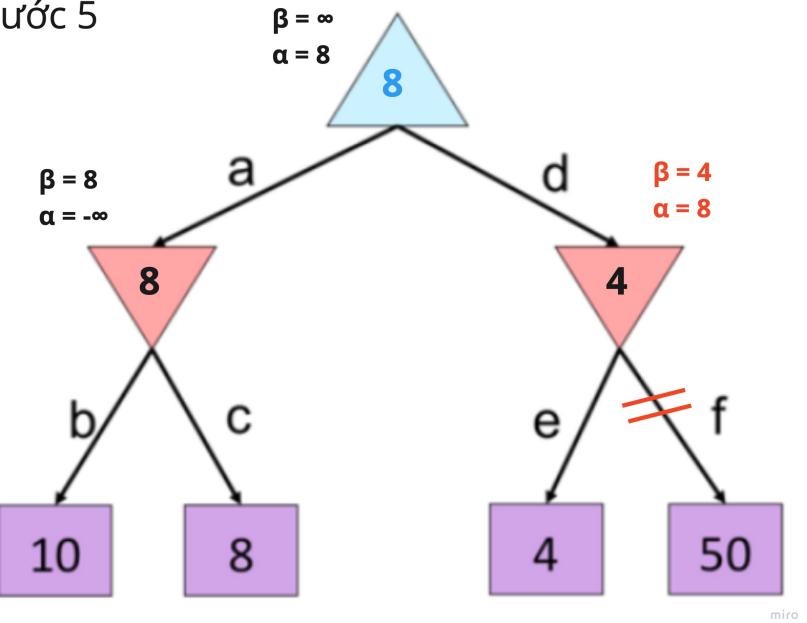


## Bài 5. Tỉa nhánh các cây tìm kiếm

## Hình a.

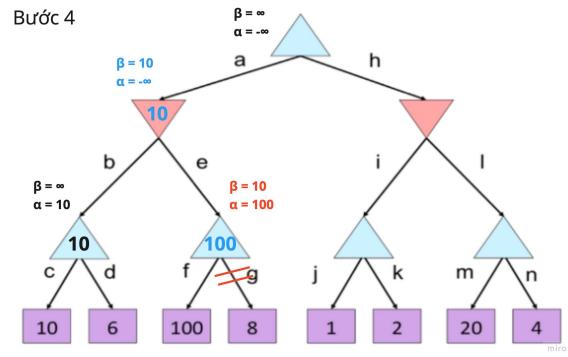
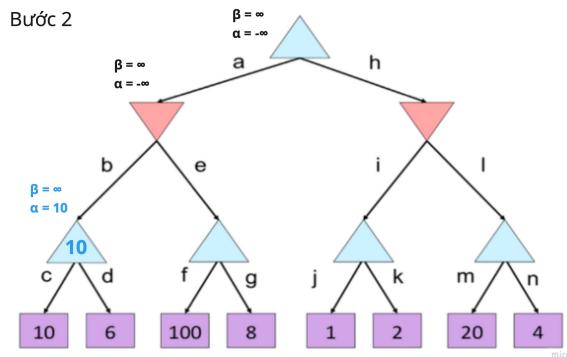
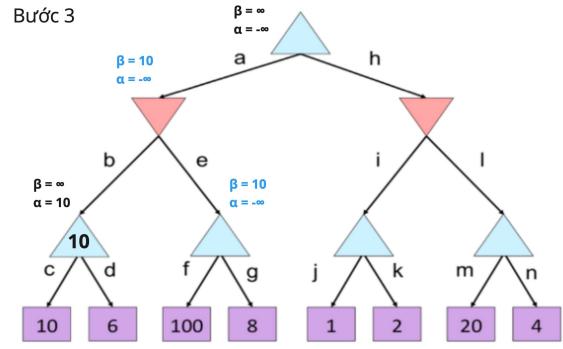
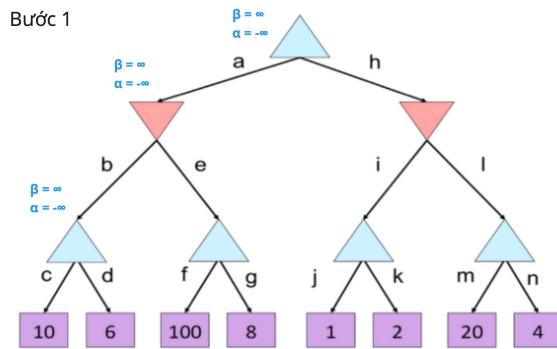


### Bước 5

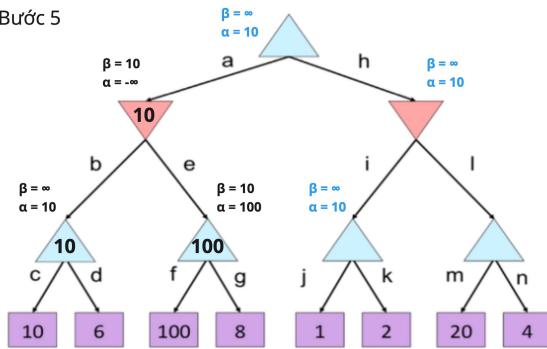


miro

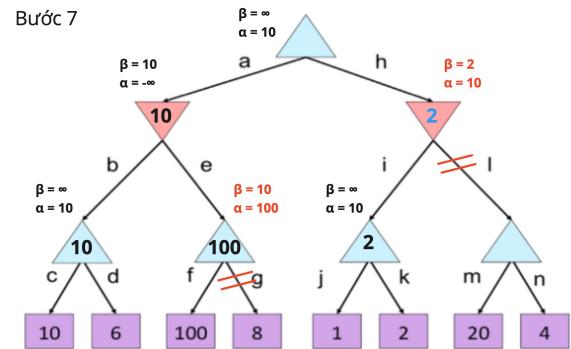
### Hình b.



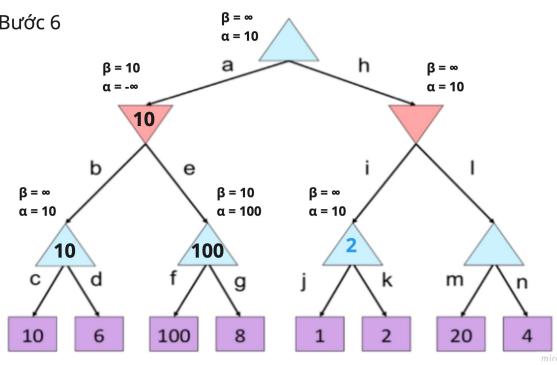
Bước 5



Bước 7



Bước 6



Bước 8

