



Bài toán Knapsack:

Trong bảo tàng có N viên đá quý có trọng lượng và giá trị tương ứng. Bạn là một siêu trộm, nhưng hôm nay chỉ mang một cái balo có khả năng chứa tối đa là M . Thời gian không còn nhiều, bạn phải lấy được giá trị cao nhất nhưng đảm bảo balo không quá tải.

Bạn sẽ làm như thế nào?

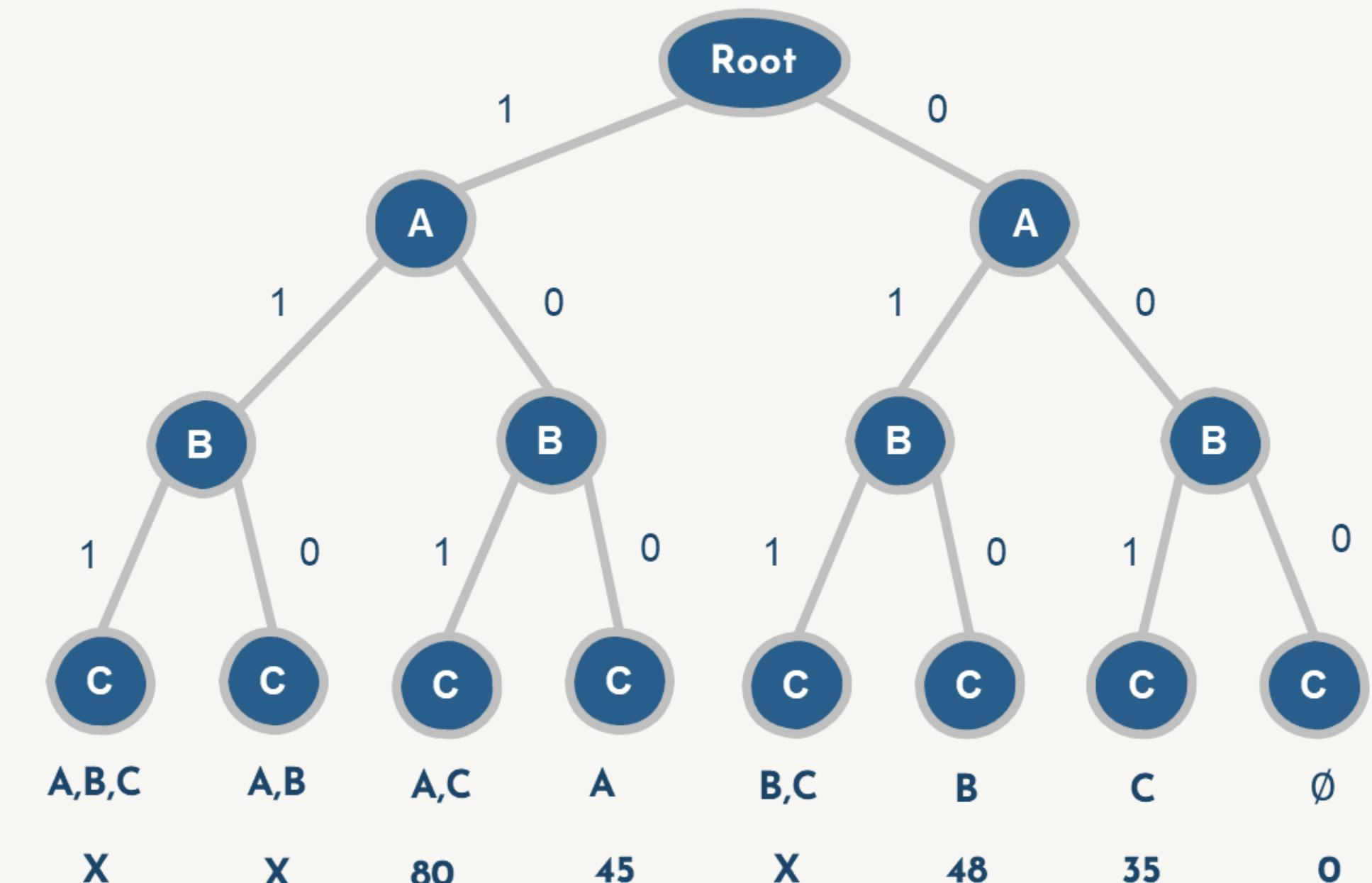


Backtracking

	A	B	C
Trọng lượng	5	8	3
Giá trị	45	48	36

+ $N = 3$
 $M = 10$

+ Không lấy: 0
Lấy: 1



Cải tiến Backtracking



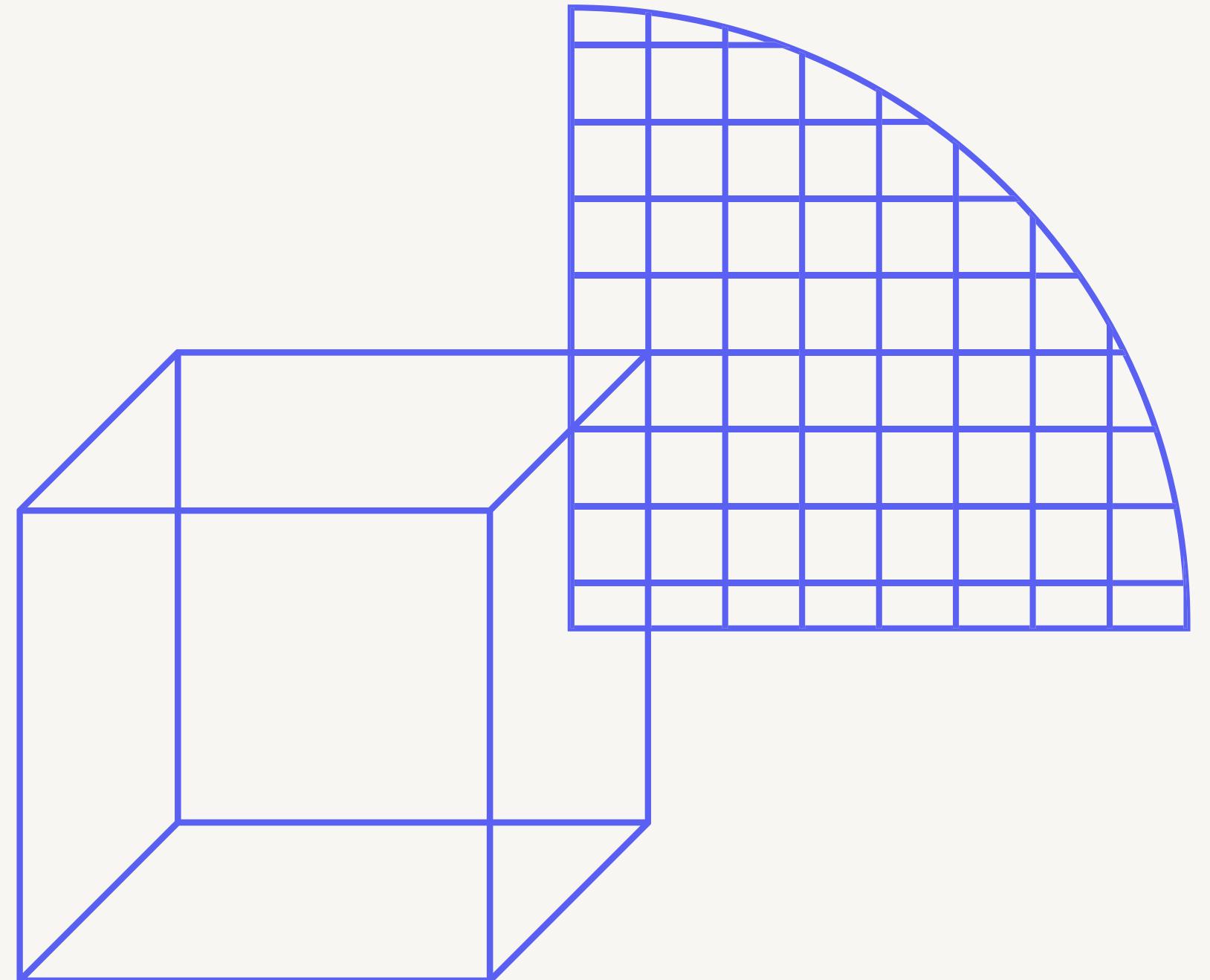
Liệu có giải pháp nào tốt hơn?



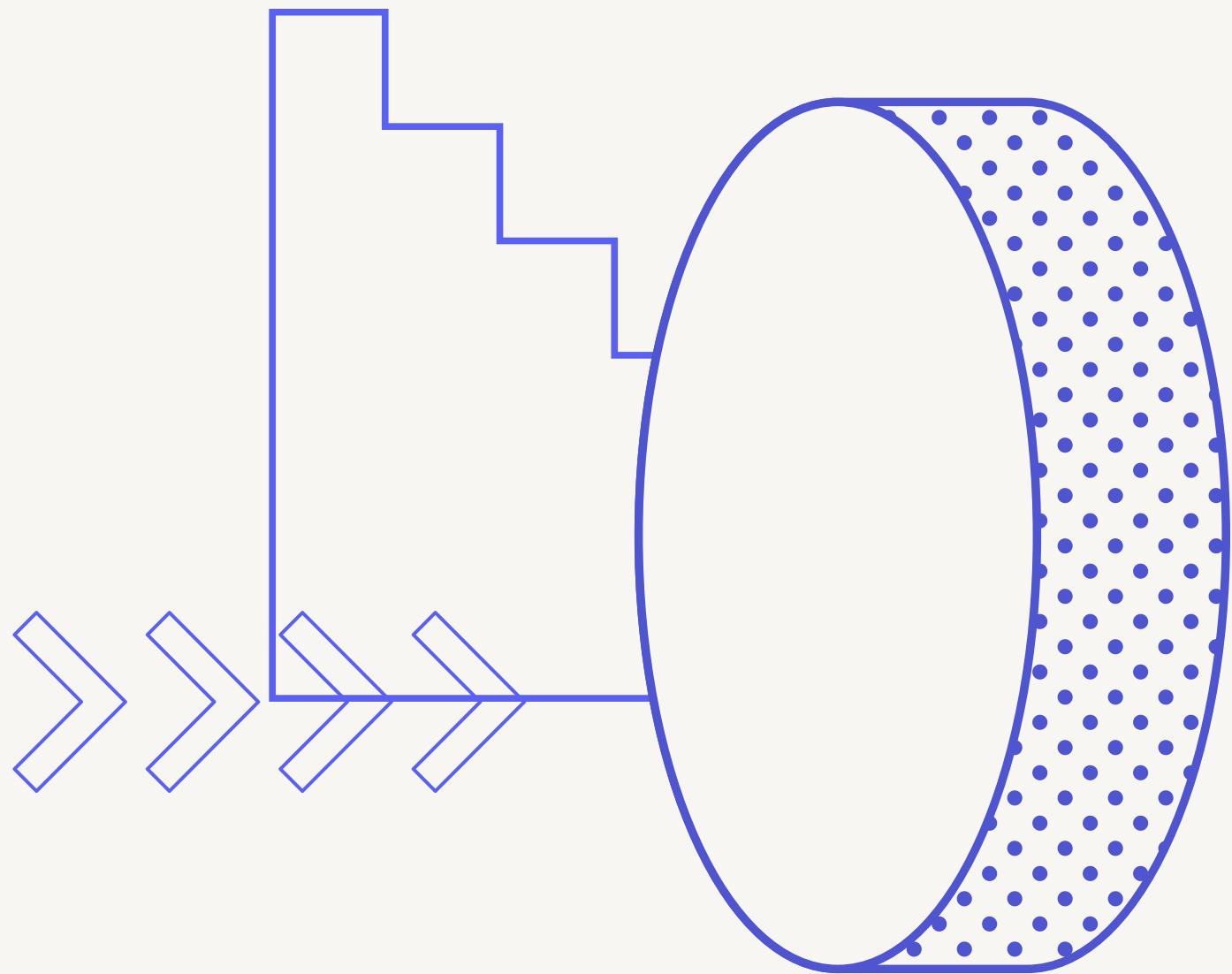
Ta có thể làm tốt hơn nếu chúng ta biết ràng buộc về giải pháp tốt nhất.



BRANCH AND BOUND

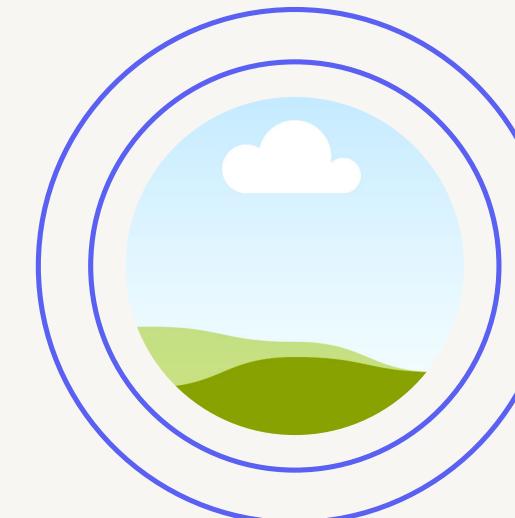


Thành viên nhóm 01



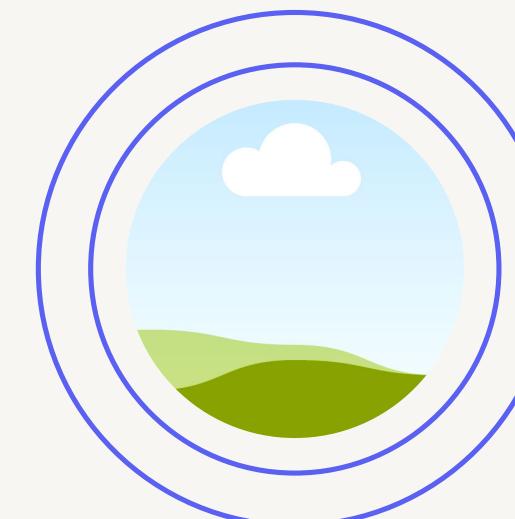
NGUYỄN ĐỨC LẬP - 21522280

Nhóm trưởng



NGÔ ĐĂNG KHOA - 21522821

Thành viên



HỒ ĐĂNG KHOA - 21520992

Thành viên

Nội dung

Những nội dung sẽ được thảo luận trong phần này:

+ KHÁI NIỆM

+ ĐẶC ĐIỂM THUẬT TOÁN

+ MÔ HÌNH PHỔ QUÁT

+ BÀI TOÁN ÁP DỤNG

+ ƯU VÀ NHƯỢC ĐIỂM

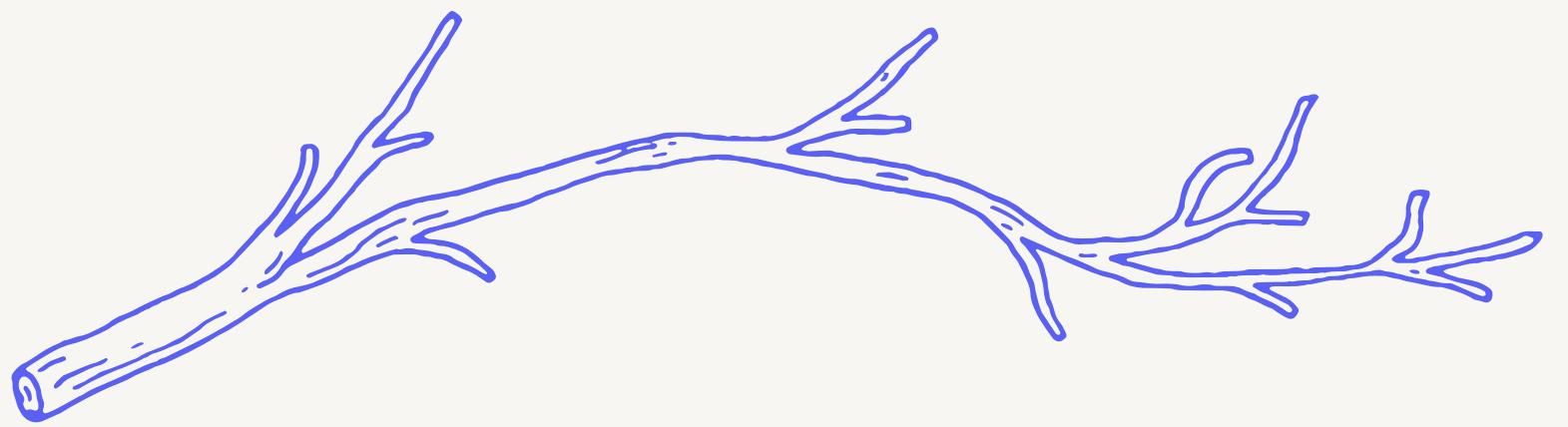
+ KHI NÀO SỬ DỤNG BnB



KHÁI NIỆM

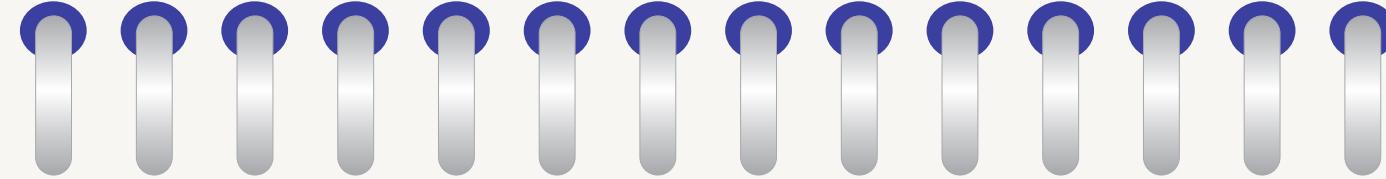
Branch and Bound là gì?





BRANCH?

Branch là cái dùng để phân nhánh và ghi lại luồng của lịch sử. Branch đã phân nhánh sẽ không ảnh hưởng đến branch khác nên có thể tiến hành nhiều thay đổi đồng thời trong cùng 1 repository.



BOUND?

Cận là ước lượng không phải dự đoán. Để xác định cận, phải chắc rằng số đó lớn hơn hoặc bằng (cận trên) hoặc bé hơn hoặc bằng (cận dưới) giá trị lớn nhất hoặc giá trị bé nhất trong 1 bộ.

Branch and Bound?

Để dễ hình dung thì BnB giống như một “cây cảnh” và nhiệm vụ của ta là dùng “cận” như một cái “kéo” để tỉa gọn những nhánh “giải pháp” kém hiệu quả như để loại bỏ những “nhánh xấu” thì ta sẽ thu được “cây càng” đẹp hơn.

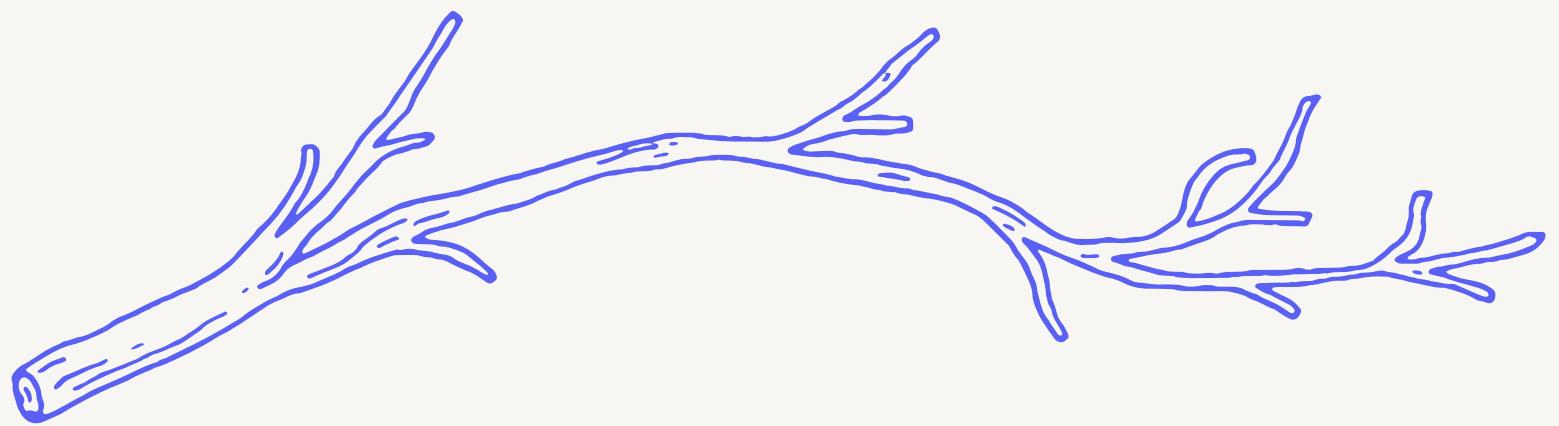
- Thuật toán Branch and Bound là thuật toán được thiết kế để sử dụng với các bài toán tối ưu tổ hợp và tối ưu rời rạc.
- Về bản chất Branch and Bound là cải tiến của Backtracking, nó ràng buộc giá trị tốt nhất từ gốc đến mọi nút.





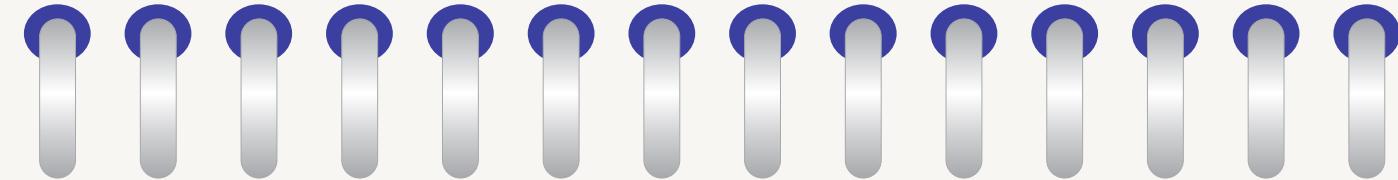
ĐẶC ĐIỂM BÀI TOÁN





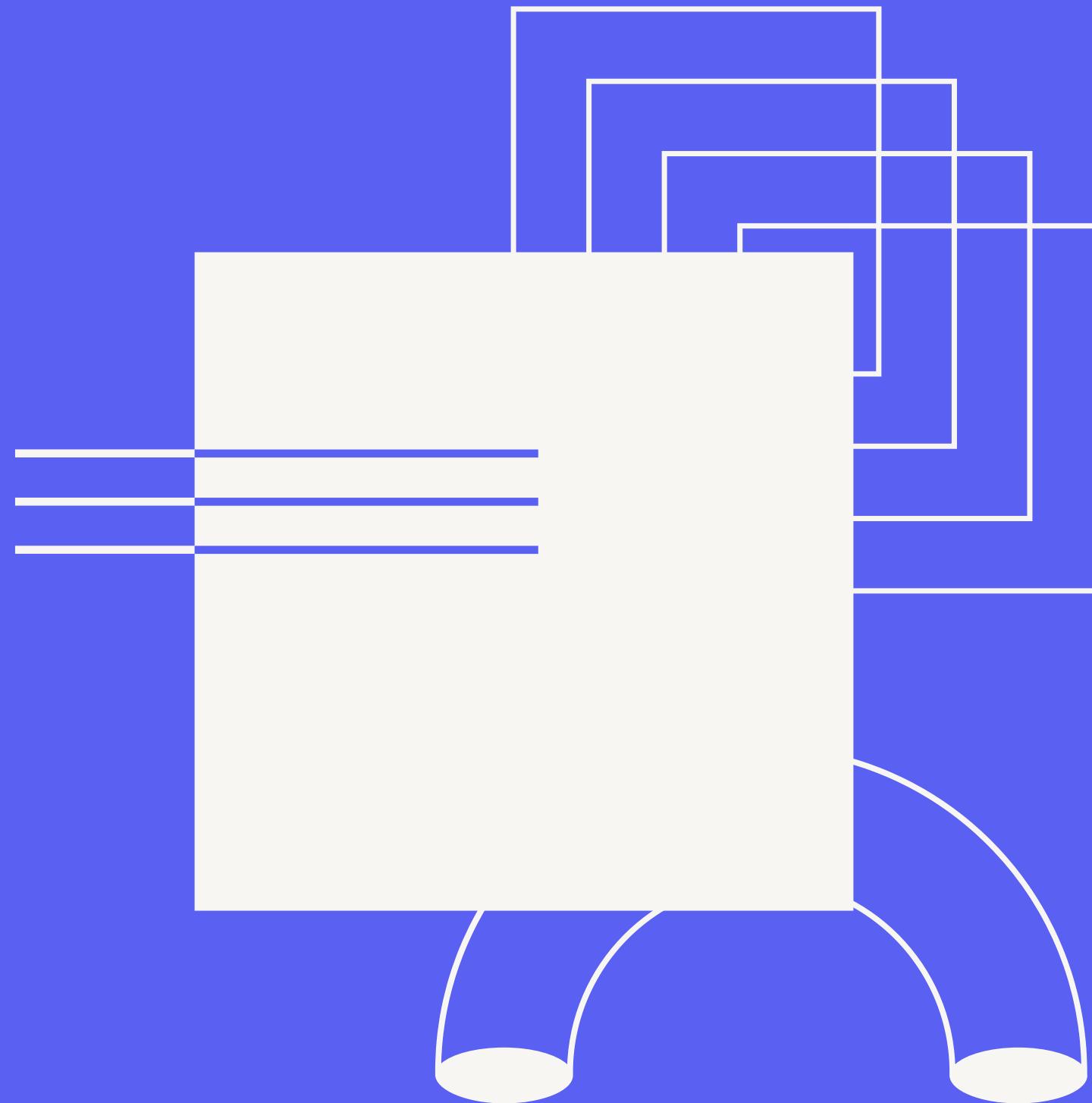
BRANCH

- Mỗi nút n biểu diễn cho một phương án nào đó.
- Nút n có các nút con tương ứng với các khả năng có thể lựa chọn phương án từ nút n.



BOUND

- Nút được kiểm tra dựa trên các cận trên và dưới của giải pháp tối ưu.
- Nếu giải pháp của nút đang xét chưa tối ưu thì ta không nên tiếp tục.



MÔ HÌNH PHỔ QUÁT

Một mô hình ta có thể dựa vào đó để phát triển lên những thuật toán riêng.



- 1. Khởi tạo:** Tạo nút gốc

- 2. Phân chia:** Chia tập giá trị thành các tập con

- 3. Đánh giá:** Xem nút con có là giải pháp tốt hơn

- 4. Lưới cắt:** Sử dụng bound để loại bỏ các nút không tìm năn

- 5. Lặp lại:** Lặp lại bước 2 đến 4 cho đến hết

- 6. Kết thúc:** Trả về giải pháp tốt nhất tìm được

SỬ DỤNG ĐỆ QUY

Duyệt theo chiều sâu (Depth first search)

```
<<CODE>

function branch_and_bound(node):
    if node is a leaf node:
        return the solution value at the node
    else:
        best_solution = infinity
        for each child of node:
            if child is promising:
                solution = branch_and_bound(child)
                if solution < best_solution:
                    best_solution = solution
        return best_solution
```

SỬ DỤNG STACK

Duyệt theo chiều sâu (Depth first search)

```
<CODE>

function branch_and_bound(root):
    stack.push(root)
    best_solution = infinity
    while not stack.empty():
        node = stack.pop()
        if node is a leaf node:
            best_solution = min(best_solution, node.solution)
        else:
            for each child of node:
                if child is promising:
                    stack.push(child)
    return best_solution
```



BÀI TOÁN ÁP DỤNG

Lý thuyết phải đi đôi với thực hành





BÀI TOÁN TÌM DÃY CON TĂNG DÀI NHẤT (LONGEST INCREASING SUBSEQUENCE)

Một bài toán đơn giản, dễ hiểu.



BÀI TOÁN BA LÔ (KNAPSACK)

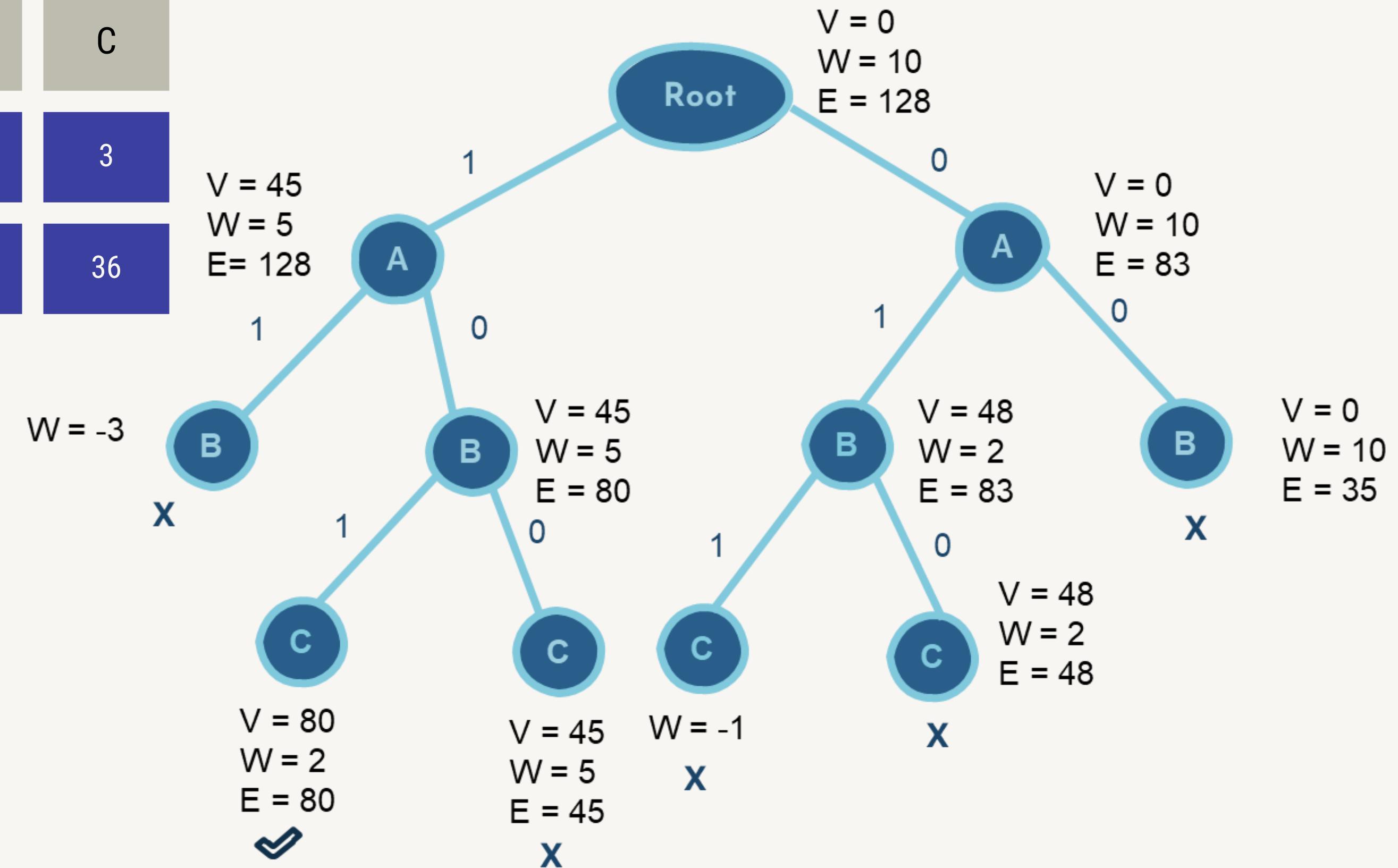
Bài toán nâng cao, thể hiện được hiệu quả của BnB.

Knapsack:

	A	B	C
Trọng lượng	5	8	3
Giá trị	45	48	36

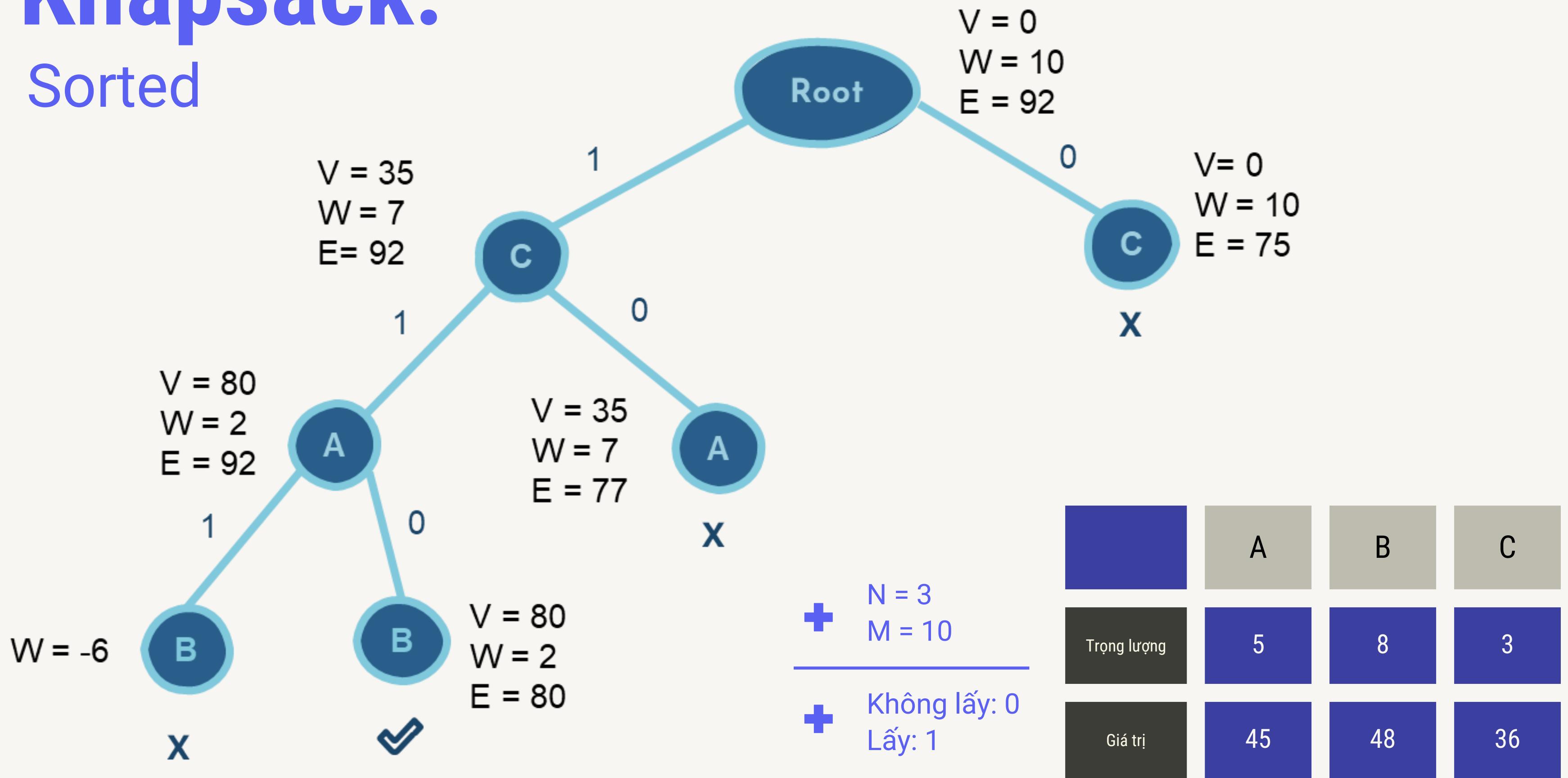
+ $N = 3$
M = 10

+ Không lấy: 0
+ Lấy: 1



Knapsack:

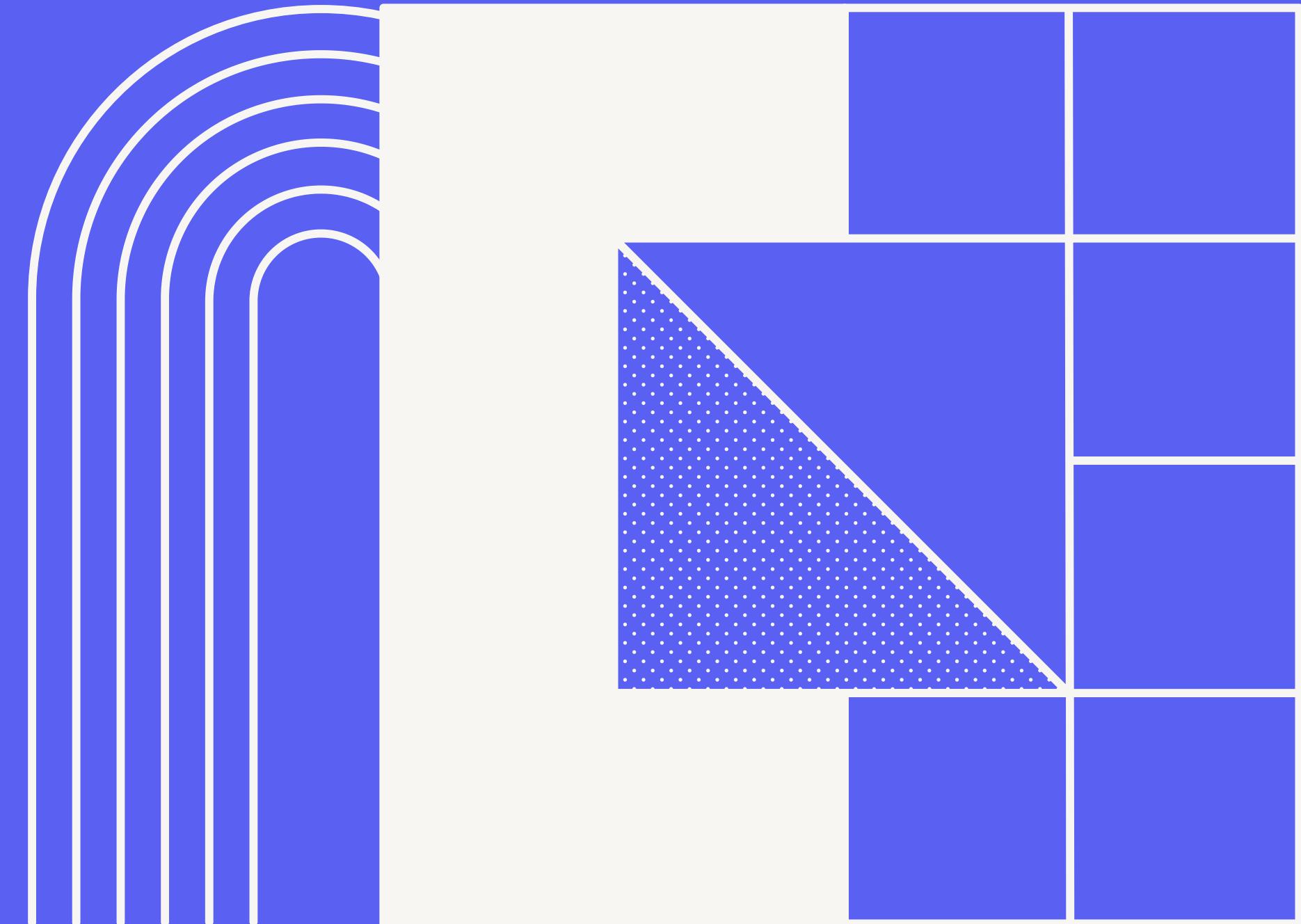
Sorted

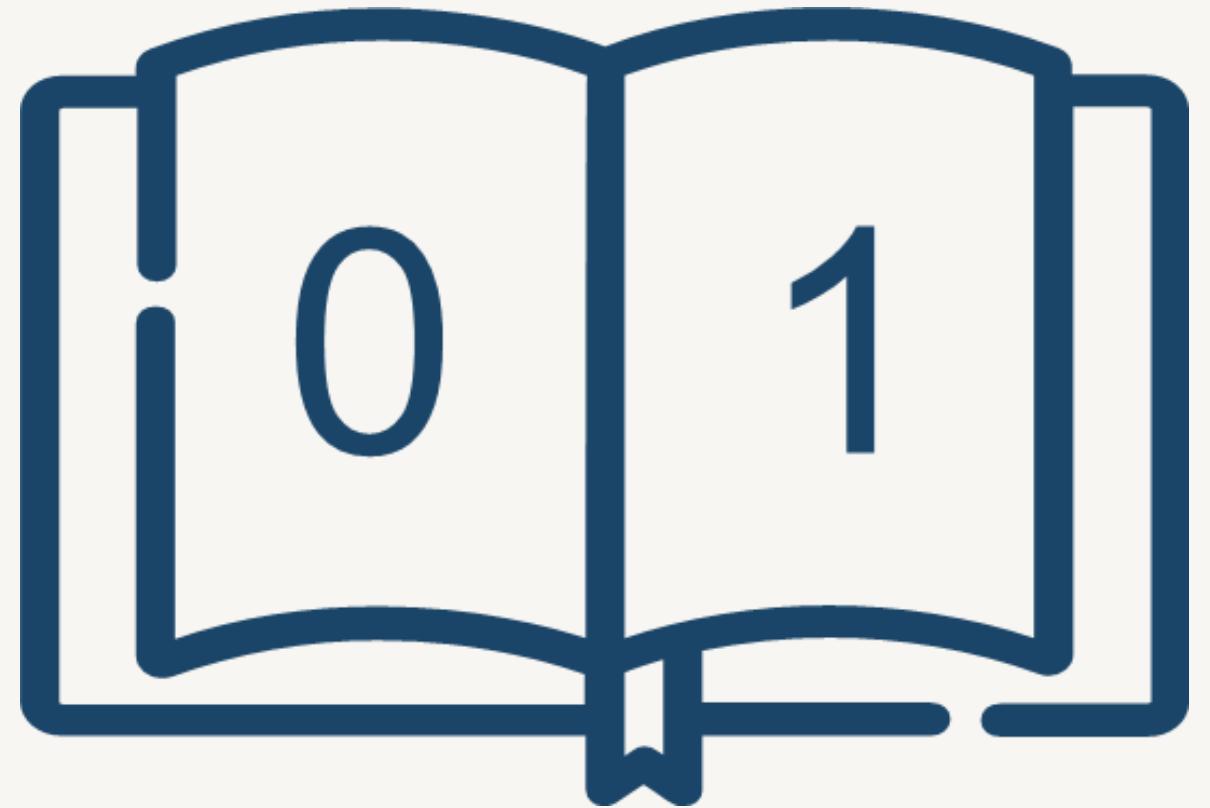




KHI NÀO SỬ DỤNG BnB?

Một câu hỏi không hề đơn giản.





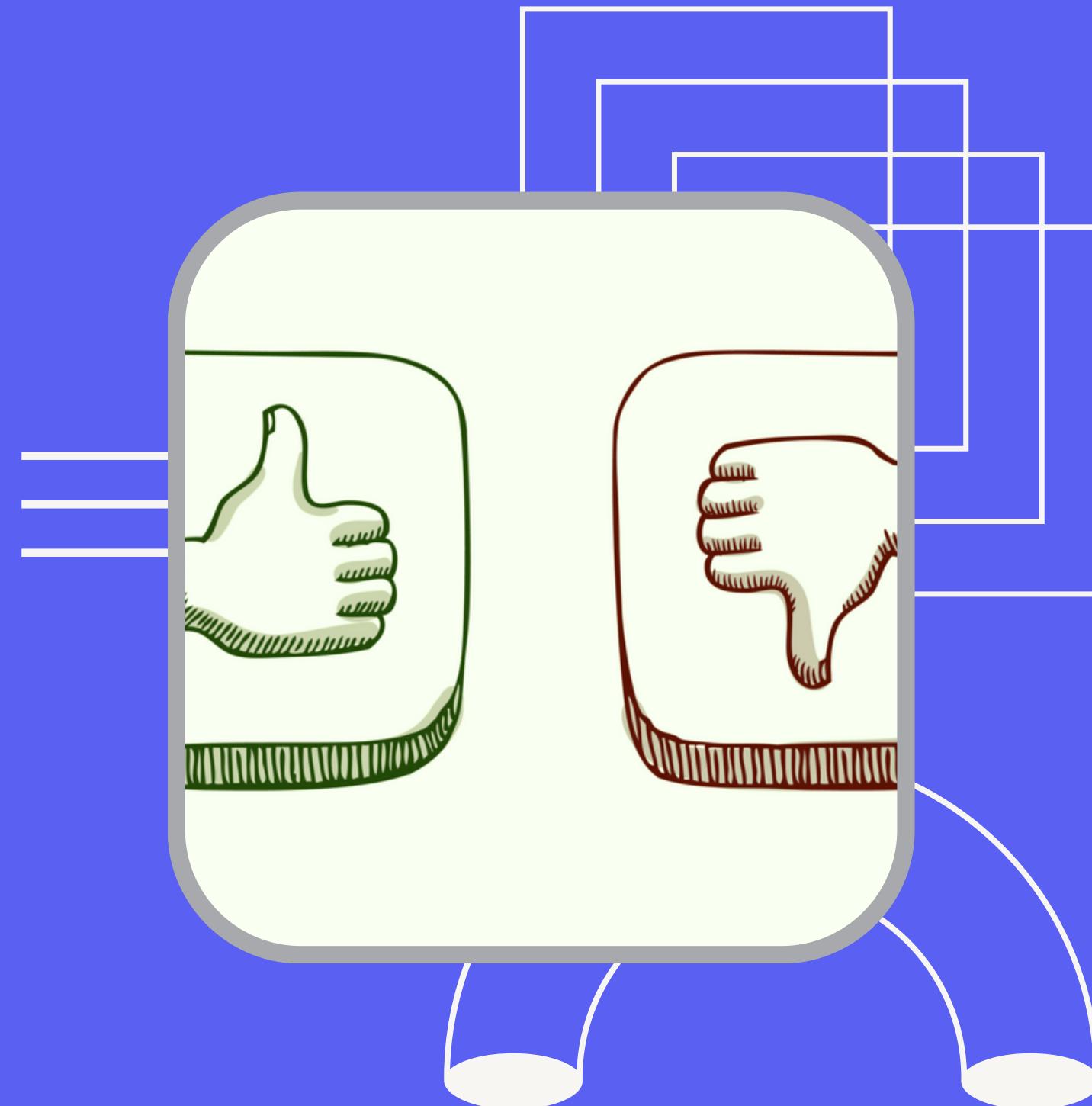
BÀI TOÁN TỐI ƯU RỜI RẠC (DISCRETE OPTIMIZATION PROBLEM)

Zero-One Integer Programming, Network Flow Problem,...



BÀI TOÁN TỐI ƯU TỔ HỢP (COMBINATORIAL OPTIMIZATION PROBLEMS)

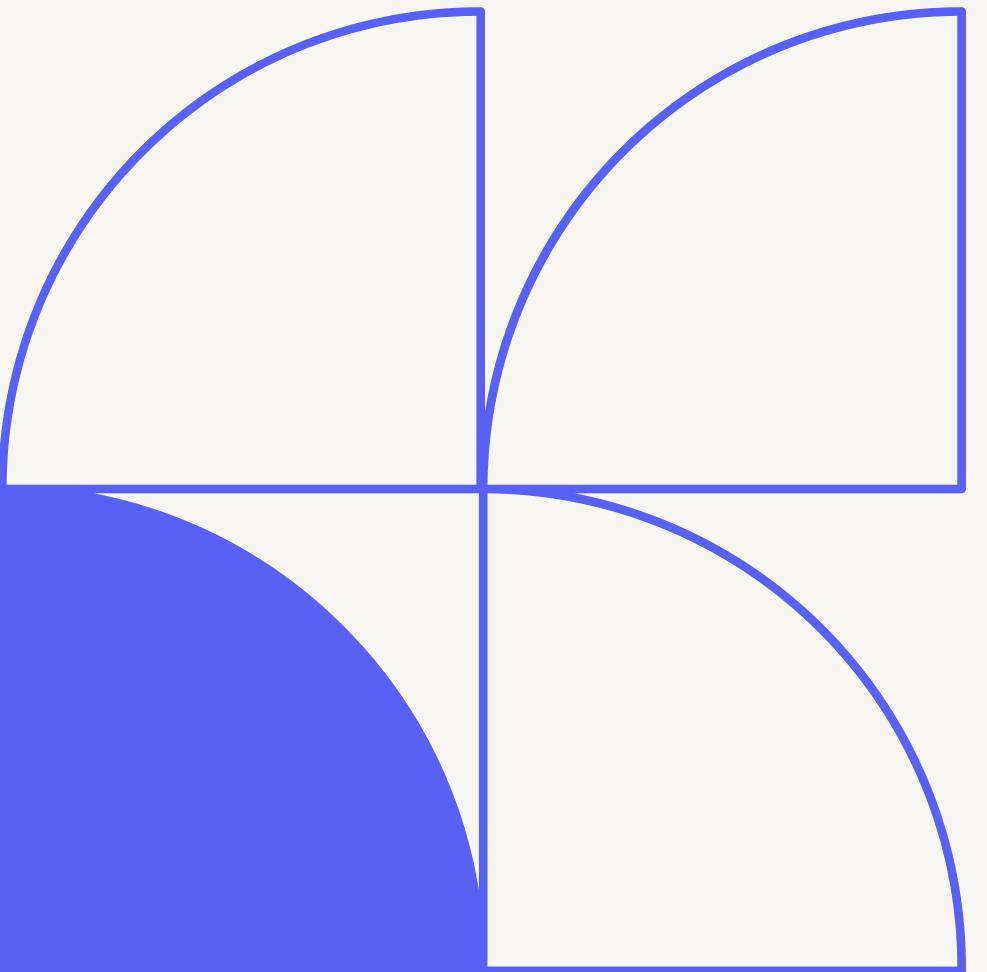
Boolean Satisfiability, Integer Linear Programming,...



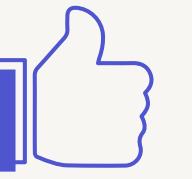
ƯU & NHƯỢC ĐIỂM

Cái gì cũng có 2 mặt.

ƯU ĐIỂM



#



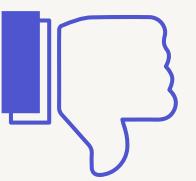
#



#



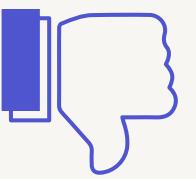
#



#



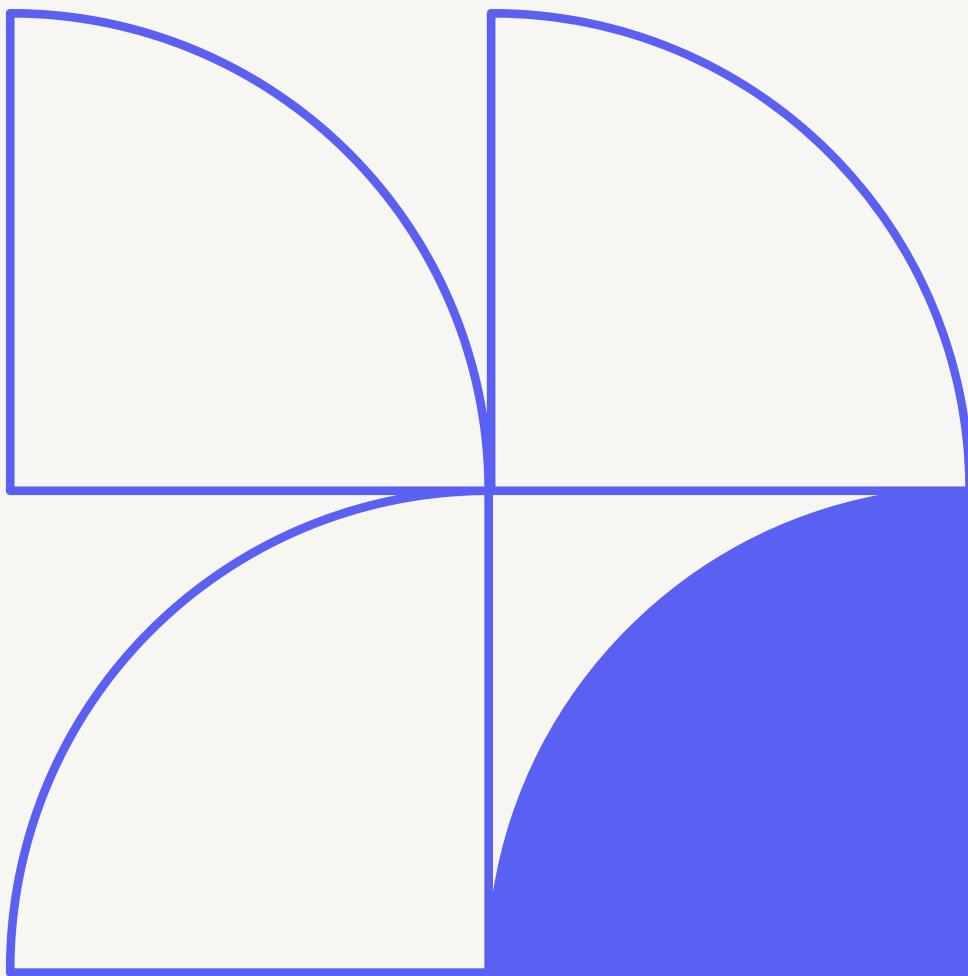
#



#



#



NHƯỢC ĐIỂM

NGUỒN THAM KHẢO



https://en.wikipedia.org/wiki/Branch_and_bound



<https://www.geeksforgeeks.org/branch-and-bound-algorithm>



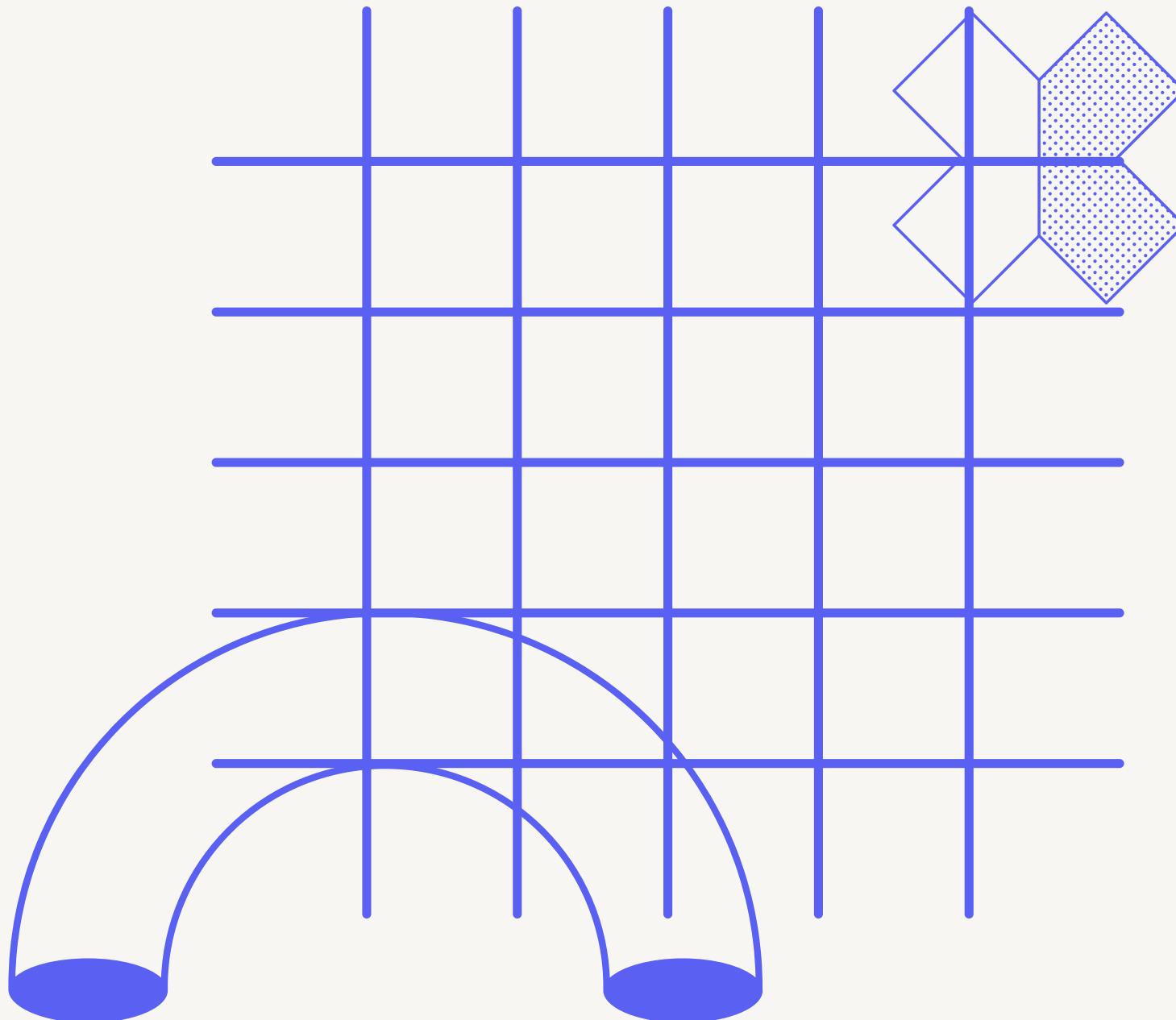
<https://www.coursera.org/lecture/discrete-optimization/knapsack-5-relaxation-branch-and-bound>



"Branch and Bound Algorithms - Principles and Examples" của Jens Clausen.



"Introduction to Algorithms" của Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest và Clifford Stein.



Thank you!

Email us at 21522280@gm.uit.edu.vn
if you have more questions.