

*Professor: Bùi Tiến Lên*

# DATA STRUCTURE ALGORITHM

The image shows a stage with blue curtains. A spotlight shines down on a circular graphic in the center. The graphic consists of a dashed brown circle with 24 small white circles with purple outlines arranged around its perimeter. The text 'GROUP7' is written in a black, serif font in the center of the dashed circle.

GROUP7

# MEMBER

---

---







Nguyễn Anh Tuấn  
Edit





Phùng Hữu Tài  
Edit



Nguyễn Thành Luân  
Presentation



Trần Thông Lực  
Presentation





Phan Nhật Triều  
Presentation

# OUR TEAM



Nguyễn Anh Tuấn  
Edit



Phùng Hữu Tài  
Edit



Nguyễn Thành Luân  
Presentation



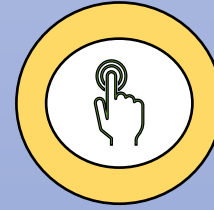
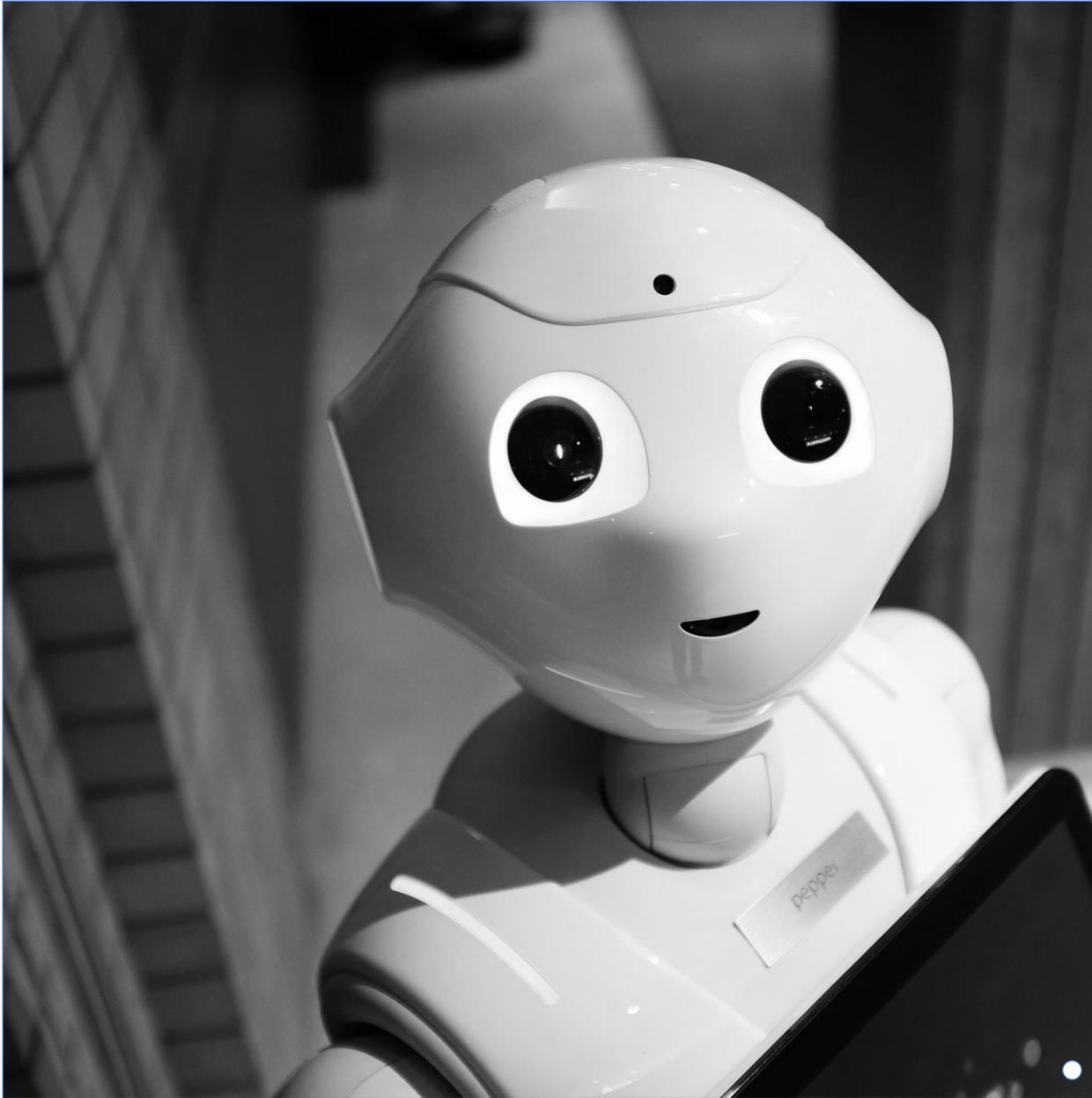
Trần Thông Lực  
Presentation



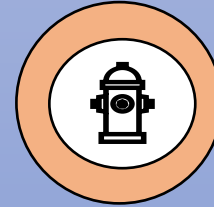
Phan Nhật Triều  
Presentation

# GIẢI THUẬT NHÁNH CẬN





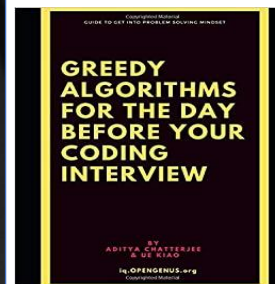
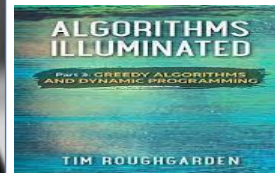
01. History of algorithm



02. Analyse algorithm



03. Application



# Processing...





Quy hoạch nguyên là mô hình toán học của rất nhiều bài toán nảy sinh trong các lĩnh vực khác nhau.



Tuy nhiên khác với các bài toán quy hoạch tuyến tính thông thường, bài toán quy hoạch nguyên rất khó giải. Thực tế thì chưa có một thuật toán tối ưu nào thực sự hữu hiệu để giải tất cả các bài toán quy hoạch nguyên.



Năm 1960, Land và Doig đưa ra thuật toán nhánh cận để giải bài toán quy hoạch nguyên



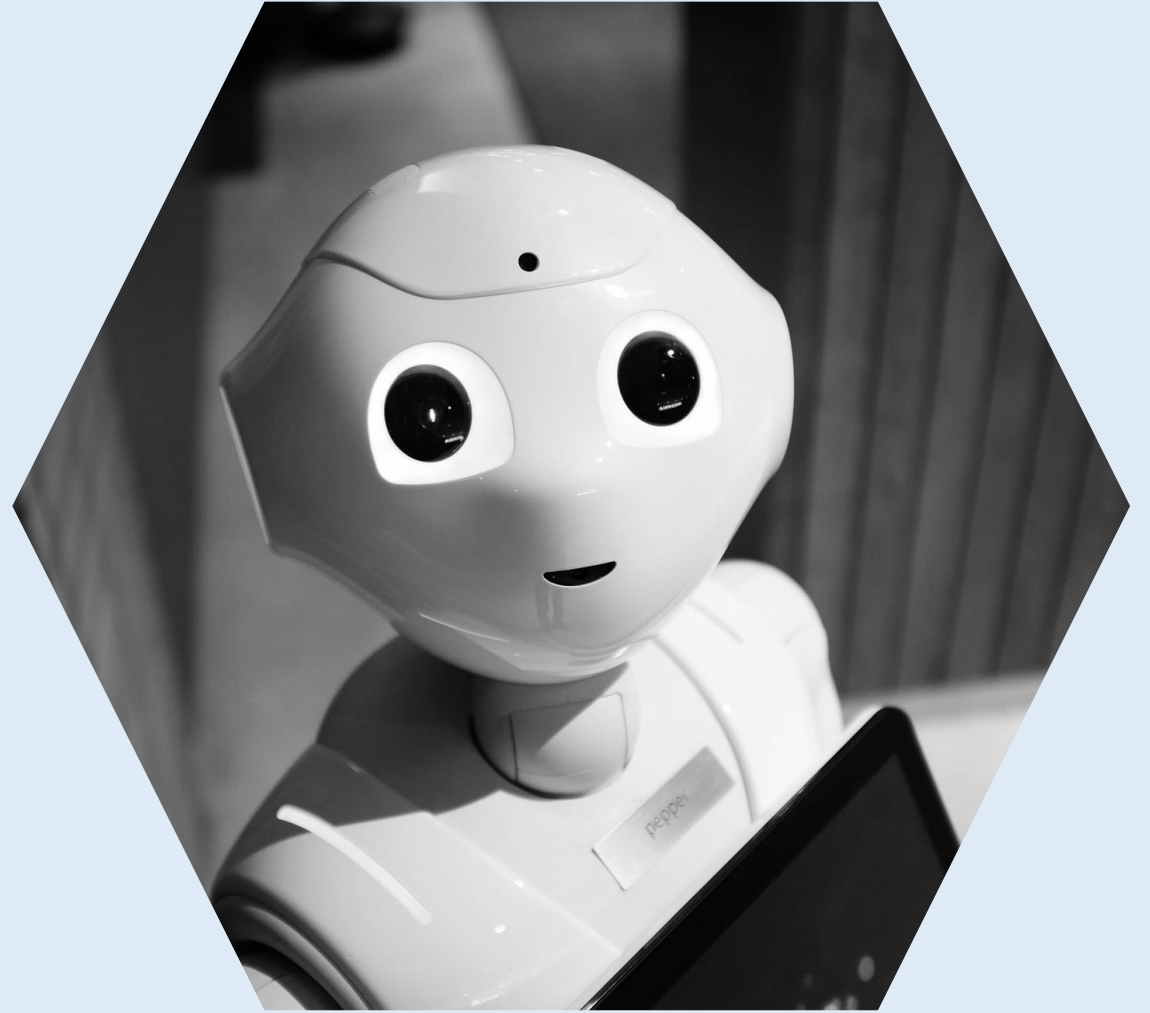


Alison Doig



Ailsa Land

Năm 1965, Dakin đã hoàn thiện phương pháp nhánh cận và nó trở thành phương pháp tối ưu rõ rệt so với các phương pháp trước đây





Tên “Branch And Bound ” xuất hiện lần đầu tiên trong công trình của Little *et al.* về vấn đề *nhân viên bán hàng du lịch*



Which is the fastest way?





## NOWADAY

Be using for many  
application of the  
question that how to  
solve combinatorics

01

Bài toán tìm đường đi của người giao hàng



02

Bài toán phân công lao động



03

Bài toán cái balo



04

Bài toán tam giác số



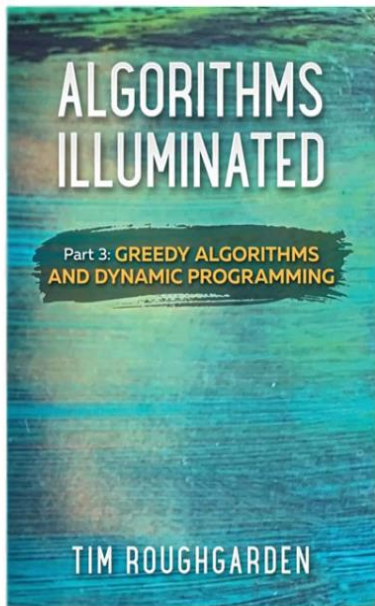
05

Bài toán người du lịch

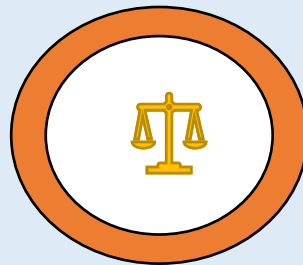




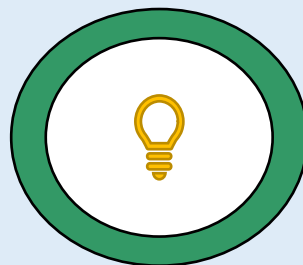
# Thuật toán nhánh cận là gì?



Mô hình giải quyết các bài toán tổ hợp tối ưu và rời rạc



Hệ thống các giải pháp ứng viên bằng phương pháp tìm kiếm thông tin trạng thái



Khám phá các nhánh của cây, đại diện cho các tập con của tập giải pháp

## II. Phân tích kĩ thuật nhánh cận






Là bài toán tìm ra tổ hợp tốt nhất trong  
những tổ hợp có thể tạo ra, thỏa mãn  
yêu cầu cho trước






# Bài toán

Giả sử bài toán yêu cầu tìm phương án  $X=(x_1, x_2, \dots, x_k, \dots)$  thỏa mãn những điều kiện nào đó và phương án này là tốt nhất theo tiêu chí cụ thể nào đó.



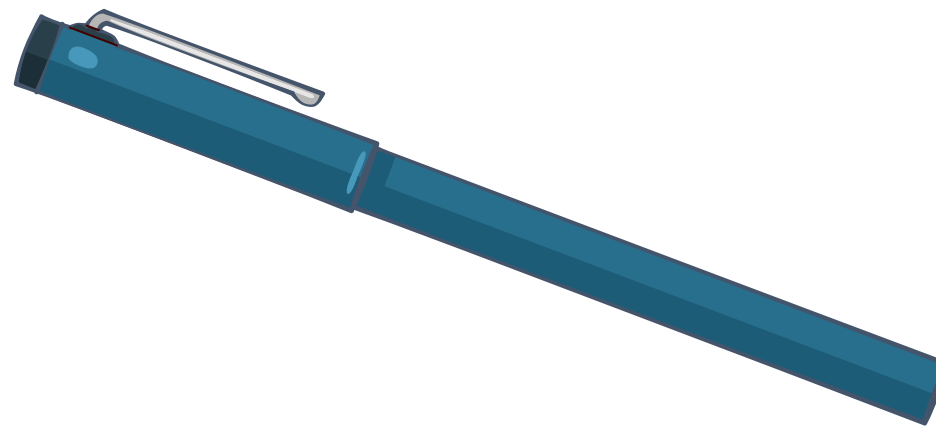
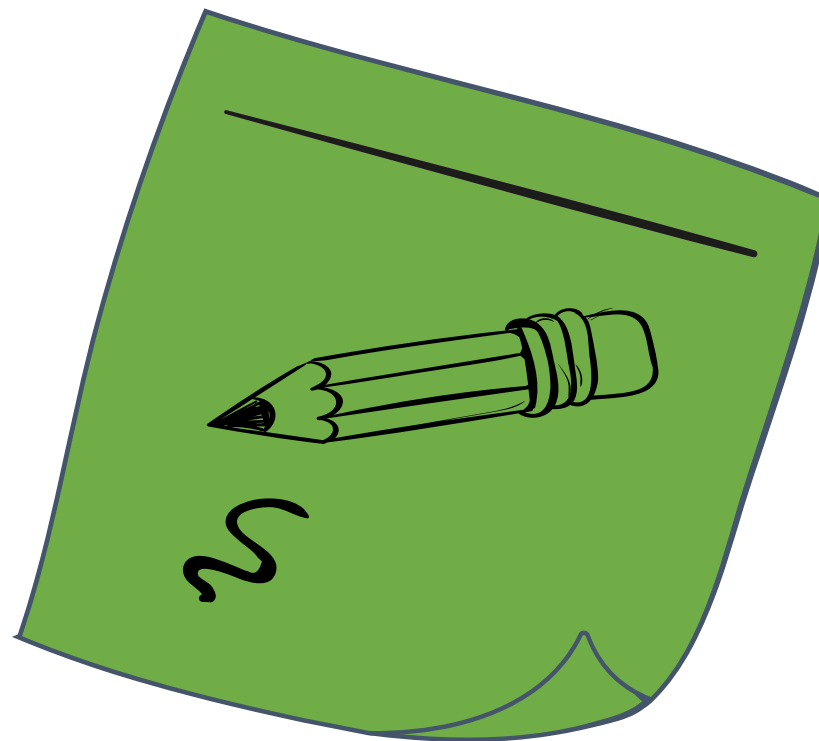
Hay bài toán được biểu diễn dưới dạng tổng quát như sau:

- Cho tập hữu hạn phần tử  $D$
  - Hàm mục tiêu  $f(x)$  được xác định trên  $D$
  - Mỗi phần tử  $x$  thuộc  $D$  có dạng  $X(x_1, x_2, x_3, \dots, x_k)$  được gọi là 1 phương án
  - Yêu cầu: Tìm phương án  $x_0$  sao cho  $f(x_0)$  đạt giá trị cực đại (hoặc cực tiểu) trên  $D$ .
- 

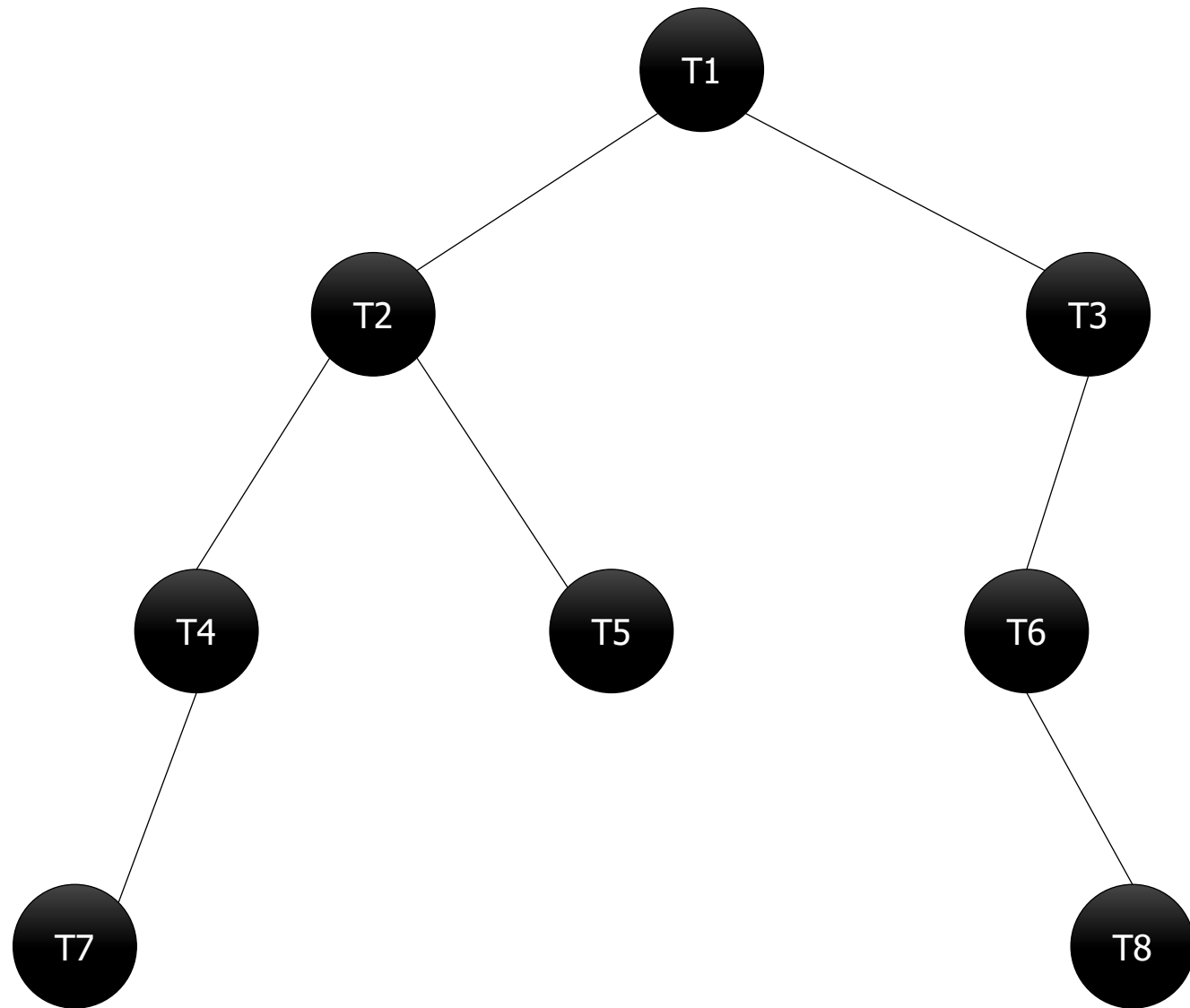


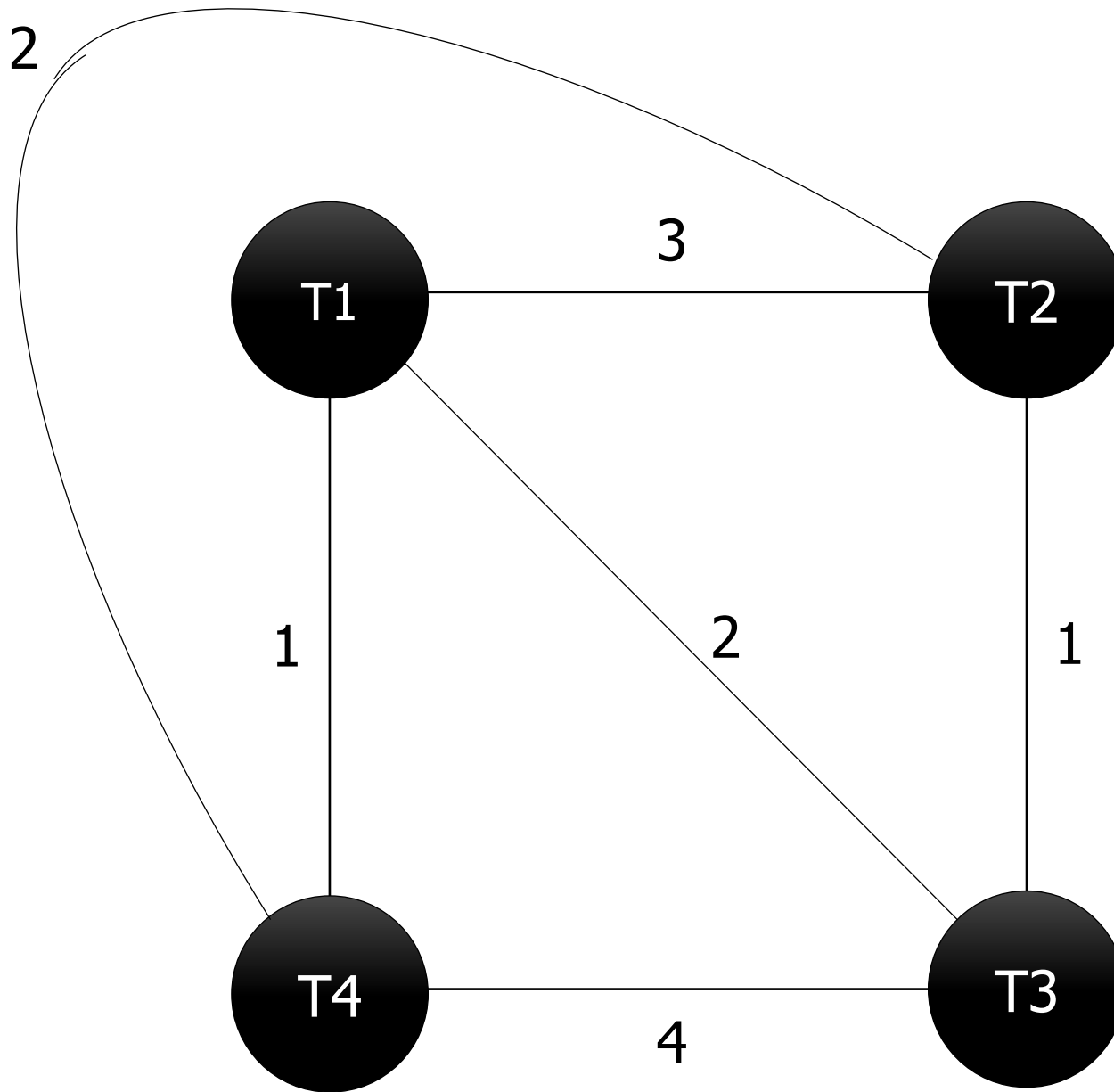
# Ý tưởng cơ bản

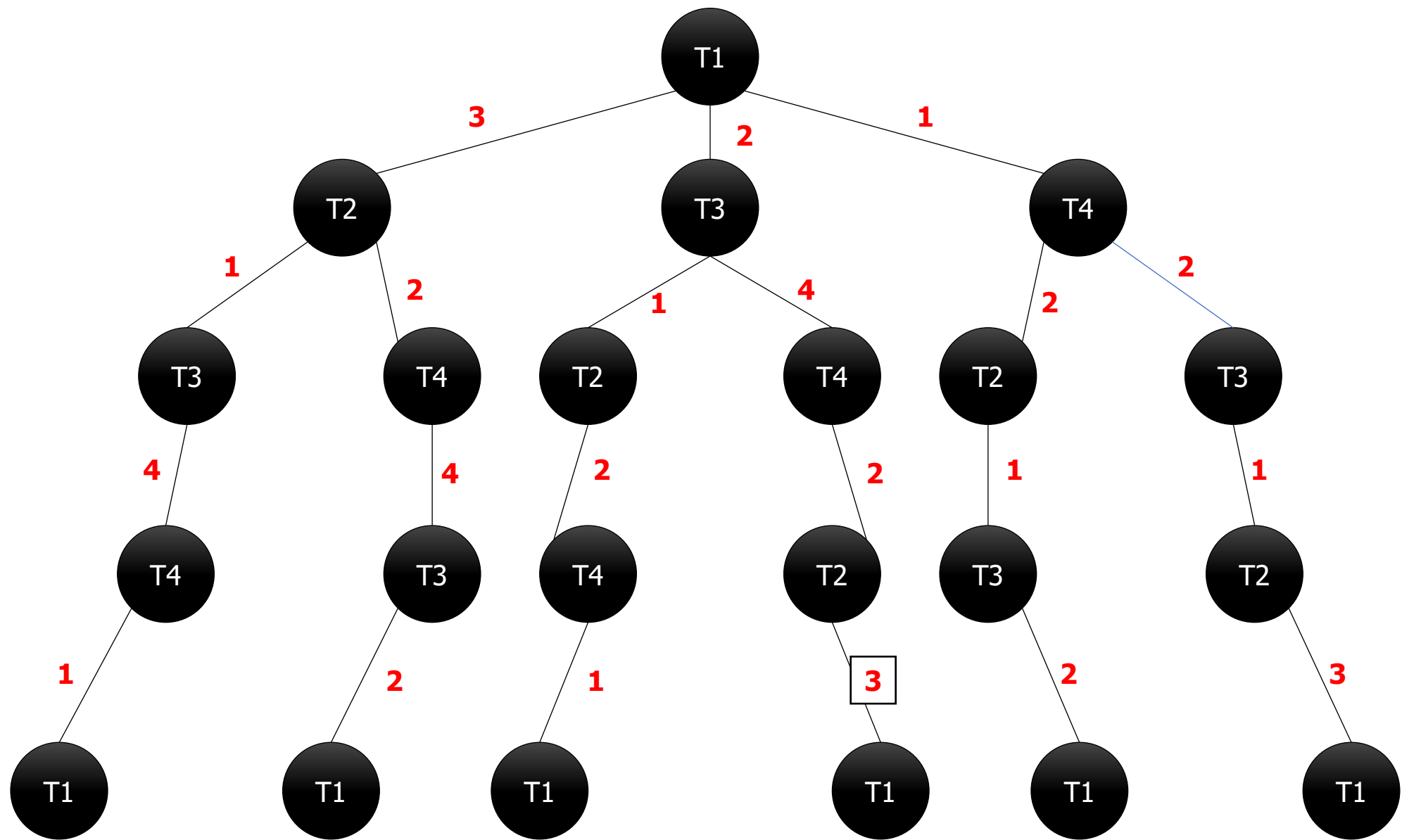
Kỹ thuật nhánh cận phát triển từ ý tưởng của thuật toán quay lui và thuật toán ràng buộc bao gồm việc liệt kê từng bước các giải pháp



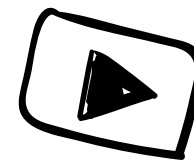




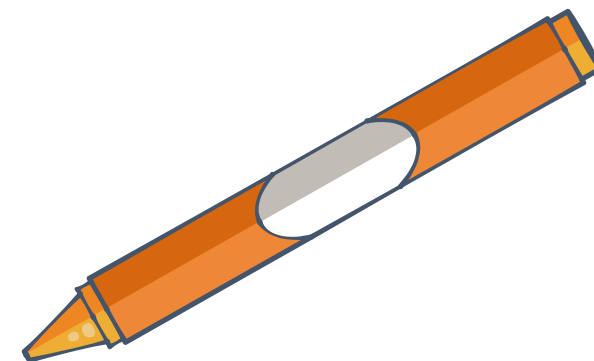


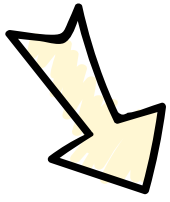






# Cách làm tổng quát





# Cách làm tổng quát

- Xét phương án hoàn chỉnh  $X = (x_1, \dots, x_k, \dots, x_n)$  và  $A = (a_1, \dots, a_k)$  với  $x_i = a_i$ , gọi là phương án bộ phận cấp  $k$  ( $k \leq n$ ).
- Giả sử xây dựng được hàm  $G$  (hàm tính cận trên/dưới) như sau: với mọi phương án bộ phận  $(a_1, \dots, a_k)$  và với mọi  $k=1, \dots, n$ .

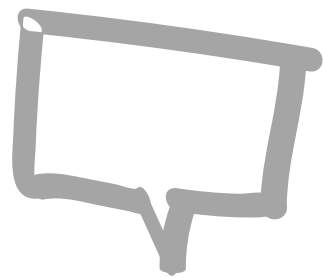
$$G(a_1, \dots, a_k) \leq \min \{f(x) \mid x \in D, x_i = a_i\}.$$

- Gọi  $\bar{x}$  là phương án hoàn chỉnh tốt nhất hiện có và  $\bar{f} = f(\bar{x})$  gọi là kỷ lục hiện tại. Nếu 1 phương án bộ phận  $a = (a_1, \dots, a_k)$  nào đó mà  $G(a_1, \dots, a_k) > \bar{f}$

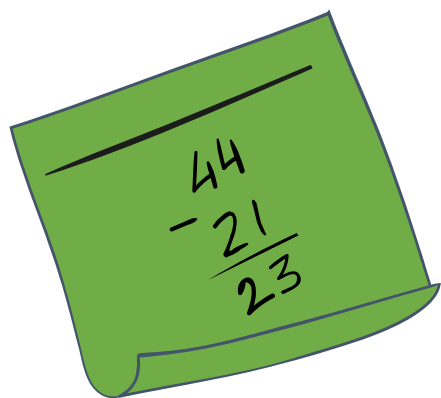
$$\Rightarrow f < G(a_1, \dots, a_k) \leq \min \{f(x) \mid x \in D, x_i = a_i\}.$$

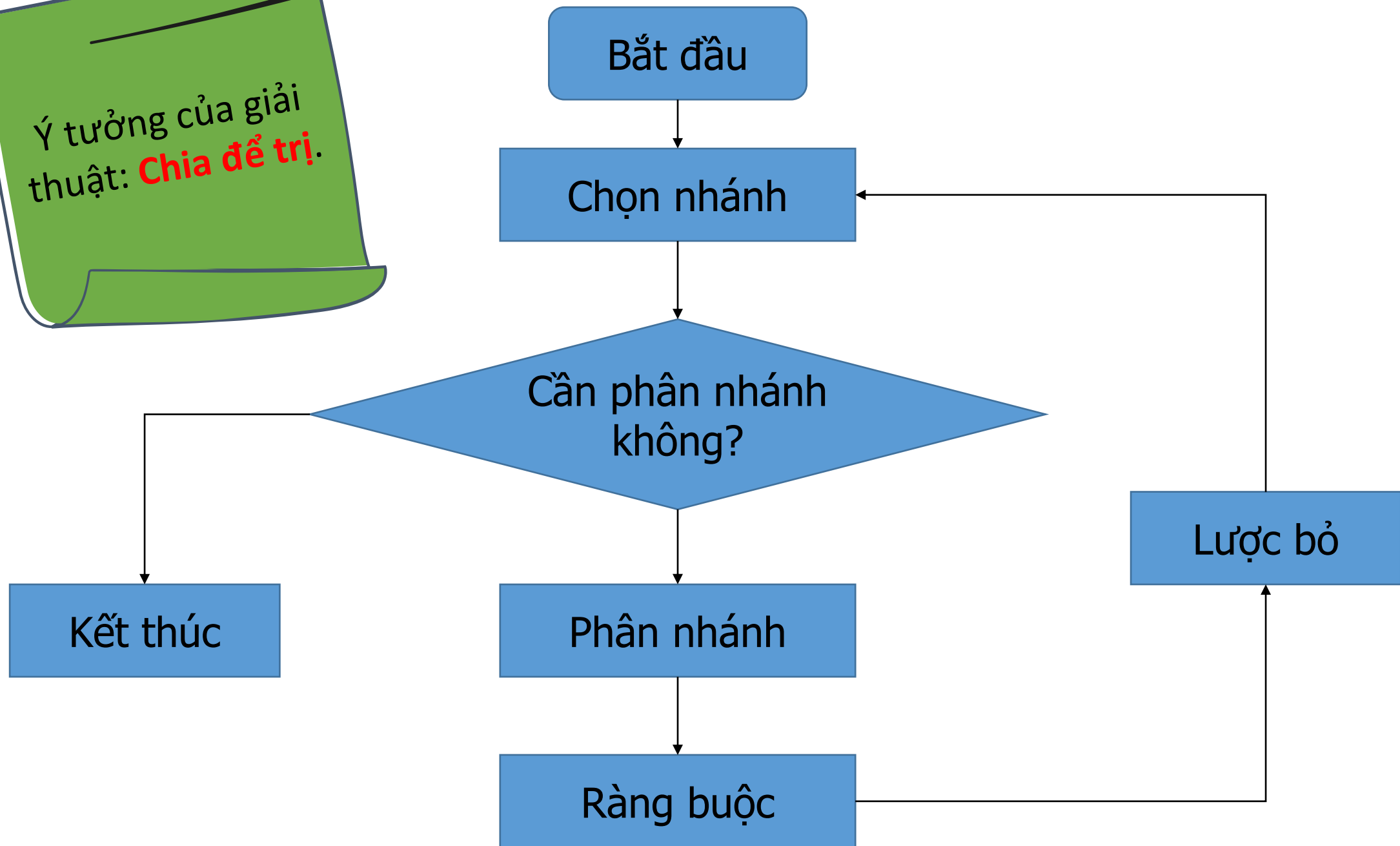
$\Rightarrow$  Các phương án hoàn chỉnh  $x = (x_1, \dots, x_n)$  phát triển từ phương án bộ phận  $a = (a_1, \dots, a_k)$  đều có hàm mục tiêu  $f(x) > \text{kỷ lục hiện tại } \bar{f}$ , nên ta loại bỏ tất cả các phương án hoàn chỉnh  $x$  phát triển từ phương án bộ phận  $a$ .





# Sơ đồ tổng quát của giải thuật







```
1  Procedure Init;  
2  begin  
3      khởi tạo 1 cấu hình bất kỳ BESTCONFIG>;  
4  end;  
5  {Thủ tục này thử chọn cho  $x_1$  tất cả các giá trị nó có thể nhận}  
6  Procedure Try (i: Integer);  
7  Begin  
8      For (Mọi giá trị V có thể gán cho  $x_1$ ) do  
9          Begin  
10             <Thử cho  $x_1 := V$ >;  
11             If (Việc thử trên vẫn còn hy vọng tìm ra cấu hình tốt hơn BESTCONFIG) then  
12                 If ( $x_1$  là phần tử cuối cùng trong cấu hình) then  
13                     <Cập nhật BESTCONFIG>  
14                 Else  
15                     Begin  
16                         <ghi nhận việc thử  $x_1 = V$  nếu cần>;  
17                         Try(i+1); //Gọi đệ quy, chọn tiếp  $x_{i+1}$ .  
18                         Bỏ ghi nhận việc thử cho  $x_1 = V$  (nếu cần)>;  
19                     End;  
20             End;  
21 End;  
22 Begin  
23     Init;  
24     Try(1);  
25     <Thông báo cấu hình tối ưu BESTCONFIG>  
26 End.
```

# III.APPLICATION

Nhóm 7-20CTT4



Bài toán tìm đường đi  
của người giao hàng

01



Bài toán phân công lao  
động

02



03

7  
5 6  
8 9 7  
5 1 7 6  
8 4 5 3 7

Bài toán tam giác số



Bài toán cái balo

04

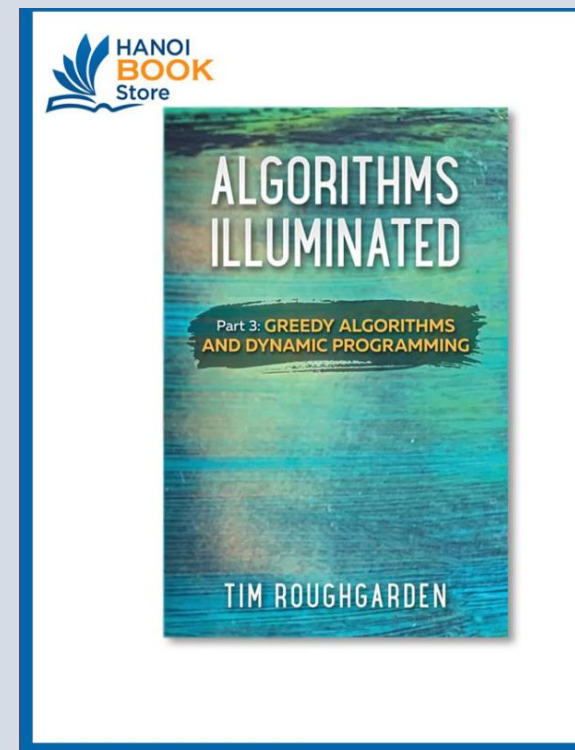
# Bài toán cái Balo





# Bài toán cái Balô:

Cho một cái ba lô có thể đựng một trọng lượng  $W$  và  $n$  loại đồ vật, mỗi đồ vật  $i$  có một trọng lượng  $g_i$  và một giá trị  $v_i$ . Tất cả các loại đồ vật đều có số lượng không hạn chế. Tìm một cách lựa chọn các đồ vật đựng vào ba lô, chọn các loại đồ vật nào, mỗi loại lấy bao nhiêu sao cho tổng trọng lượng không vượt quá  $W$  và tổng giá trị là lớn nhất



# BT cái ba lô – Ví dụ:

Ta có một ba lô có trọng lượng là 37 và 4 loại đồ vật với trọng lượng và giá trị tương ứng được cho trong bảng bên dưới:

Đồ vật	Trọng lượng	Giá trị
A	15	30
B	10	25
C	2	2
D	4	6

Đồ vật	Trọng lượng	Giá trị	Đơn giá
B	10	25	2.5
A	15	30	2
D	4	6	1.5
C	2	2	1.0

Đồ vật	Trọng lượng	Giá trị	Đơn giá
B	10	25	2.5
A	15	30	2
D	4	6	1.5
C	2	2	1.0

$$CT = 37 * 2.5 = 92.5$$

$$TGT = 0$$

$$W = 37, CT = 92.5$$

A

$$x_B = 3$$

$$x_B = 2$$

$$x_B = 1$$

$$x_B = 0$$

$$TGT = 75$$

$$W = 7$$

$$CT = 89$$

B

$$TGT = 50$$

$$W = 17$$

$$CT = 84$$

C

$$TGT = 25$$

$$W = 27$$

$$CT = 79$$

D

$$TGT = 0$$

$$W = 37$$

$$CT = 74$$

E

$$x_A = 0$$

$$x_A = 1$$

$$x_A = 0$$

$$TGT = 75$$

$$W = 7$$

$$CT = 85.5$$

F

$$TGT = 80$$

$$W = 2$$

$$CT = 83$$

K

$$TGT = 50$$

$$W = 17$$

$$CT = 75.25$$

L

$$x_D = 1$$

$$x_D = 0$$

$$TGT = 81$$

$$W = 3$$

$$CT = 84$$

G

$$TGT = 75$$

$$W = 7$$

$$CT = 82$$

H

$$x_C = 1$$

$$x_C = 0$$

$$TGT = 83$$

$$W = 1$$

I

$$TGT = 81$$

$$W = 3$$

J

Cắt tĩa

$X1 = (0, 3, 1, 1)$ ;  $TGT = 83$   
 $X2 = (0, 3, 1, 0)$ ;  $TGT = 81$   
 Phương án TNTT: X1  
 Giá LNTT = 83

Phương án tối ưu:  
 $X(0, 3, 1, 1)$ ;  $TGT: 83$

INPUT	OUTPUT
4 A 15 30 B 10 25 C 2 2 D 4 6 37	83 A: 0 B: 3 C: 1 D: 1

# MÃ GIẢI

```
#include<iostream>
using namespace std;
struct Object {
    int Weight;
    int value;
    float Key;
    char Type;
};
int MAX_weight = INT_MIN;
int Result[50];
//Merge Sort
void Merge(Object arr[], int mid, int left, int right) { ... }
void MergeSort(Object arr[], int left, int right) { ... }
void Solved(int i, int n, int weight, int TGT, Object a[], int Temp[])
{
    if (i == n || weight == 0)
    {
        MAX_weight = TGT;
        for (int z = 0; z < i; z++) {
            Result[a[z].Type - 65] = Temp[a[z].Type - 65];
        }
        return;
    }
    int max_numobject = weight / a[i].Weight;
    for (int j = max_numobject; j >= 0; j--)
    {
        TGT += j * a[i].value;
        weight -= j * a[i].Weight;
        float upper_bound = TGT + a[i + 1].Key * weight;
        if (upper_bound > MAX_weight)
        {
            Temp[a[i].Type - 65] = j;
            Solved(i + 1, n, weight, TGT, a, Temp);
        }
        TGT -= j * a[i].value;
        weight += j * a[i].Weight;
    }
}
```

```
{
    Temp[a[i].Type - 65] = j;
    Solved(i + 1, n, weight, TGT, a, Temp);
}
TGT -= j * a[i].value;
weight += j * a[i].Weight;
}

int main()
{
    int n;
    cin >> n;
    Object* a = new Object[n+1];
    char x;
    int y, z;
    for (int i = 0; i < n; i++)
    {
        cin >> x >> y >> z;
        a[i].Type = x;
        a[i].Weight = y;
        a[i].value = z;
        a[i].Key = float(z) / y;
    }
    a[n].Key = 0; //Xét với đồ vật cuối.
    MergeSort(a, 0, n - 1); // Sắp xếp theo đơn giá tăng dần với đơn giá = giá trị / trọng lượng
    int Weight;
    cin >> Weight;
    int* Temp = new int[n];
    Solved(0, n, Weight, 0, a, Temp);
    cout << "Value max: " << MAX_weight << endl;
    for (int i = 0; i < n; i++)
    {
        cout << char(i+65) << ": " << Result[i] << "\n";
    }
}
```



# KẾT QUẢ

```
Microsoft Visual Studio Debug Console
4
A 15 30
B 10 25
C 2 2
D 4 6
37
Value max: 83
A: 0
B: 3
C: 1
D: 1

C:\Users\DELL\OneDrive - VNU-HCMUS\Desktop\Project1\Debug\Project2.exe (process 1584) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

# Time Complexity

---

Mức độ phức tạp trong trường hợp xấu nhất của Branch và Bound vẫn giống như của giải thuật Vết cạn  $O(n!)$  vì trong trường hợp xấu nhất, chúng ta có thể không bao giờ có cơ hội cắt bớt một nút. Trong khi đó, trên thực tế, nó hoạt động rất tốt tùy thuộc vào từng trường hợp khác nhau của TSP. Độ phức tạp cũng phụ thuộc vào việc lựa chọn hàm giới hạn vì chúng là hàm quyết định có bao nhiêu nút được cắt bỏ.

# TÀI LIỆU THAM KHẢO

---

- Sách giải thuật và lập trình : Thầy LÊ MINH HOÀNG – Đại học sư phạm Hà Nội, 1999-2002
- [Geeksforgeeks.org](https://www.geeksforgeeks.org/): Web học tập lập trình.