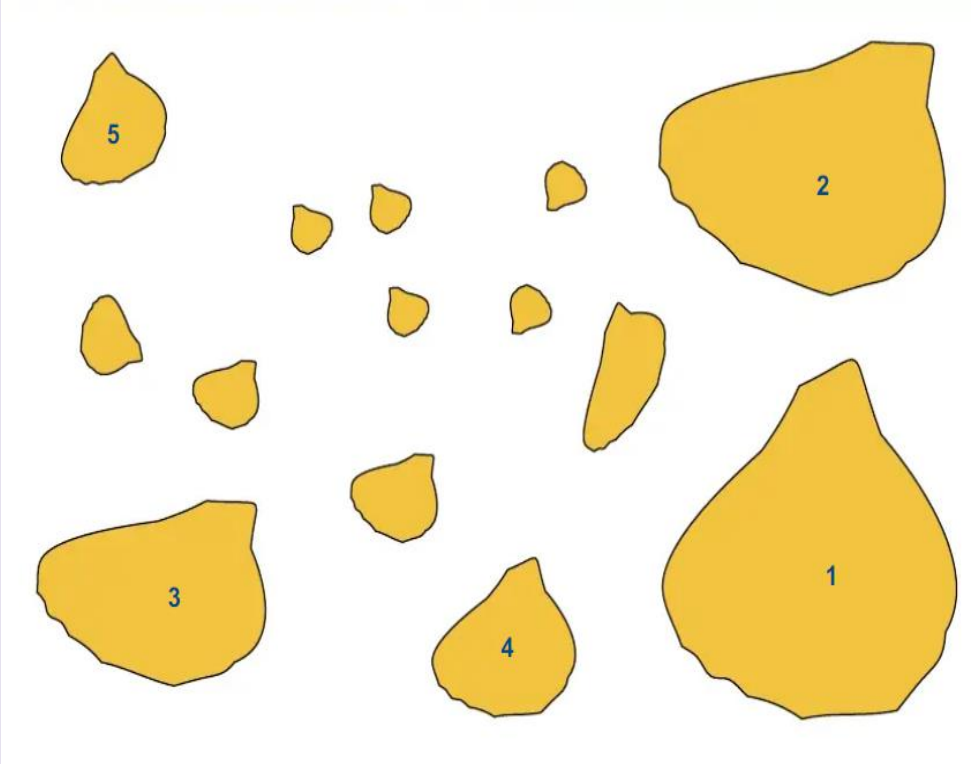
Phương pháp thiết kế thuật toán:

Greedy Algorithm

Nhóm 7: Nguyễn Vĩnh Hưng

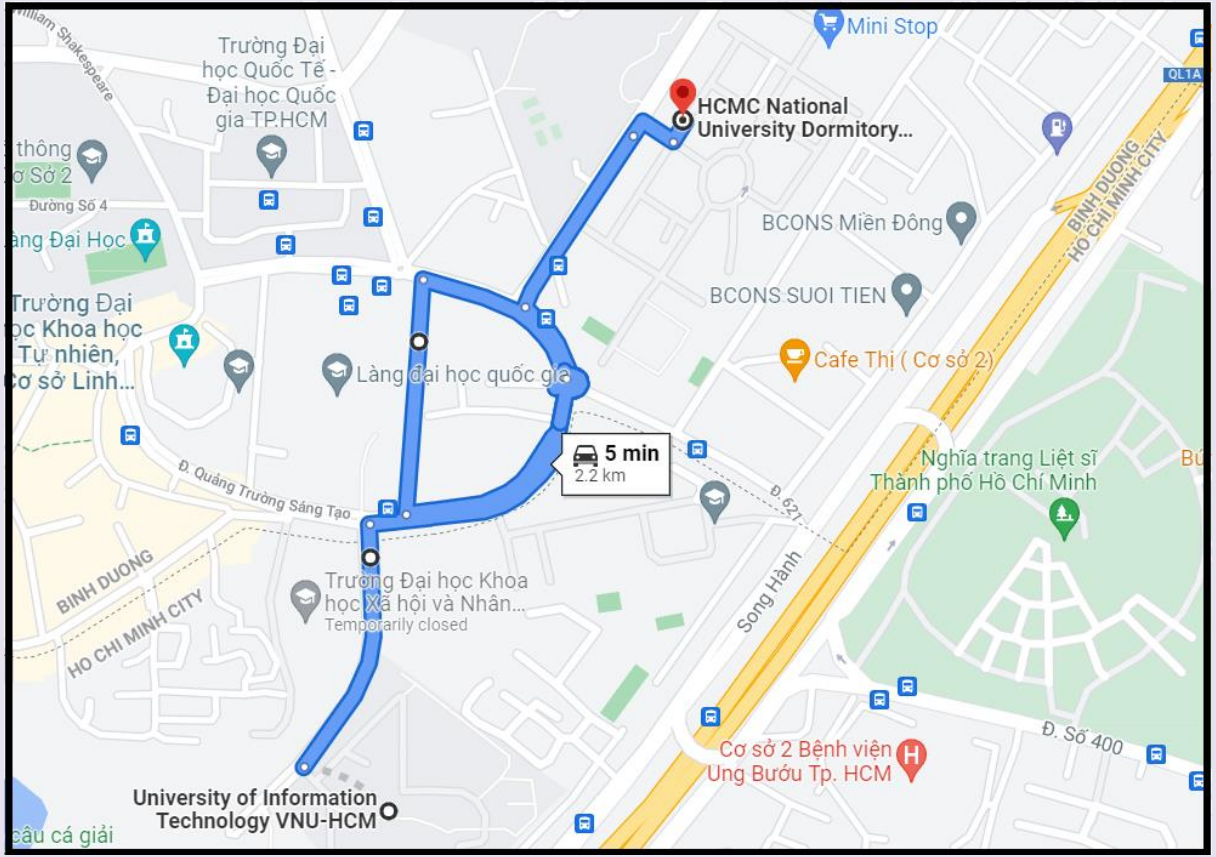
Giới thiệu

Greedy Algorithm

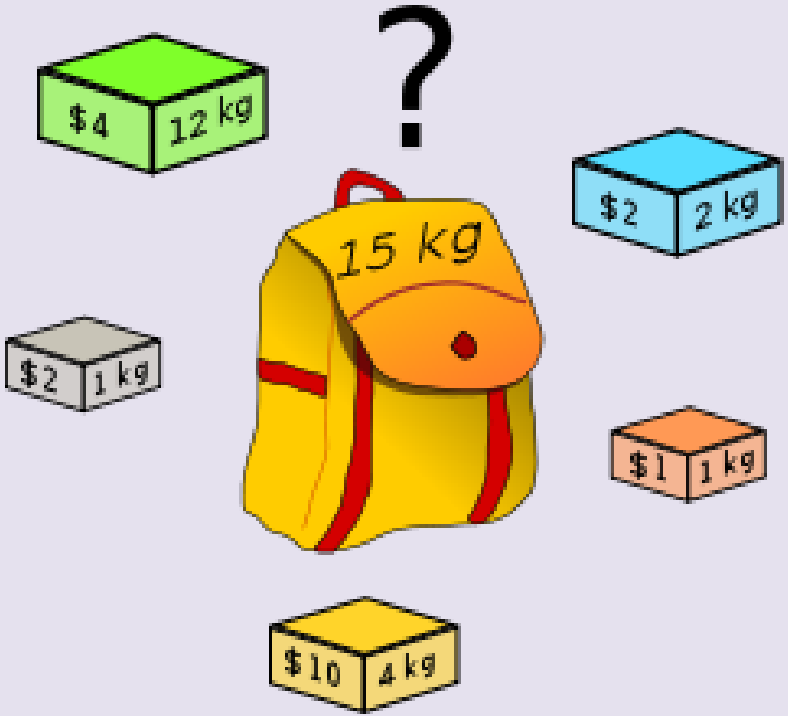


Có rất nhiều khối vàng với các hình dáng khác nhau, nếu bạn được chọn 1 khối vàng duy nhất bạn sẽ chọn khối nào?

Greedy Algorithm

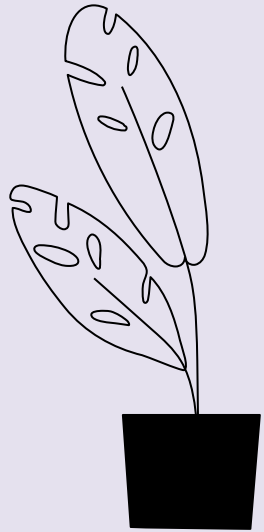


Greedy Algorithm



Balo có thể chưa tối đa 15kg, có 5 loại hộp với giá trị khác nhau, không giới hạn số lượng. Tìm các đặt vào balo các hộp để được giá trị tối ưu theo trực giác?

-> bạn nữ

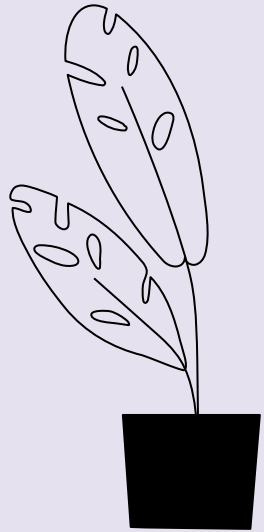


02

Khái niệm

Greedy Algorithm

Giải thuật tham lam (Greedy Algorithm) là giải thuật tối ưu hóa tổ hợp. Giải thuật tìm kiếm, lựa chọn giải pháp tối ưu địa phương ở mỗi bước với hi vọng tìm được giải pháp tối ưu toàn cục.



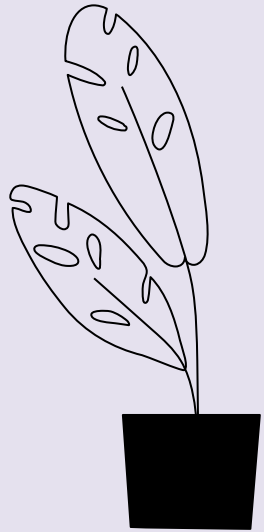
Greedy Algorithm

Giải thuật tham lam (Greedy Algorithm) là giải thuật tối ưu hóa tổ hợp. Giải thuật tìm kiếm, lựa chọn giải pháp tối ưu địa phương ở mỗi bước với hi vọng tìm được giải pháp tối ưu toàn cục.

PHƯƠNG PHÁP THAM LAM:

- Sắp xếp dữ liệu tăng dần hay giảm dần theo các tiêu chí đưa ra
- Khởi tạo tập giải pháp (câu trả lời) là rỗng.
- Tại mỗi bước:

- Chúng ta luôn chọn giá trị tốt nhất tại thời điểm đó mà không quan tâm đến tương lai (tối ưu cục bộ)
- Chúng ta hy vọng việc chọn tối ưu cục bộ tại mỗi bước sẽ cho tối ưu toàn cục



Greedy Algorithm

TẠI SAO THUẬT TOÁN THAM LAM LẠI ĐƯỢC GỌI LÀ THAM LAM?

Lược đồ chung

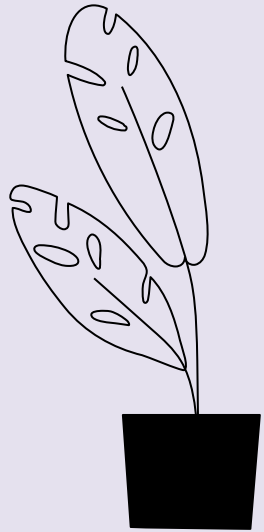
```
procedure Greedy;
(* Giả sử C là tập các ứng cử viên *)
begin
   S: = ∅; (*  S lời giải xây dựng theo thuật toán *)
   while (C ≠ ∅) and not Solution(S) do
   begin
        x ← Select(C);
        C:= C \ x;
        if Feasible (S ∪ x) then S:= S ∪ x ;
   end;
   if Solution(S) then return S;
end;
```

2/2/2017

Greedy Algorithm

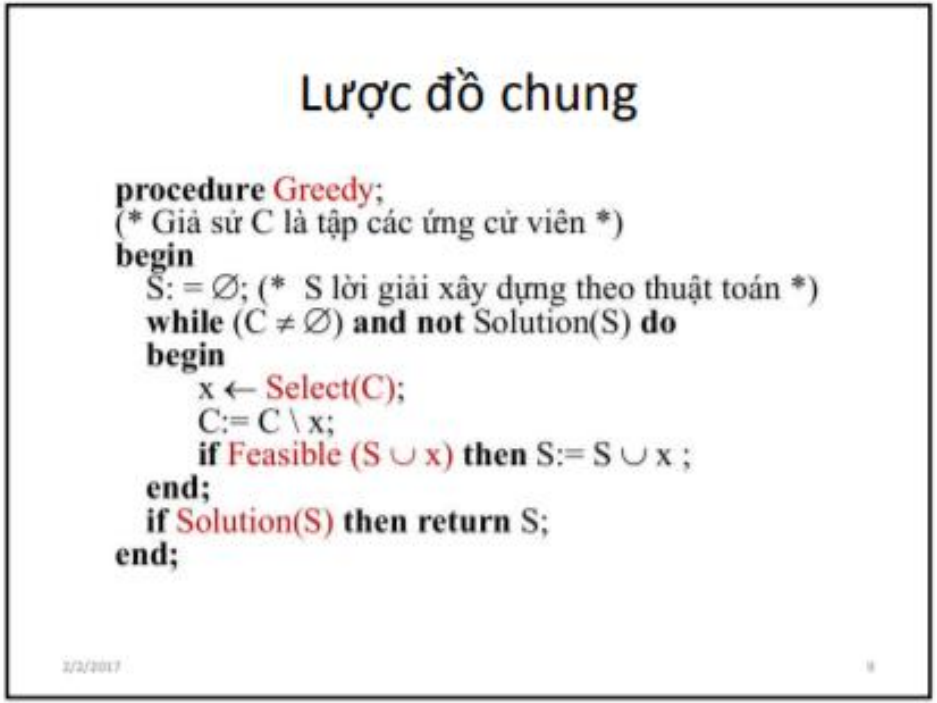
Nhận dạng bài toán Greedy:

- **Bài toán có thể phân thành bài toán con**
- **Tại mỗi bài toán con sẽ luôn chọn ứng viên tối ưu nhất**



Greedy Algorithm

Sử dụng:



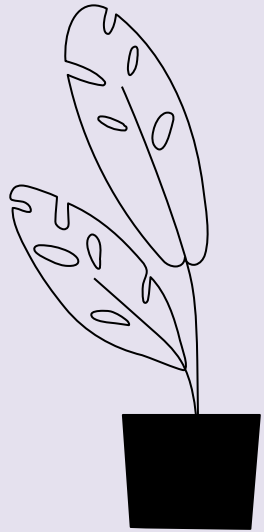
03

Ví dụ

Greedy Algorithm

Bài toán 1: Cho mảng a[n] gồm n phần tử, cho biết thời gian làm công việc i là a[i], hãy tìm số công việc tối đa có thể làm trong T đơn vị thời gian.
Input: Hàng đầu tiên là n và T
Hàng thứ 2 chứa mảng a.
Ví dụ:

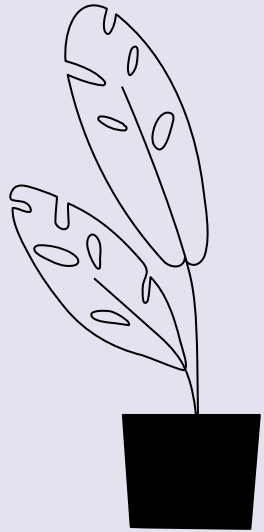
input	output
5 10 2 5 7 4 1	3 Giải thích: các công việc thứ 1,4,5 được thực hiện, a[1] + a[4] + a[5] = 2+4+1 = 7 <= 10



Greedy Algorithm

Các bước thực hiện:
currentTime: lưu tổng thời gian tại bước thứ i
numberOfThings: lưu tổng số công việc bước thứ I

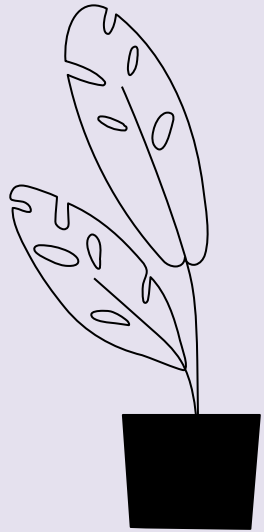
- Sắp xếp mảng **A** theo thứ tự không giảm.
- Chọn từng mục việc từ a[1] -> a[n].
- Thêm thời gian a[i] vào **currentTime** và tăng **numberofThings** lên 1 nếu hợp lệ .
- Thêm một vào **numberOfThings** .



Greedy Algorithm

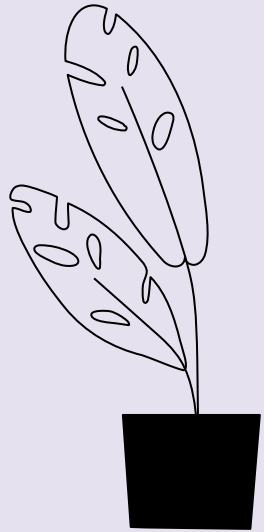
Các bước thực hiện:
currentTime: lưu tổng thời gian tại bước thứ i
numberOfThings: lưu tổng số công việc bước thứ I

- Sắp xếp mảng **A** theo thứ tự không giảm.
- Chọn từng mục việc từ a[1] -> a[n].
- Thêm thời gian a[i] vào **currentTime** và tăng **numberofThings** lên 1 nếu hợp lệ .
- Thêm một vào **numberOfThings** .



Greedy Algorithm

```
#input
n,T = map(int,input().split())
a = list(map(int, input().split()))
#sort
a.sort();
#solve
numberOfThings = 0;
currentTime = 0;
for x in a:
    if(numberOfThings + x <= T):
        numberOfThings = numberOfThings + x
        currentTime = currentTime + 1
print(currentTime);
```



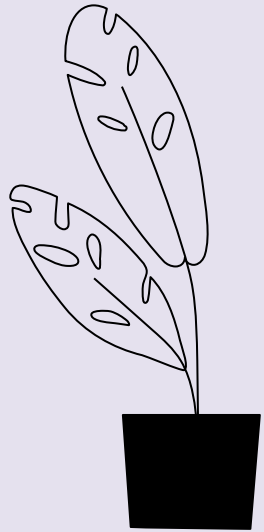
Greedy Algorithm

Bài toán 2: Hãy tưởng tượng bạn là một máy bán hàng tự động. Ai đó đưa cho bạn £1 và mua đồ uống với giá £0,70p. Không có đồng 30p bằng đồng bảng Anh , làm thế nào để bạn tính toán tiền lẻ trả lại, hãy tìm cách sử dụng ít nhất số đồng tiền?

Greedy Algorithm

Bài toán: Hãy tưởng tượng bạn là một máy bán hàng tự động. Ai đó đưa cho bạn £1 và mua đồ uống với giá £0,70p. Không có đồng 30p bằng đồng bảng Anh , làm thế nào để bạn tính toán tiền lẻ trả lại, hãy tìm cách sử dụng ít nhất số đồng tiền?

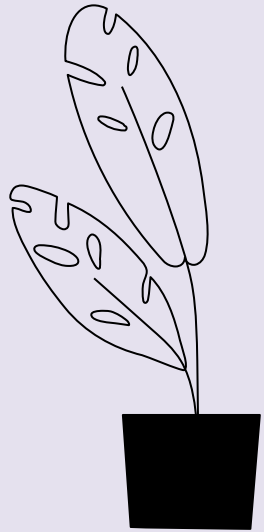
```
1   #input
2   T = int(input())
3   a = [1,2,5,10,20,50]
4   #sort
5   a.sort(reverse=True);
6   print(T,'=',end = " ")
7   #solve
8   sum = 0;
9   b = []
10  for x in a:
11      while(sum + x <= T):
12          sum = sum + x
13          b.append(x)
14  print(*b,sep = " + ");
15
```



Greedy Algorithm

Tham lam liệu có phải là tối ưu nhất?

Không



Greedy Algorithm

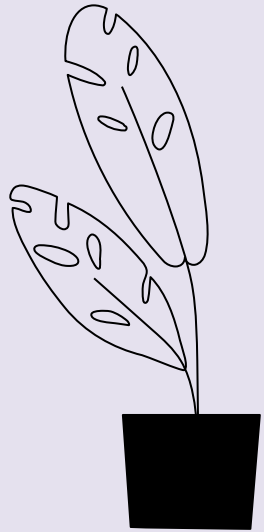
Bài toán 3: Hãy tưởng tượng bạn là một máy bán hàng tự động. Ai đó đưa cho bạn £1 và mua đồ uống với giá £0,70p. Không có đồng 30p bằng đồng bảng Anh , và chỉ có các loại đồng tiền 1p, 15p, 25p trong máy với số lượng không hạn chế.

Greedy Algorithm

Bài toán 3: Hãy tưởng tượng bạn là một máy bán hàng tự động. Ai đó đưa cho bạn £1 và mua đồ uống với giá £0,70p. Không có đồng 30p bằng đồng bảng Anh , và chỉ có các loại đồng tiền 1p, 15p, 25p trong máy với số lượng không hạn chế.

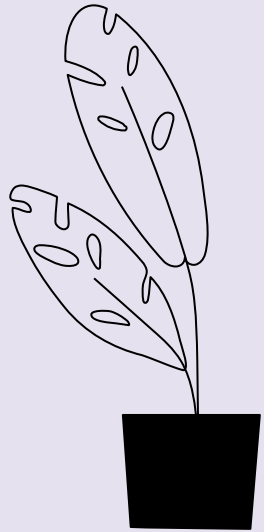
```
#input
T = int(input())
a = [1,15,25]
#sort
a.sort(reverse=True);
print(T,'=',end = " ")
#solve
sum = 0;
b = []
for x in a:
    while(sum + x <= T):
        sum = sum + x
        b.append(x)
print(*b,sep = " + ");

```



Greedy Algorithm

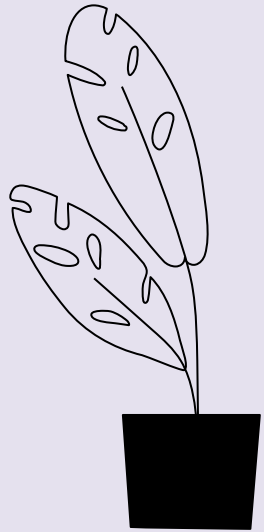
Thuật toán Prim’s có phải tham lam??



Greedy Algorithm

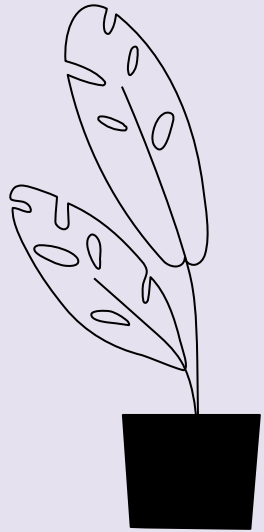
Thuật toán Prim’s có phải tham lam??

- Tạo một cây mới với một đỉnh duy nhất (được chọn ngẫu nhiên)
- Trong tất cả các cạnh chưa có trong cây mới, hãy tìm cạnh có trọng số tối thiểu và chuyển nó sang cây mới
- Lặp lại bước 2 cho đến khi tất cả các đỉnh nằm trong cây



Greedy Algorithm

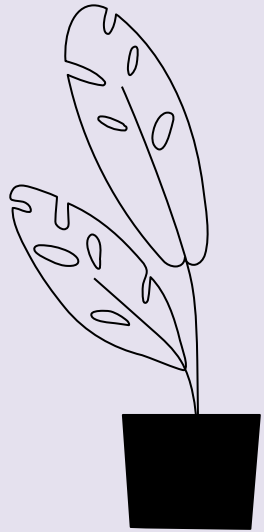
Thuật toán Dijkstra có phải tham lam



Greedy Algorithm

Thuật toán Dijkstra có phải tham lam

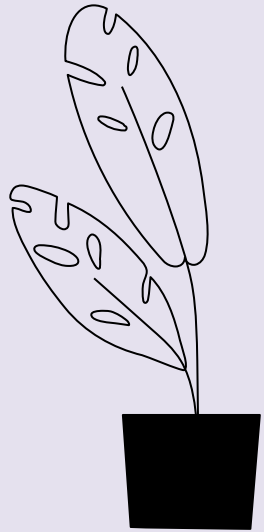
- Tạo một cây mới với một đỉnh duy nhất (được chọn ngẫu nhiên)
- Trong tất cả các cạnh chưa có trong cây mới, hãy tìm cạnh có trọng số tối thiểu và chuyển nó sang cây mới
- Lặp lại bước 2 cho đến khi tất cả các đỉnh nằm trong cây



Greedy Algorithm

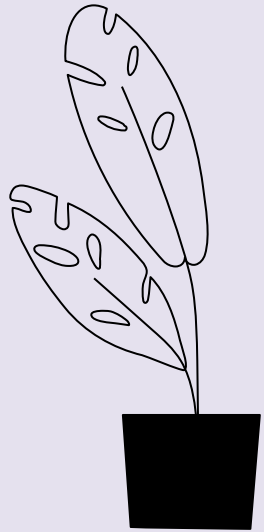
So sánh giữa Greedy, Divide và Dynamic Programming

Greedy	Divide	Dynamic Programming
Tối ưubằng cách đưa ra lựa chọn tốt nhất tại thời điểm hiện tại	Tối ưu hóa bằng cách chia nhỏ một vấn đề con thành các phiên bản đơn giản hơn và giải quyết	Tương tự Divide, nhưng lưu vào bộ nhớ đệm cho mỗi bài toán nhỏ để trả lời cho các bài toán lớn hơn
Không phải lúc nào cũng tìm ra giải pháp tối ưu, nhưng rất nhanh	Luôn tìm ra giải pháp tối ưu, nhưng chậm hơn Greedy	Luôn luôn tìm ra giải pháp tối ưu, nhưng có thể là vô nghĩa trên các tập dữ liệu nhỏ.
Yêu cầu gần như không tốn nhiều bộ nhớ	Yêu cầu khá nhiều bộ nhớ để ghi nhớ các cuộc gọi đệ quy	Yêu cầu nhiều bộ nhớ để ghi nhớ / lập mảng hoặc danh sách



Greedy Algorithm

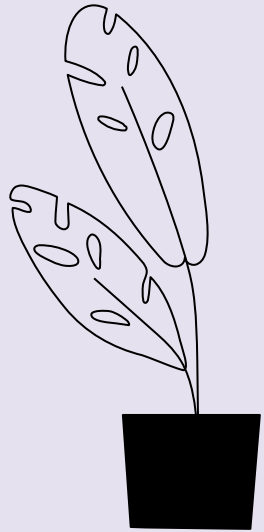
Ứng dụng thực tế:



Greedy Algorithm

Ứng dụng thực tế:

- **Áp dụng cho công việc: Xếp lịch công việc, phân chia công việc.**
- **Nén dữ liệu: Hash, Huffman,…**
- **Trả tiền lẻ**
- **Tìm đường đi**
- **Trong trò chơi: ví dụ cờ tướng, cờ vua**



Thank You