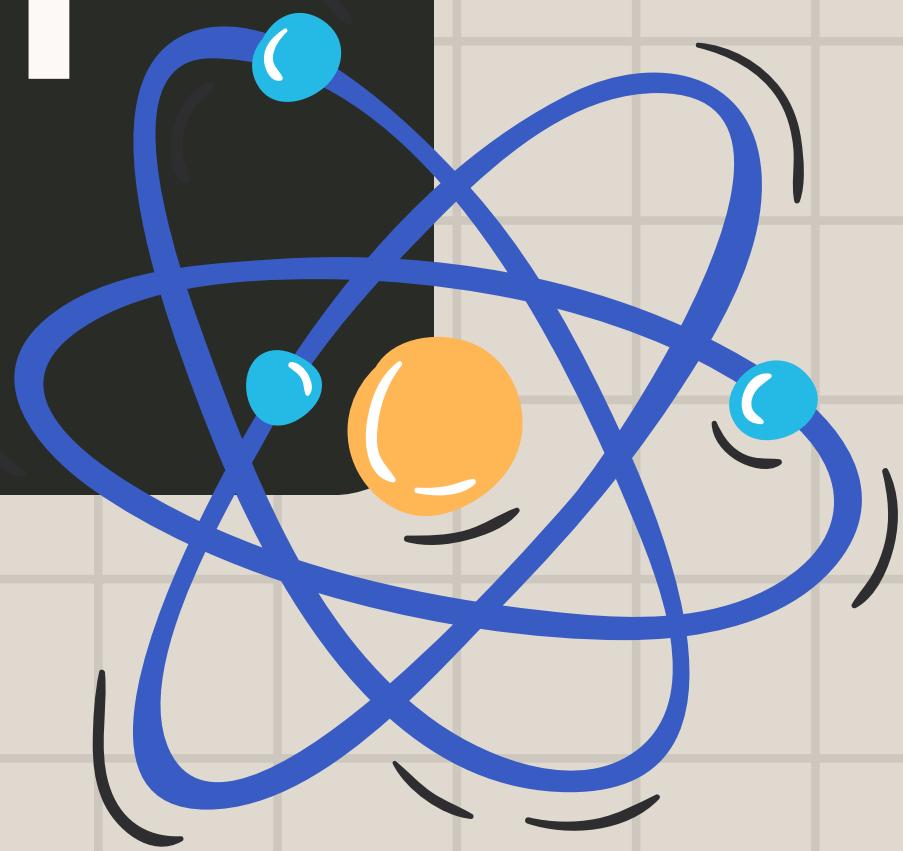


GREEDY approach

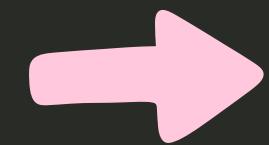


Đời Sống Thực Tế



Bạn là một đầu bếp trong nhà hàng, sau khi khách thanh toán, bạn phải trả lại tiền thừa sao cho đúng số tiền và dùng ít tờ nhất có thể. Bạn có sẵn các mệnh giá như 50.000, 20.000, 10.000, 5.000, 2.000 và 1.000 đồng. Hãy tìm cách kết hợp các tờ tiền và trả lại tiền một cách tốt nhất ?

Đời Sống Thực Tế



Luôn ưu tiên tờ tiền có giá trị lớn nhất còn phù hợp

Bắt đầu từ mệnh giá lớn nhất, chọn nhiều nhất có thể sao cho không vượt quá số tiền cần đổi, sau đó trừ đi phần đã chọn và chuyển sang mệnh giá nhỏ hơn. Lặp lại cho đến khi số tiền cần đổi bằng 0.

Đời Sống Thực Tế



Ví dụ:

Đổi 75,000đ

Giả sử có các mệnh giá: 50,000đ - 20,000đ - 10,000đ - 5,000đ - 2,000đ - 1,000đ

Đời Sống Thực Tế



Ví dụ:

Đổi 75,000đ

Giả sử có các mệnh giá: 50,000đ - 20,000đ - 10,000đ - 5,000đ - 2,000đ - 1,000đ

Thực hiện từ tờ lớn nhất:

Bước 1: 50,000đ

- $75,000 \div 50,000 = 1$ tờ (vì 2 tờ $= 100,000 > 75,000$)
- Còn lại: $75,000 - 50,000 = 25,000$ đ

Bước 2: 20,000đ

- $25,000 \div 20,000 = 1$ tờ
- Còn lại: $25,000 - 20,000 = 5,000$ đ

Bước 3: 10,000đ

- $5,000 \div 10,000 = 0$ tờ (vì $10,000 > 5,000$)
- Còn lại: vẫn là 5,000đ

Bước 4: 5,000đ

- $5,000 \div 5,000 = 1$ tờ
- Còn lại: $5,000 - 5,000 = 0$ đ

Tổng: 3 tờ tiền = 75,000đ ✓

Greedy approach

Cách tiếp cận tham lam là cách giải quyết vấn đề bằng cách đưa ra các lựa chọn nhằm tối ưu (cục bộ) ở mỗi bước để tìm ra giải pháp cho bài toán lớn.

Greedy approach

Vậy đặc điểm của các bài toán áp dụng được greedy approach là gì ?

Greedy approach



Chọn lựa

Không cần xem xét toàn cục,
và lựa chọn này dẫn đến
nghiệm tối ưu toàn cục.



Greedy approach

Tính chất nghiệm con tối ưu

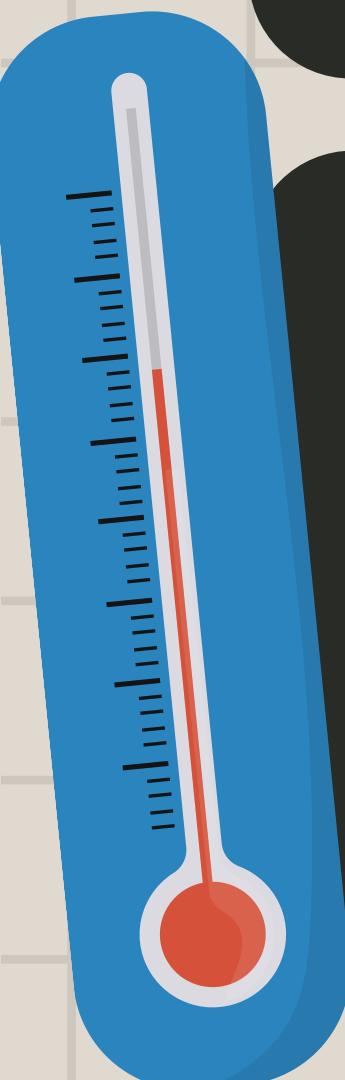
Nghiệm tối ưu của toàn bộ bài toán có thể được xây dựng từ nghiệm tối ưu của các bài toán con



Greedy approach

Không quay đầu

Greedy hoạt động tốt khi một lựa chọn được đưa ra sẽ không phải sửa lại.



Greedy approach

Áp dụng được CTDL



Greedy thường yêu cầu chọn phần tử “tốt nhất” nhiều lần
nên có cấu trúc dữ liệu hỗ trợ hiệu quả





Greedy approach



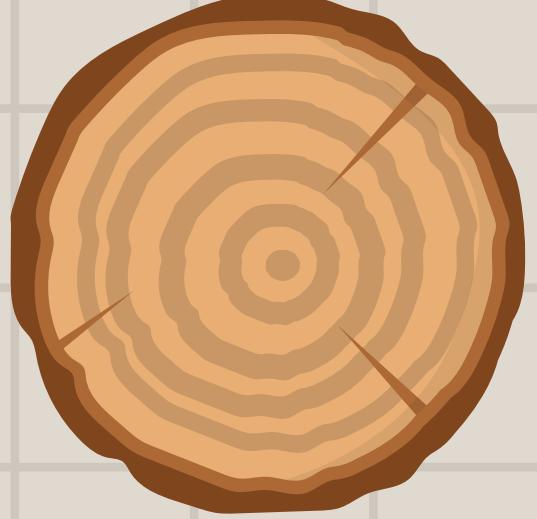
Tính lựa chọn tham lam

Việc lựa chọn tại một thời điểm chỉ phụ thuộc vào lựa chọn trước đó.



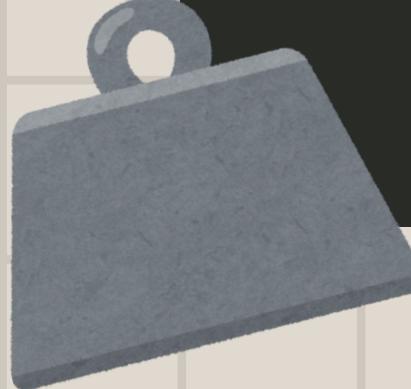


Greedy approach



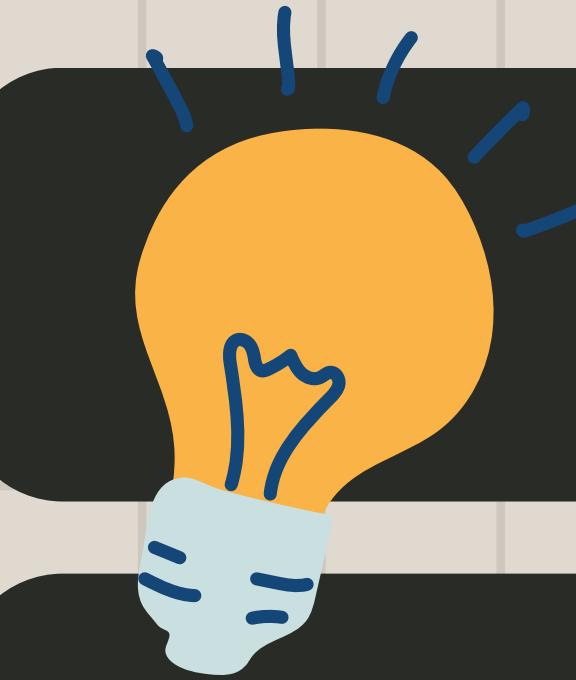
Phải có tiêu chí để đánh giá

Nếu không có thứ tự thì không thể tham lam được.

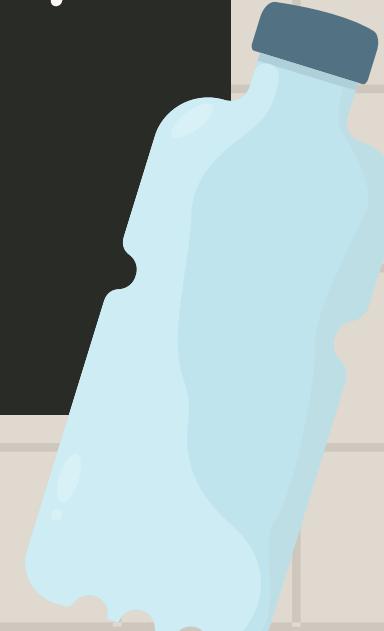


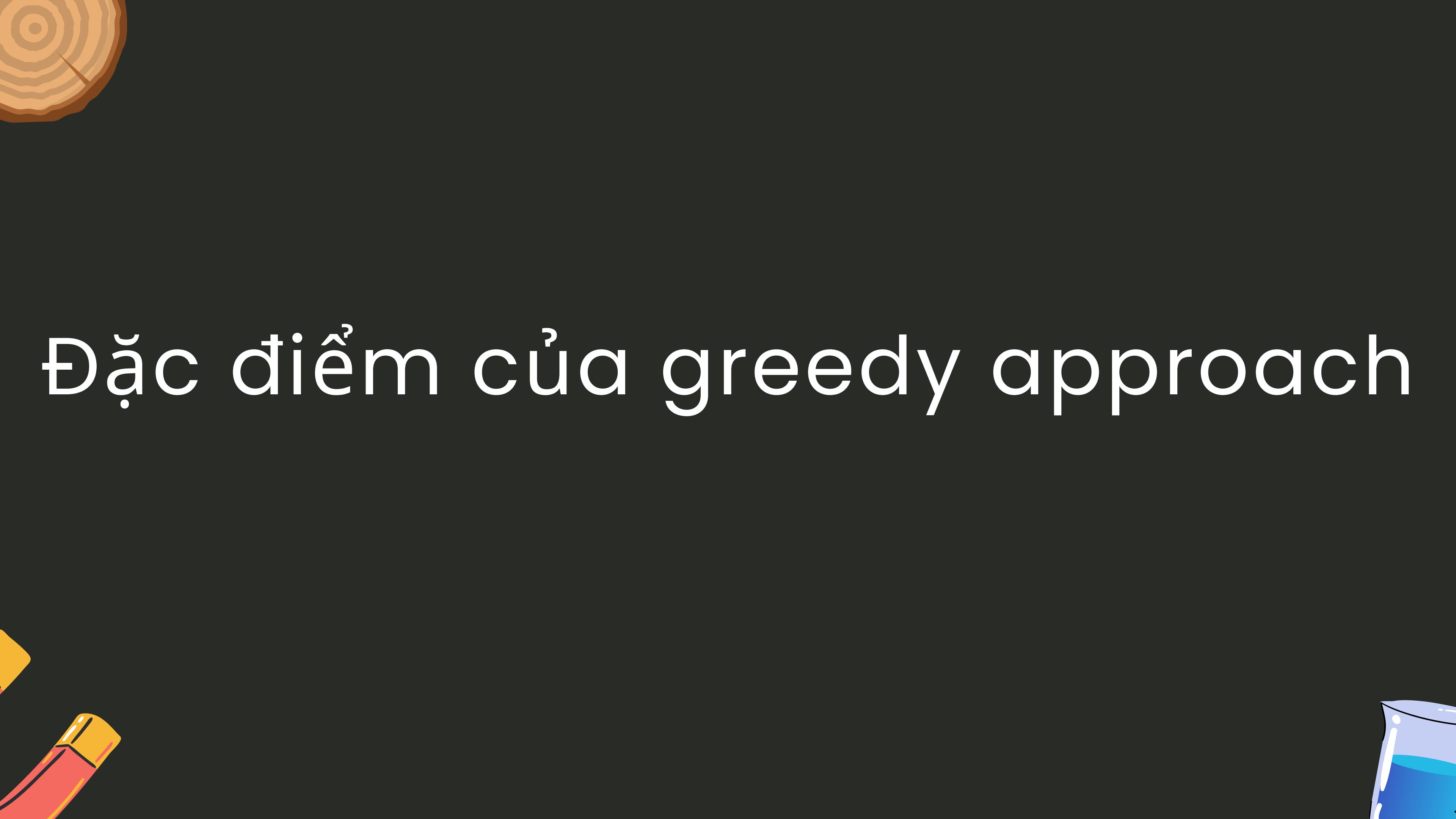
Greedy approach

Nhận xét



Từ khóa để nói về Greedy approach là local minimum(cực tiểu địa phương).





Đặc điểm của greedy approach

Greedy approach



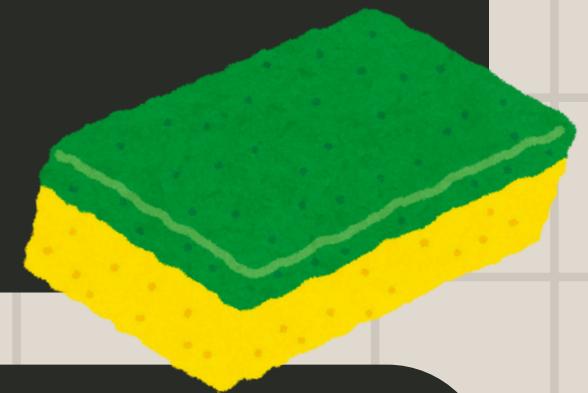
Đơn giản

Các thuật toán tham lam
thường đơn giản để thiết kế



Greedy approach

Hiệu quả



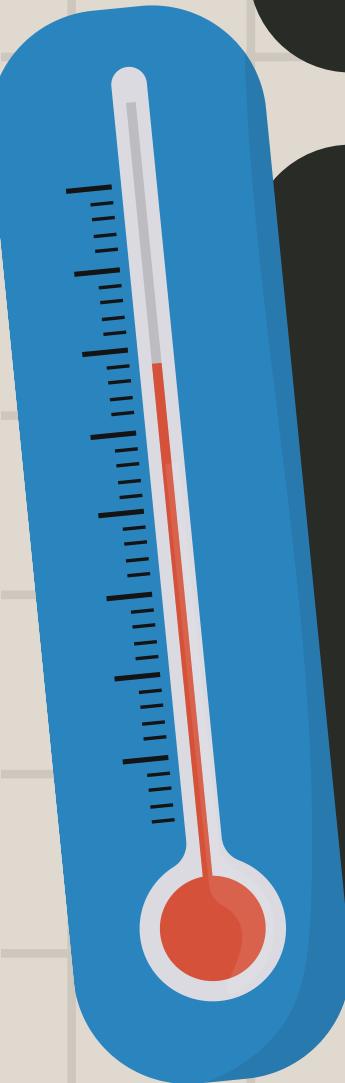
Các thuật toán tham lam thường hiệu quả về độ phức tạp
về thời gian



Greedy approach

Giải pháp khả thi

Mặc dù không phải lúc nào cũng tối ưu, nhưng một cách tiếp cận tham lam luôn tạo ra một giải pháp khả thi cho một vấn đề

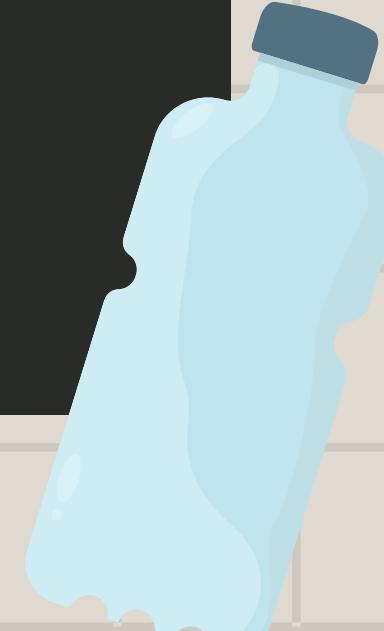


Greedy approach

Tối ưu hóa cục bộ



Thuật toán đưa ra lựa chọn xuất hiện tốt nhất tại thời điểm hiện tại mà không xem xét các tác động lâu dài



Greedy approach



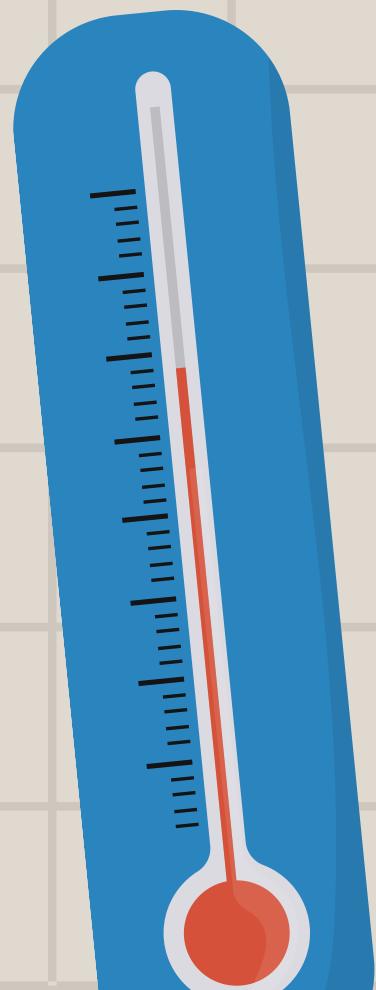
không thể đảo ngược

Một khi lựa chọn được đưa ra, thuật toán tham lam sẽ không quay trở lại hoặc xem xét lại các quyết định trước đó, ngay cả khi giai đoạn sau tiết lộ rằng một lựa chọn ban đầu khác có thể đã dẫn đến kết quả tổng thể tốt hơn



Một số ví dụ về hướng tiếp cận tham lam

**Cho đồ thị n đỉnh và m cạnh
có trọng số, hãy tìm trọng số
của cây khung nhỏ nhất của
đồ thị**



Quay lui

Thuật toán Prim

Thuật toán Kruskal



Thuật toán Kruskal

Kruskal($N, E, cost$) :

sort edges in E by increasing $cost$

while $|T| < |N| - 1$:

 let (u, v) be the next edge in E

if u and v are on different components:

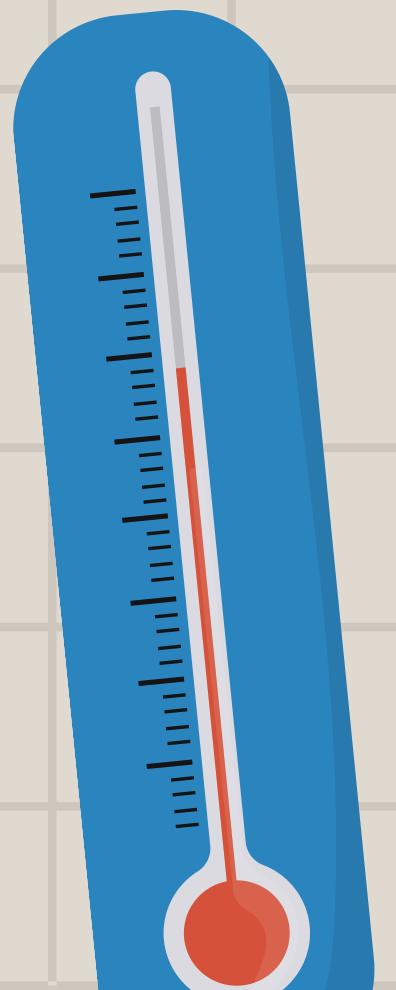
 join the components of u and v

$T = T \cup \{(u, v)\}$

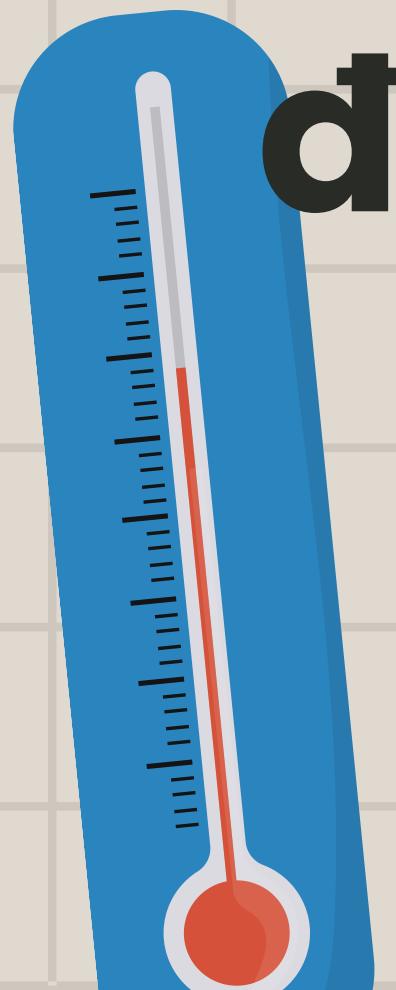
return T



**Cho đồ thị n đỉnh và m cạnh
có trọng số. Hãy tìm đường đi
ngắn nhất từ đỉnh 1 đến mọi
đỉnh trong đồ thị**



**Cho đồ thị n đỉnh và m cạnh
có trọng số không âm. Hãy
tìm đường đi ngắn nhất từ
đỉnh 1 đến mọi đỉnh trong đồ
thị**



Quay lui

Bellman-Ford

Thuật toán Dijkstra



Thuật toán Dijkstra

```
function Dijkstra(Graph, source):
    for each vertex v in Graph.Vertices:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

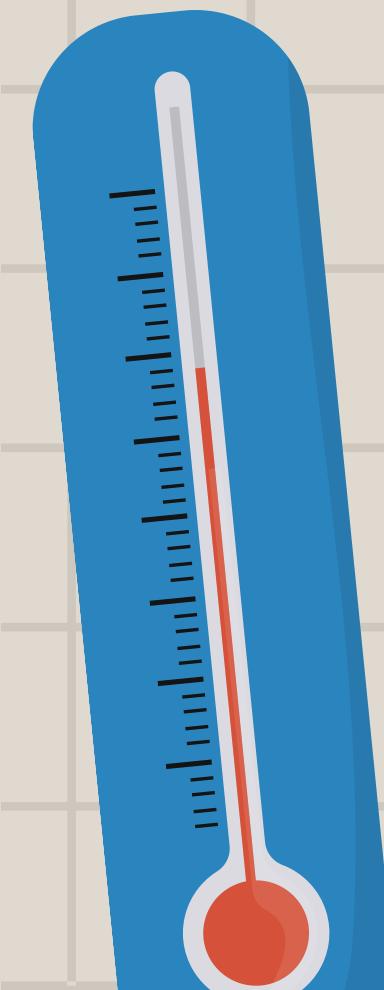
    while Q is not empty:
        u ← vertex in Q with minimum dist[u]
        Q.remove(u)

        for each arc (u, v) in Q:
            alt ← dist[u] + Graph.Edges(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```



Ưu điểm và nhược điểm của greedy approach



Ưu điểm của greedy approach

Độ phức tạp rất tốt

Đơn giản

Có thể ứng dụng trên nhiều loại bài toán

Nhược điểm của greedy approach

Chỉ áp dụng được cho một số bài toán

Nhược điểm của greedy approach

Thuật toán dijkstra không hoạt động trên đồ thị có trọng số âm.

Nhược điểm của greedy approach

Phải tìm được đúng bài toán
cục bộ.

Nhược điểm của greedy approach

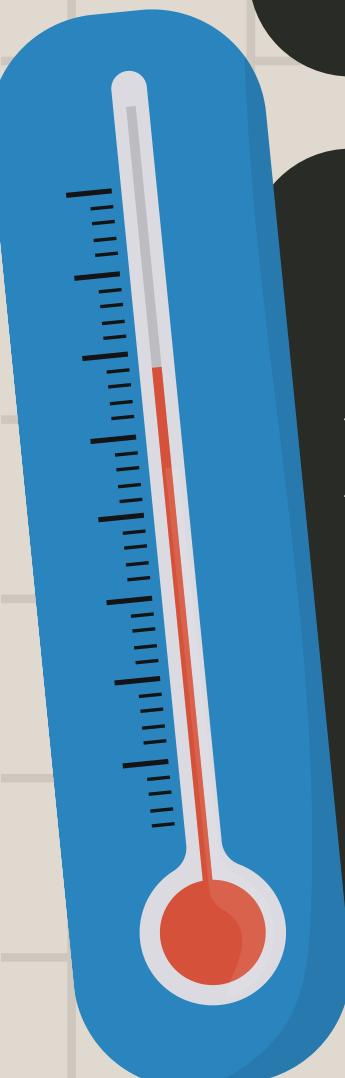
Phải chứng minh phức tạp.

Quy trình chung

Framework của Greedy approach

Mô hình hóa bài toán

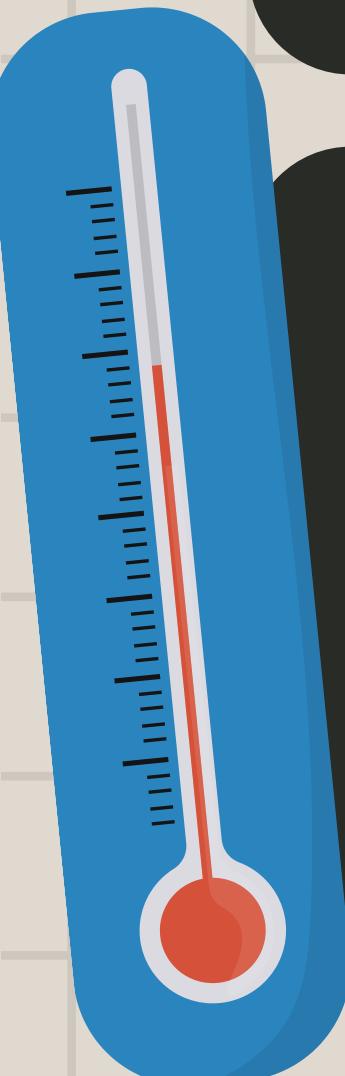
Xác định các đối tượng, ràng buộc, và hàm mục tiêu cần tối ưu.



Framework của Greedy approach

Xác định tiêu chí tham lam

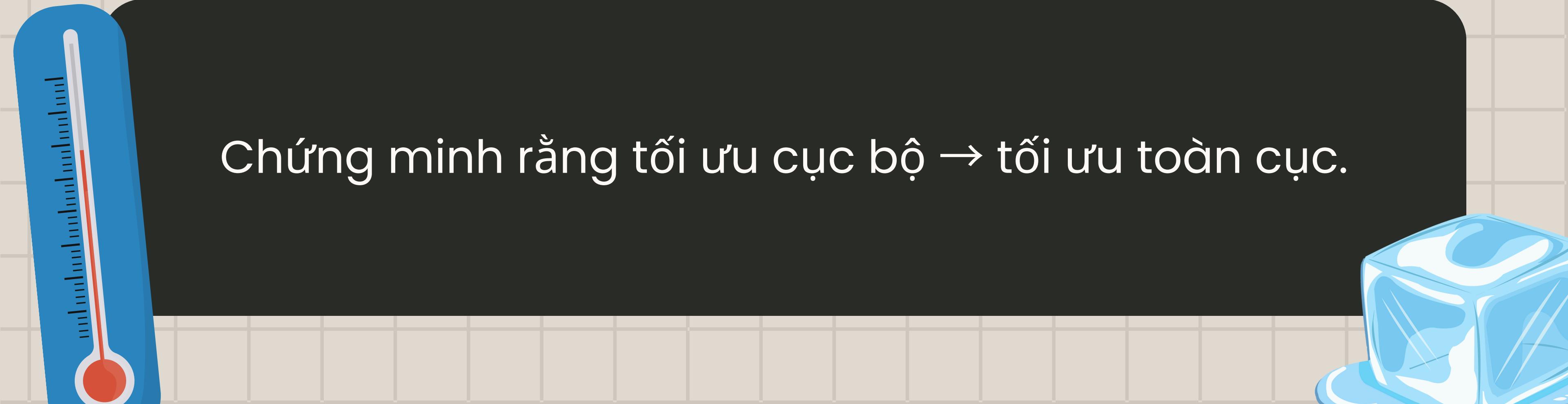
Quy tắc chọn phần tử “tốt nhất hiện tại” và xây dựng đáp án cục bộ.



Framework của Greedy approach

Kiểm chứng tính đúng đắn

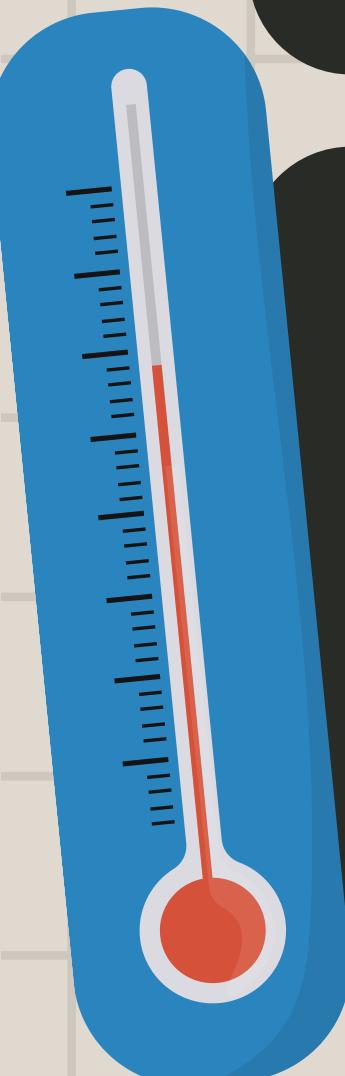
Chứng minh rằng tối ưu cục bộ \rightarrow tối ưu toàn cục.



Framework của Greedy approach

Xây dựng quá trình tham lam

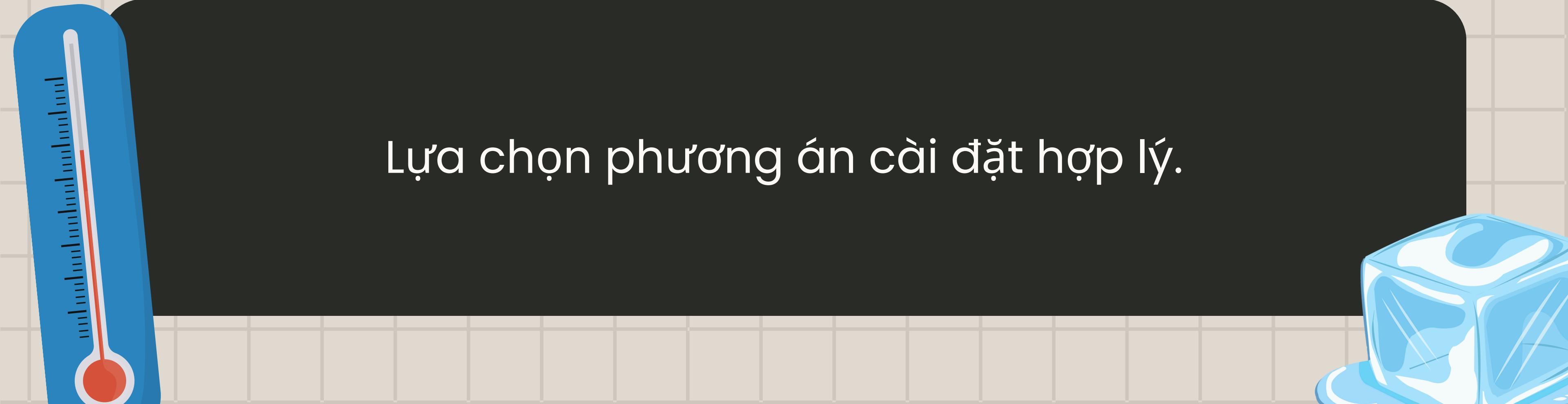
Sử dụng cấu trúc dữ liệu phù hợp với tiêu chí tham lam
nhằm tối ưu độ phức tạp



Framework của Greedy approach

Tính toán độ phức tạp

Lựa chọn phương án cài đặt hợp lý.



Template thường thấy của greedy approach

```
def greedy(data):
    ans = []
    while |data| > 0:
        choose element x which is the current most optimal solution
        if x exists:
            ans <- ans union x
        data <- data \ x
    else stop
```