

ÁP DỤNG GIẢI THUẬT GREEDY CHO BÀI TOÁN CÁI TÚI

Giảng viên: TS. Đặng Ngọc Hoàng Thành

Mã lớp học phần: 24D1INF50904201

NỘI DUNG

1. Tổng quan đề tài

- Giới thiệu bài toán cái túi
- Phát biểu bài toán
- Một số hướng tiếp cận

2. Giải thuật Greedy

- Giới thiệu giải thuật Greedy
- Độ phức tạp của giải thuật
- Ứng dụng của giải thuật

3. Thiết kế giao diện

- Trực quan về giao diện

4. Kết quả thực nghiệm

- Các tình huống
- Phân tích và đánh giá

5. Kết luận

- Các kết quả đạt được
- Những hạn chế và hướng phát triển

TỔNG QUAN ĐỀ TÀI

BÀI TOÁN CÁI TÚI

Knapsack Problem

- Là **bài toán tối ưu hóa tổ hợp** nguồn gốc từ đầu thế kỷ 19, được chú ý rộng rãi vào thế kỷ 20.
- Có **tính ứng dụng cao trong nhiều lĩnh vực thực tiễn** như quản lý dự án, cắt ghép và tối ưu hóa mạng lưới, sử dụng nguyên vật liệu trong sản xuất...và nhiều vấn đề khác.
- Tuy nhiên, **bài toán thuộc loại NP - khó**, nghĩa là không có thuật toán chính xác nào để giải quyết nhanh chóng cho tất cả trường hợp.
- Dù còn nhiều thách thức trong việc giải tối ưu cho mọi trường hợp, bài toán cái túi vẫn đóng vai trò quan trọng trong việc tối ưu hóa quy trình và giảm thiểu chi phí.

PHÁT BIỂU BÀI TOÁN

Yêu cầu và phát biểu bài toán

Cho một túi có **trọng lượng tối đa** là W và một tập hợp các đồ vật, mỗi đồ vật có trọng lượng và giá trị tương ứng đi kèm. Quyết định các đồ vật cần lấy sao cho thỏa mãn yêu cầu:

- **Tổng giá trị** của các đồ vật là **lớn nhất**
- **Tổng trọng lượng** các vật **không vượt quá** trọng lượng tối đa của túi

Ta phát biểu bài toán như sau:

- n là **số vật phẩm**
- v_i là **giá trị** của vật phẩm thứ i ($1 \leq i \leq n$)
- w_i là **trọng lượng** của vật phẩm thứ i ($1 \leq i \leq n$)
- W là **sức chứa tối đa** của cái túi
- x_i là biến nhị phân cho vật phẩm thứ i :
 - $x_i = 1$ nếu vật phẩm thứ i được chọn.
 - $x_i = 0$ nếu vật phẩm thứ i không được chọn.

$$\max \sum_{i=1}^n v_i x_i$$

với $\sum_{i=1}^n w_i x_i \leq W$ và $x_i \in \{0,1\}$, $1 \leq i \leq n$

MỘT SỐ HƯỚNG

Tiếp cận giải quyết bài toán —

Dynamic Programming

Chia nhỏ các vấn đề thành các trường hợp con
Giải quyết từng trường hợp con một cách tối ưu.

Greedy Algorithm

Tính đơn giá của các đồ vật (đơn giá = giá trị / trọng lượng)
Sắp xếp và chọn các đồ vật **theo thứ tự đơn giá giảm dần** để thêm vào túi.
Chọn tiếp các đồ vật cho đến khi **đạt giới hạn trọng lượng của túi** hoặc hết đồ vật.

Binary Programming

Sử dụng biến nhị phân để biểu thị việc chọn hay không chọn từng vật phẩm.
Sau đó **giải quyết bằng phương pháp tối ưu hóa**.

GREEDY ALGORITHM

GIỚI THIỆU

Thuật toán Greedy —

- Là thuật toán giải quyết bài toán theo kiểu **metaheuristic**.
- Thuật toán tham lam lựa chọn giải pháp được cho là tốt nhất ở thời điểm hiện tại.
- Lựa chọn có thể phụ thuộc vào lựa chọn trước, nhưng **không thể** phụ thuộc vào lựa chọn trong tương lai.
- Thuật toán tiến triển theo hướng thực hiện các lựa chọn **theo vòng lặp** và đồng thời **thu nhỏ bài toán đã cho** thành bài toán nhỏ hơn.



Ưu điểm:

- Đưa ra giải pháp tốt trong thời gian ngắn
- Dễ dàng để triển khai trong thực tế

Nhược điểm:

- Đối với một số bài toán, có thể không là thuật toán chính xác và tối ưu nhất.
- Các kết quả thu được của một số bài toán sẽ có sự khác nhau tùy thuộc vào thứ tự đầu vào.

ĐỘ PHỨC TẠP

Của thuật toán Greedy —

- Phụ thuộc vào từng thuật toán cụ thể và cấu trúc dữ liệu sử dụng.
- **Độ phức tạp thời gian** thường là $O(n \log n)$ hoặc $O(n^2)$
- Một số thuật toán sẽ có **độ phức tạp không gian** là $O(n)$
- Trên thực tế, độ phức tạp kể trên chỉ là ước tính trung bình, có thể tốt hơn hoặc xấu hơn tùy thuộc vào thuật toán và cấu trúc dữ liệu sử dụng.

ỨNG DỤNG

Cho bài toán cài túi —

- Xét các vật theo thứ tự **đơn giá giảm dần**
- **Input:** Trọng lượng tối đa mà túi có thể chứa, trọng lượng và giá trị của mỗi đồ vật.
- **Output:** Danh sách các vật được chọn, giá trị tối đa và tổng trọng lượng của túi sau khi bỏ các vật vào.
- **Ý tưởng:** Ưu tiên lựa chọn các đồ vật có **đơn giá** từ cao đến thấp cho vào túi đến khi túi đầy.

Hàm `__init__(self, max_capacity)`

```
class Knapsack:  
    def __init__(self, max_capacity):  
        self.max_capacity = max_capacity  
        self.items = [] # List of (weight, value) tuples  
        self.output_items = [] # List to store selected items
```

- Nhận tham số đầu vào là **đối tượng hiện tại** của class (self) và **trọng lượng tối đa của túi** (max_capacity)

Hàm knapsack_greedy (self)

```
def knapsack_greedy(self):
    # Filter out items with zero weight or zero value
    valid_items = [(weight, value) for weight, value in self.items if weight > 0 and value > 0]

    # calculate value-to-weight ratio for each item
    items_with_ratio = [(value / weight, weight, value) for weight, value in valid_items]
    items_with_ratio.sort(reverse=True, key=lambda x: x[0]) # Sort by ratio in descending order

    total_value = 0
    total_weight = 0
    remaining_capacity = self.max_capacity
    for _, weight, value in items_with_ratio:
        if weight <= remaining_capacity:
            total_value += value
            total_weight += weight
            remaining_capacity -= weight
            self.output_items.append((weight, value)) # Add selected item
        if remaining_capacity == 0:
            break
```

Hàm knapsack_greedy (self)

```
else:  
    # Fractional part of the last item  
    fraction = remaining_capacity / weight  
    fractional_value = fraction * value  
    total_weight += remaining_capacity  
    total_value += fractional_value  
    self.output_items.append((remaining_capacity, fractional_value)) # Add fractional item  
break  
  
return total_value, total_weight
```

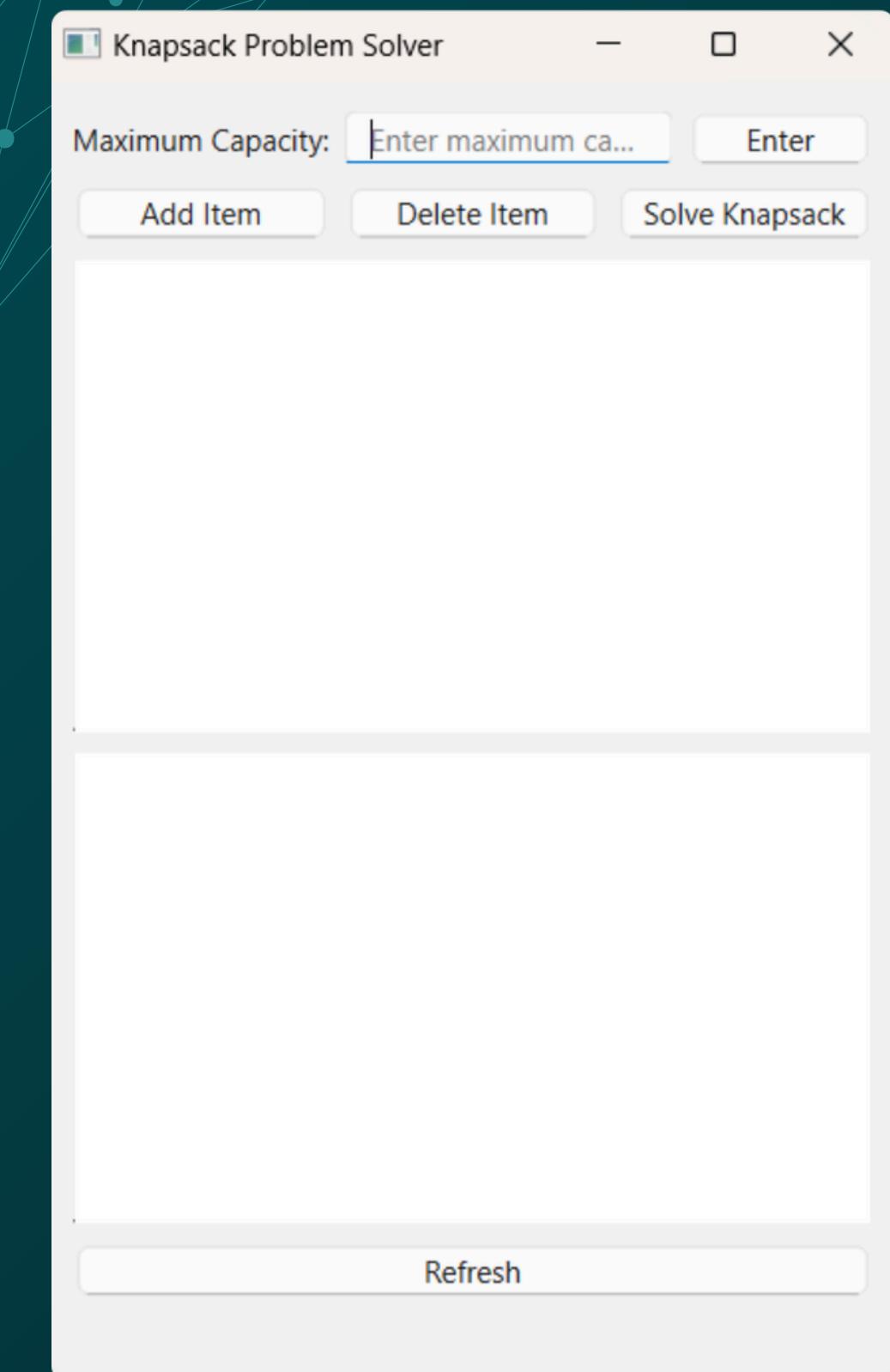
- Nhận tham số đầu vào là **đối tượng hiện tại** của class (self), tham chiếu đến đối tượng hiện tại của class.

THIẾT KẾ GIAO DIỆN

GIAO DIỆN MÀN HÌNH

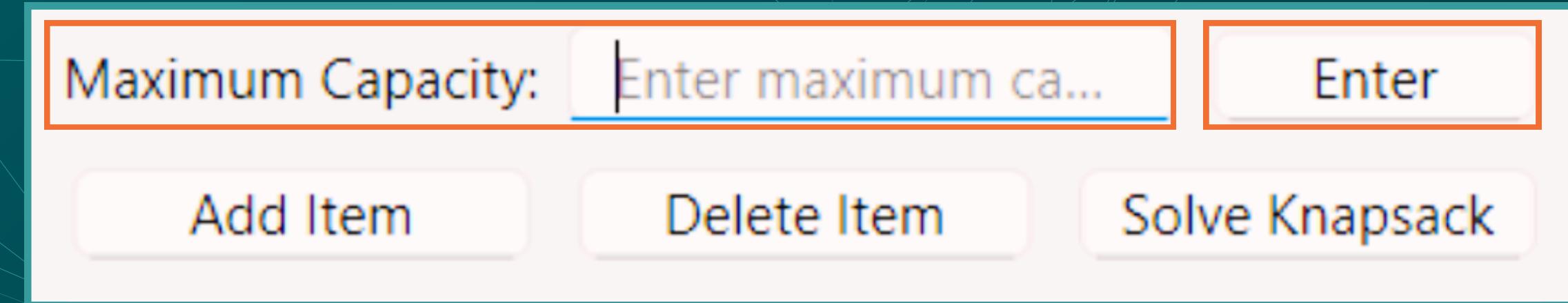
Giao diện màn hình chính giải bài toán —

- Giao diện màn hình đơn giản cho phép nhập thông tin bài toán cái túi và giải quyết bài toán.
- Người dùng có thể **nhập trọng lượng tối đa** của cái túi, **thêm hoặc xóa các đồ vật** với khối lượng và giá trị tương ứng.
- Nhấn nút **“Solve Knapsack”** để giải bài toán.

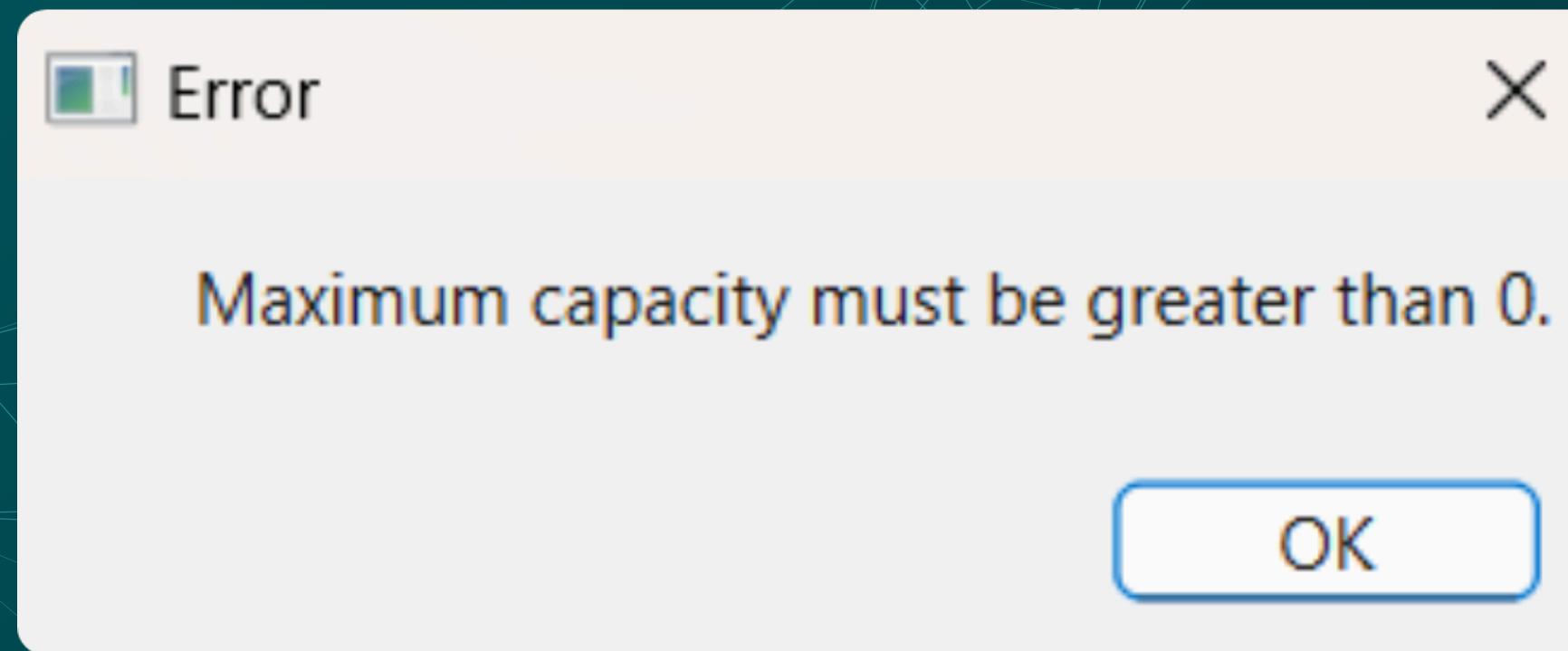


CHI TIẾT CHỨC NĂNG

Các nút và trường nhập



- **Trường nhập “Maximum Capacity”:** Đây là trường để người dùng nhập trọng lượng tối đa của túi.
- **Nút “Enter”:** Đây là nút xác nhận trọng lượng tối đa nhập vào và xác nhận cập nhật trọng lượng tối đa mới. Sau khi nhấn nút này mới có thể thêm, xóa đồ vật.

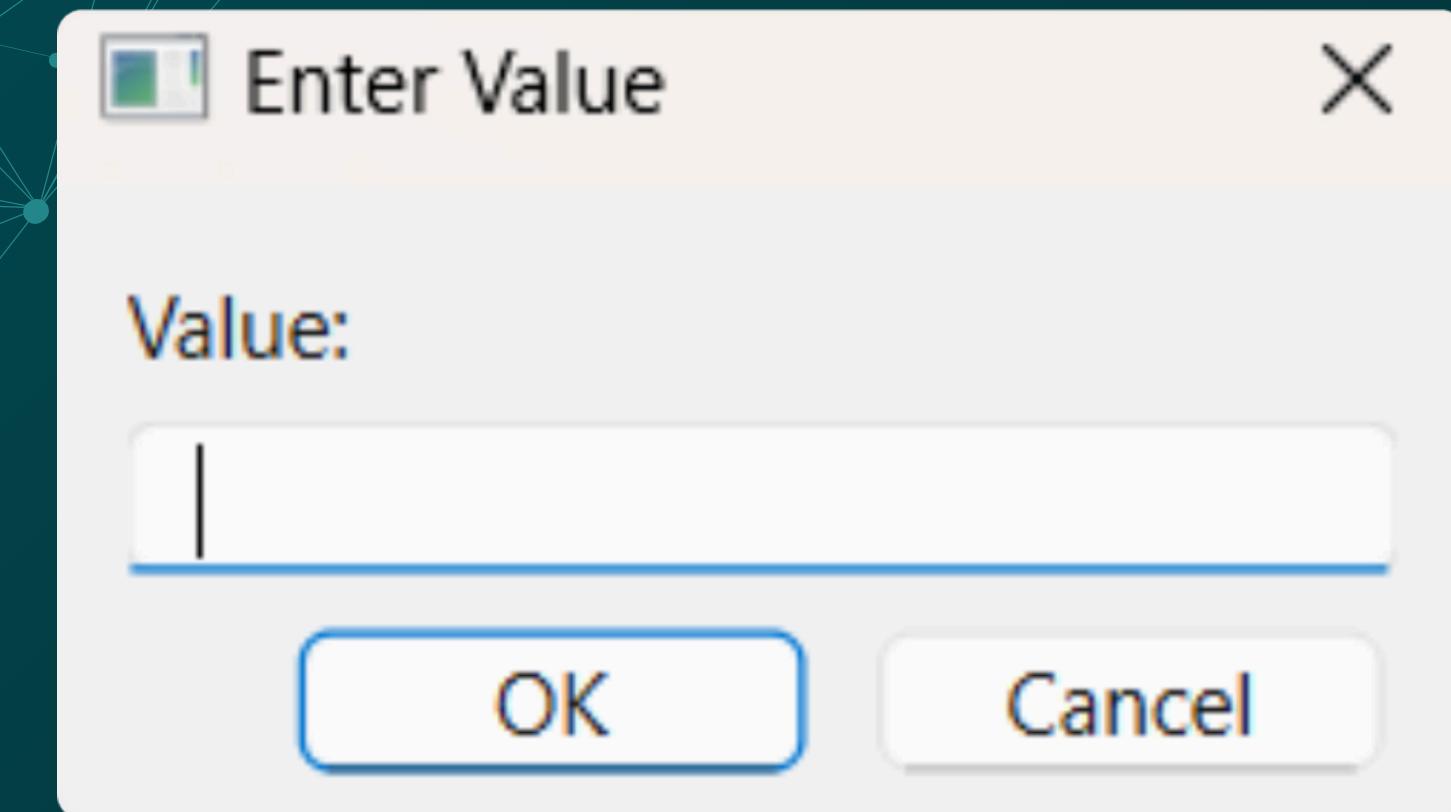
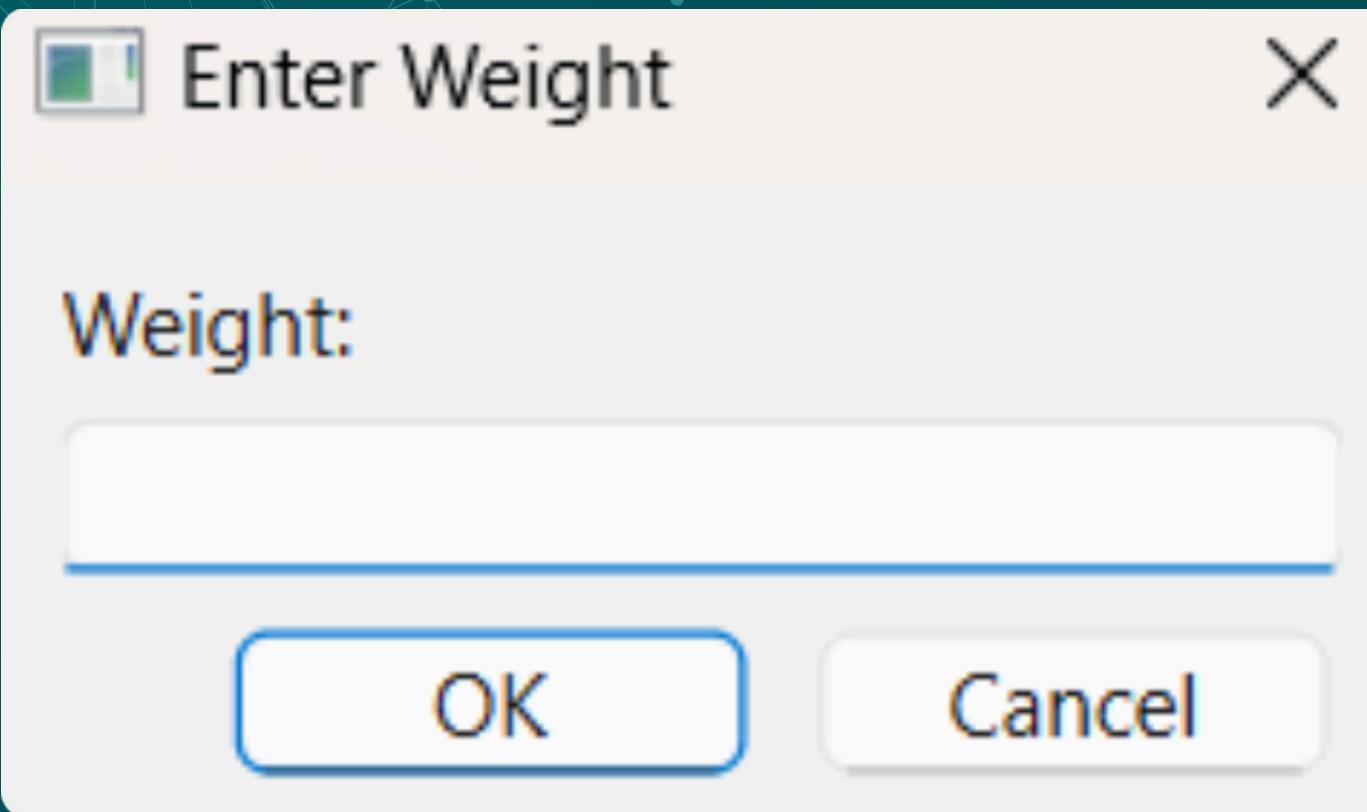


- **Cửa sổ báo lỗi:** Đây là cửa sổ hiện ra nếu người dùng nhập trọng lượng tối đa của túi bằng 0.

Maximum Capacity: Enter

Add Item Delete Item Solve Knapsack

- **Nút “Add Item”:** Đây là nút để thêm các đồ vật với giá trị và khối lượng tương ứng. Sau khi nhấn nút này, sẽ xuất hiện hai cửa sổ nhỏ, lần lượt chứa các trường để nhập khối lượng của vật và giá trị của vật.



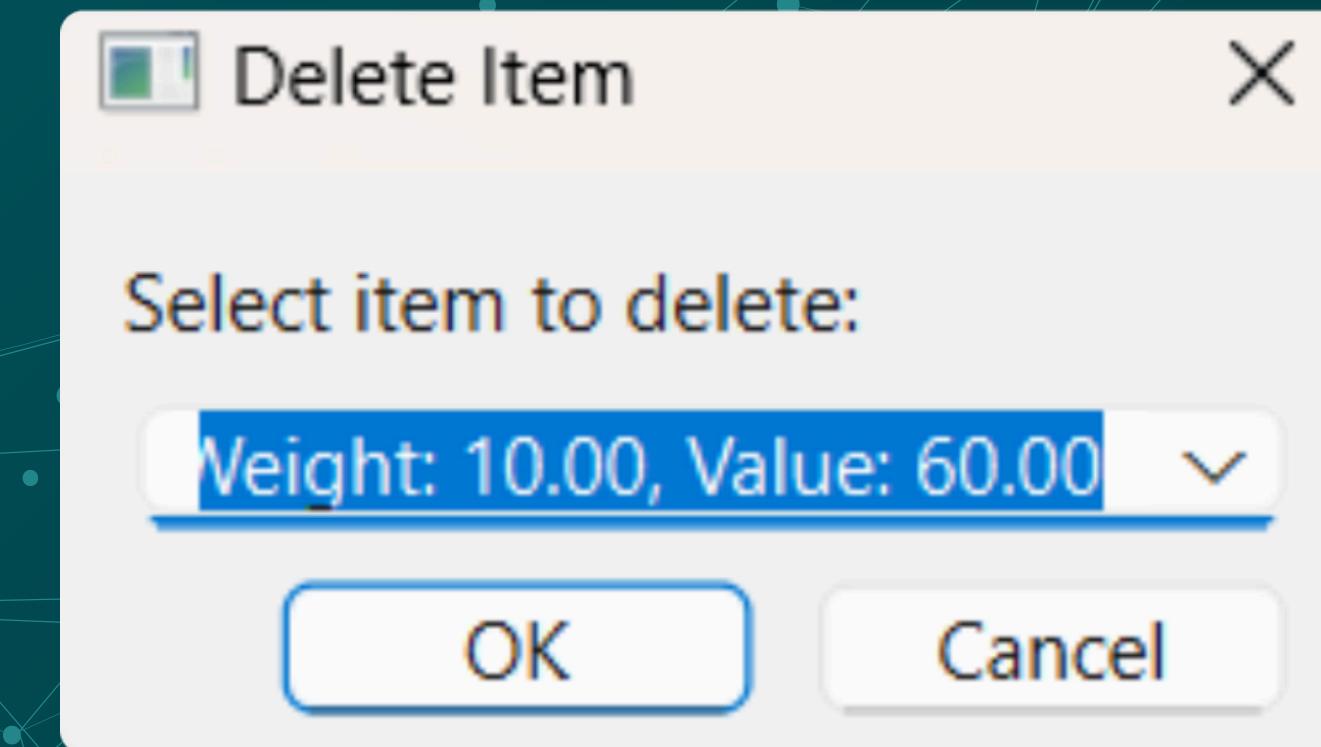
*Cửa sổ chứa trường nhập
“Weight”*

*Cửa sổ chứa trường nhập
“Value”*

Maximum Capacity: Enter

Add Item Delete Item Solve Knapsack

- **Nút “Delete Item”:** Đây là nút để xóa đồ vật một cách riêng lẻ mà không cần phải nhập lại từ đầu. Sau khi xóa đồ vật, trường hiển thị kết quả sẽ tự động được xóa và cập nhật mới.



*Cửa sổ chứa trường nhập
“Select item to delete”*

Maximum Capacity:

- **Nút “Solve Knapsack”:** Đây là nút để thực hiện giải quyết bài toán sau khi đã nhập thông tin khối lượng và giá trị của các đồ vật.

Added item - Weight: 1.00, Value: 3.00
Added item - Weight: 2.00, Value: 6.00
Added item - Weight: 12.00, Value: 36.00
Added item - Weight: 18.00, Value: 54.00
Added item - Weight: 20.00, Value: 60.00
Added item - Weight: 10.00, Value: 30.00
Added item - Weight: 15.00, Value: 45.00

Selected items:
Weight: 6.00, Value: 75.00
Weight: 7.00, Value: 60.00
Weight: 1.00, Value: 5.00
Weight: 1.00, Value: 5.00

Maximum value: 145.00
Total weight: 15.00

Trường hiển thị đồ vật nhập vào

Trường hiển thị kết quả bài toán

Refresh

- **Nút “Refresh”:** Đây là nút cập nhật toàn bộ cửa sổ, sau khi nhấn nút này, toàn bộ các trường sẽ được làm mới, trạng thái các nút cũng sẽ trở lại như cũ.

KẾT QUẢ THỰC NGHIỆM

CÁC TÌNH HUỐNG

Của bài toán cái túi

Tình huống 1: Trọng lượng tối đa của túi là 15 (kg), trọng lượng và giá trị tương ứng của các vật được thể hiện trong bảng sau:

Items	1	2	3	4	5	6	7
Value	5	10	15	7	8	9	4
Weight	1	3	5	4	1	4	2

Kết quả thu được ở tình huống 1:

Selected items:

Weight: 1.00, Value: 8.00
Weight: 1.00, Value: 5.00
Weight: 3.00, Value: 10.00
Weight: 5.00, Value: 15.00
Weight: 4.00, Value: 9.00
Weight: 1.00, Value: 2.00

Maximum value: 49.00
Total weight: 15.00

Tình huống 2: Trọng lượng tối đa của túi là 15 (kg), trọng lượng và giá trị tương ứng của các vật được thể hiện trong bảng sau:

Items	1	2	3	4	5	6	7
Value	5	10	15	20	30	35	40
Weight	1	2	3	4	6	7	8

Kết quả thu được ở tình huống 2:

Selected items:

Weight: 1.00, Value: 5.00

Weight: 2.00, Value: 10.00

Weight: 3.00, Value: 15.00

Weight: 4.00, Value: 20.00

Weight: 5.00, Value: 25.00

Maximum value: 75.00

Total weight: 15.00

Tình huống 3: Trọng lượng tối đa của túi là 15 (kg), trọng lượng và giá trị tương ứng của các vật được thể hiện trong bảng sau:

Items	1	2	3	4	5	6	7	8
Value	5	10	15	20	75	60	40	50
Weight	1	2	3	4	6	7	8	12

Kết quả thu được ở tình huống 3:

Selected items:

Weight: 6.00, Value: 75.00

Weight: 7.00, Value: 60.00

Weight: 1.00, Value: 5.00

Weight: 1.00, Value: 5.00

Maximum value: 145.00

Total weight: 15.00

Tình huống 4: Trọng lượng tối đa của túi là 15.6 (kg), trọng lượng và giá trị tương ứng của các vật được thể hiện trong bảng sau:

Items	1	2	3	4	5	6	7	8
Value	5	10	15.3	21	75.9	60	40	50.45
Weight	1.2	2	3	4	6	7.4	9	12.75

Kết quả thu được ở tình huống 4:

Selected items:
Weight: 6.00, Value: 75.90
Weight: 7.40, Value: 60.00
Weight: 2.20, Value: 11.55

Maximum value: 147.45
Total weight: 15.60

Tình huống 5: Trọng lượng tối đa của túi là 100 (kg), trọng lượng và giá trị tương ứng của các vật được thể hiện trong bảng sau:

Items	1	2	3	4	5	6	7
Value	3	0	50	60	0	29	70
Weight	1	2	12	18	4	10	15

Kết quả thu được ở tình huống 5:

Selected items:
Weight: 15.00, Value: 70.00
Weight: 12.00, Value: 50.00
Weight: 18.00, Value: 60.00
Weight: 1.00, Value: 3.00
Weight: 10.00, Value: 29.00

Maximum value: 212.00
Total weight: 56.00

Tình huống 6: Trọng lượng tối đa của túi là 20 (kg), trọng lượng và giá trị tương ứng của các vật được thể hiện trong bảng sau:

Items	1	2	3	4	5	6	7
Value	3	6	36	54	60	30	45
Weight	1	2	12	18	20	10	15

Kết quả thu được ở tình huống 6:

Selected items:
Weight: 1.00, Value: 3.00
Weight: 2.00, Value: 6.00
Weight: 12.00, Value: 36.00
Weight: 5.00, Value: 15.00

Maximum value: 60.00
Total weight: 20.00

PHÂN TÍCH VÀ ĐÁNH GIÁ

Các tình huống đã cho

Tình huống 1

- Trọng lượng, giá trị và các **đơn giá** của các đồ vật đều **khác nhau**.
- Sắp xếp và chọn các đồ vật theo thứ tự đơn giá từ cao đến thấp cho đến khi đầy túi.

Selected items:

Weight: 1.00, Value: 8.00

Weight: 1.00, Value: 5.00

Weight: 3.00, Value: 10.00

Weight: 5.00, Value: 15.00

Weight: 4.00, Value: 9.00

Weight: 1.00, Value: 2.00

Maximum value: 49.00

Total weight: 15.00

Tình huống 2

- Trọng lượng và giá trị của các đồ vật **khác nhau**, nhưng đơn giá **giống nhau**
- Sắp xếp và chọn các đồ vật **không** theo thứ tự đơn giá từ cao đến thấp mà **xếp và chọn theo thứ tự đưa vào** cho đến khi đầy túi.

Selected items:

Weight: 1.00, Value: 5.00

Weight: 2.00, Value: 10.00

Weight: 3.00, Value: 15.00

Weight: 4.00, Value: 20.00

Weight: 5.00, Value: 25.00

Maximum value: 75.00

Total weight: 15.00

Tình huống 3

- Trọng lượng và giá trị của các đồ vật **khác nhau**, nhưng có một số vật có tỷ lệ đơn giá **giống nhau** (Item 1, 2, 3, 4, 7)
- Sắp xếp và chọn các đồ vật theo thứ tự đơn giá từ cao đến thấp. Đối với các đồ vật **cùng đơn giá**, thuật toán **xếp theo thứ tự đưa vào** cho đến khi đầy túi.

Selected items:

Weight: 6.00, Value: 75.00

Weight: 7.00, Value: 60.00

Weight: 1.00, Value: 5.00

Weight: 1.00, Value: 5.00

Maximum value: 145.00

Total weight: 15.00

Tình huống 4

- Trọng lượng tối đa, trọng lượng và giá trị của các vật đều là **số thực**.
- Sắp xếp và chọn các đồ vật **theo thứ tự đơn giá từ cao đến thấp**.

Selected items:

Weight: 6.00, Value: 75.90

Weight: 7.40, Value: 60.00

Weight: 2.20, Value: 11.55

Maximum value: 147.45

Total weight: 15.60

Tình huống 5

- Trọng lượng, giá trị và các đơn giá của các đồ vật đều khác nhau. Nhưng có một số vật **có giá trị bằng 0** (Item 2, 5). Trọng lượng tối đa của túi **rất lớn** (100kg)
- Sắp xếp và chọn các đồ vật theo thứ tự đơn giá từ cao đến thấp. Các vật **có giá trị bằng 0 không được thêm vào.**

Selected items:

Weight: 15.00, Value: 70.00

Weight: 12.00, Value: 50.00

Weight: 18.00, Value: 60.00

Weight: 1.00, Value: 3.00

Weight: 10.00, Value: 29.00

Maximum value: 212.00

Total weight: 56.00

Tình huống 6

- Trọng lượng, giá trị khác nhau nhưng tất cả các vật lại **có cùng tỷ lệ đơn giá**. Ngoài ra, có 1 vật có trọng lượng **bằng trọng lượng tối đa của túi**.
- Chọn các đồ vật: **theo thứ tự đưa vào danh sách lựa chọn ban đầu** cho đến khi đầy túi.

Selected items:

Weight: 1.00, Value: 3.00

Weight: 2.00, Value: 6.00

Weight: 12.00, Value: 36.00

Weight: 5.00, Value: 15.00

Maximum value: 60.00

Total weight: 20.00

KẾT LUẬN

CÁC KẾT QUẢ ĐẠT ĐƯỢC

Qua các tình huống

Tình huống	Maximum Value	Total Weight	Selected Item
Tình huống 1	49.00	15.00	(1, 8), (1, 5), (3, 10), (5, 15), (4, 9), (1, 2)
Tình huống 2	75.00	15.00	(1, 5), (2, 10), (3, 15), (4, 20), (5, 25)
Tình huống 3	145.00	15.00	(6, 75), (7, 60), (1, 5), (1, 5)
Tình huống 4	147.45	15.60	(6, 75.9), (7.4, 60), (2.2, 11.55)
Tình huống 5	212.00	56.00	(15, 70), (12, 50), (18, 60), (1, 3), (10, 29)
Tình huống 6	60.00	20.00	(1, 3), (2, 6), (12, 36), (5, 15)

HẠN CHẾ VÀ HƯỚNG PHÁT TRIỂN

Của thuật toán Greedy đối với bài toán cái túi —

Hạn chế

- Khả năng giải pháp được đưa ra rơi vào **tối ưu cục bộ**, khiến cho thuật toán **không đảm bảo tìm được giải pháp tối ưu toàn cục**.
- Thuật toán tham lam cũng có thể **nhạy cảm** và có thể **không thể xử lý** **được ràng buộc về không gian** hay những **thay đổi của đầu vào**, điều này có thể **ảnh hưởng tới thuật toán** khiến cho thuật toán **không còn tính ổn định** và **không thể đoán trước**.

Hạn chế

- Các thuật toán tham lam có thể **không được áp dụng cho các vấn đề** mà các giải pháp tối ưu trong vấn đề ấy **phụ thuộc vào thứ tự được xử lý, kích thước** hay **thành phần của đầu vào.**
- Thuật toán tham lam **không đảm bảo hoàn toàn** về việc **tìm ra được giải pháp tối ưu nhất** hay có thể đưa ra được các giải pháp mang tính tối ưu **cho một vài trường hợp.**

Hướng phát triển

- Tối ưu hóa thuật toán cùng với việc **kết hợp với thuật toán tối ưu toàn cục** như **Simulated Annealing** hay **thuật toán Genetic** sẽ giúp cho thuật toán tham lam cải thiện được khả năng tìm kiếm các giải pháp tối ưu toàn cục.
- Việc **phát triển công cụ, framework** cũng như là thuật toán tham lam song song và phân tán cũng giúp cho việc áp dụng thuật toán này trở nên hữu dụng hơn về khả năng ứng dụng mà còn mở rộng thêm cơ hội nghiên cứu và cải tiến mới, giúp cho việc giải quyết các bài toán phức tạp hiệu quả hơn.

CẢM ƠN THẦY ĐÃ LẮNG NGHE