## Bài thực hành 08:

## CÀI ĐẶT BÀI TOÁN BALO1

## Mô tả bài toán

### Đầu vào

Dòng 1: chứa 2 số n và w

Dòng 2: chứa n số nguyên a[0], a[1], ... ,a[n-1]

### Đầu ra

Dòng 1: khối lượng tối ưu của balo sau khi chọn các món hàng

Dòng 2: thứ tự các món hàng được chọn (số nhỏ ghi trước số lớn ghi sau)

### Cách xử lý

Khởi tạo một bảng dp có kích thước (n+1) x (w+1), với n là số lượng món hàng và w là khả năng chứa của balô. Các giá trị trong bảng được khởi tạo ban đầu là 0.

Với mỗi món hàng từ 0 đến n-1, thực hiện các bước sau:

Với mỗi khả năng chứa từ 0 đến w, thực hiện các bước sau:

Nếu khối lượng của món hàng hiện tại weights[i] nhỏ hơn hoặc bằng khả năng chứa j, thì ta có thể chọn món hàng này.

So sánh giá trị tối ưu giữa trường hợp chọn món hàng hiện tại và không chọn món hàng hiện tại. Lưu giá trị tối ưu vào dp[i+1][j].

Sau khi hoàn thành bước trên, giá trị tối ưu sẽ được lưu trong ô dp[n][w]. Đây là khối lượng tối ưu mà balô có thể chứa.

Để xác định các món hàng được chọn, ta thực hiện truy vết từ ô dp[n][w] về phía trước. Ta kiểm tra xem có chọn món hàng i hay không bằng cách so sánh giá trị của ô dp[i][w] và dp[i+1][w]. Nếu khác nhau, tức là đã chọn món hàng i, ta ghi nhận lại món hàng i vào danh sách kết quả và giảm giá trị của w đi weights[i].

Kết quả cuối cùng sẽ là danh sách các món hàng được chọn và khối lượng tối ưu của balô.

## Hướng dẫn thực thi chương trình

Hàm knapsack nhận vào số lượng món hàng n, khả năng chứa của balo w và một vector weights chứa khối lượng của từng món hàng.

Chúng ta sử dụng một bảng hai chiều dp để lưu trữ giá trị tối ưu tại mỗi bước tính toán. Giá trị dp[i][j] là khối lượng tối ưu mà balo có thể chứa khi chỉ xét đến i món hàng đầu tiên và có dung lượng tối đa là j.

Qua mỗi bước tính toán, chúng ta cập nhật giá trị của dp[i][j] dựa trên giá trị tối ưu đã tính toán từ trước đó.

Sau khi tính toán xong, chúng ta truy vết từ dp[n][w] để lấy ra.