1. Ví dụ 1 – có thể dùng thuật toán tham lam:

**Đề bài**: bạn được cung cấp n hoạt động với thời gian bắt đầu và thời gian kết thúc. Chọn số lượng tối đa hoạt động mà có thể thực hiện bởi một người. Giả định rằng mỗi người chỉ có thể làm tối đa một công việc một lúc.

* **Ý tưởng:** Để có thể chọn ra nhiều hoạt động có thể thực hiện nhất. Thì ý tưởng dễ thấy nhất ở đây là là ta sẽ sắp sếp theo thời gian kết thúc của các hoạt động, sau đó ta sẽ chọn ra hoạt động có thời gian kết thúc sớm nhất và đảm bảo thời gian bắt đầu của nó muộn hơn thời gian kết thúc của các hoạt động được chọn trước đó. Hoạt động được chọn đầu tiên chính là hoạt động có thời gian kết thúc sớm nhất.
* **VD:** Cho 3 hoạt động được sắp xếp theo thời gian bắt đầu và thời gian kết thúc.

start[] = {10, 12, 20};

finish[] = {20, 25, 30};

Thì số lượng công việc tối đa mà một người có thể thực hiện là 2, công việc đầu tiên và công việc cuối cùng. Hai công việc không trùng nhau về thời gian kết thúc.

* **Bài giải bằng code python:**
* **Time Complexity :**  O(n log n) đới với chương trình python chưa được sắp xếp. O(n) đối với trương trình python đã sắp xếp

1. Ví dụ 2(cũng có thể dùng tham lam nhưng sẽ có trường hợp sai)

Đề bài: Một kẻ trộm đột nhập vào một cửa hiệu tìm thấy có n mặt hàng có trọng lượng và giá trị khác nhau, nhưng hắn chỉ mang theo một cái túi có sức chứa về trọng lượng tối đa là M. Vậy kẻ trộm nên bỏ vào ba lô những món nào và số lượng bao nhiêu để đạt giá trị cao nhất trong khả năng mà hắn có thể mang đi được.

* Ý tưởng: Theo yêu cầu của bài toán thì ta cần lựa chọn những vật có giá trị càng cao và khối lượng càng thấp càng tốt. Từ đó ta suy ra thứ ta cần quan tâm đến là “đơn giá”, tức tỉ lệ giá trị trên khối lượng của mỗi đồ vật. Đơn giá càng cao thì đồ càng ngon. Sau đó chọn theo thứ tự giảm dần của đơn giá mà khối lượng còn lại của túi có thể đựng.
* VD: Tên trộm có một túi đồ có thể đựng được 37 kg, cùng với đó là 4 món đồ với giá trị và khối lượng tương ứng.

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

Ta tính đơn giá của mỗi đồ vật và sắp xếp giảm dần theo đơn giá:

|  |  |  |  |
| --- | --- | --- | --- |
| Đồ vật | Trọng lượng | Giá trị | Đơn giá |
| B | 10 | 25 | 2.5 |
| A | 15 | 30 | 2.0 |
| D | 4 | 6 | 1.5 |
| C | 2 | 2 | 1.0 |

|  |
| --- |
|  |

Theo như ý tưởng trên kia thì các đồ vật và số lượng mà ta chọn sẽ là: 3 đồ vật B và 1 đồ vật D và 1 đồ vật C. Với tổng trọng lượng là

3\*10 + 1\*4 +1\*2 = 36 và tổng giá trị là 3\*25 + 1\*6 +1\*2 = 83.

* Bài giải code python:
* Trường hợp sai của bài khi dùng tham lam để giải bài toán này:

Tham lam có thể giải quyết nhanh chóng và cũng có thể tối ưu trong một số trường hợp nhưng trong vài trường hợp đặc biệt thì nó sẽ cho ta kết quả sai.

VD: Túi có thể đựng được khối lượng tối đa là 10kg

|  |  |  |  |
| --- | --- | --- | --- |
| Đồ vật | Trọng lượng | Giá trị | Đơn giá |
| A | 7 | 9 | 9/7 |
| B | 6 | 6 | 1 |
| C | 4 | 4 | 1 |

Tham lam sẽ chọn đồ vật A nhưng phương pháp tối ưu là B và C

1. Bài tập về nhà:

Đề bài:Cho một đồ thị và một đỉnh nguồn trong đồ thị đó. Hãy tìm đường đi ngắn nhất từ đỉnh đó đến tất cả các đỉnh trong đồ thị.

* Ý tưởng cách làm:

1. Tạo một tập stpSet(shortest path tree set) lưu trữ đường đi ngắn nhất từ nguồn đến các đỉnh đã tính toán xong. Ban đầu, tập này là rỗng.
2. Gán giá trị khoảng cách cho tất cả các đỉnh trong đồ thị. Khởi tạo khoảng cách của tất cả các đỉnh là INFINITE. Còn đỉnh nguồn được gán bằng 0 đề chọn đầu tiên.
3. Khi stpSet chưa có đủ hết các đỉnh:
4. Chọn một đỉnh u chưa nằm trong stpSet và có giá trị khoảng cách nhỏ nhất.
5. Thêm u vào stpSet
6. Cập nhật khoảng cách của tất cả cá đỉnh liền kề u. Để cập nhật các giá trị khoảng cách, lặp qua tất cả các đỉnh liền kề u. Đối mới mọi đỉnh liền kề v, nếu tổng giá trị khoảng cách của u (từ nguồn) và độ dài của cạnh u-v, nhỏ hơn giá trị khoảng cách của v, thì cập nhật giá trị khoảng cách của v.

Độ phức tạp thời gian của việc thực hiện là O (V ^ 2) với V là số đỉnh.



 

 

