Greedy Approach

Group 9

Member:

- Phan Minh Nhật
- Lê Võ Tiến Phát
- Dương Nguyễn Thuận

Menu

1. Đặt Vấn Đề

Trần Đức Bo là một lập trình viên AI và anh ấy khá bận với công việc của mình. Hôm nay là Chủ nhật và Bo quyết định bỏ ra 5h để dọn dẹp. Tuy nhiên trong nhà lại có quá nhiều việc để làm và Bo mong muốn làm được nhiều việc nhất trong thời gian cho phép. Hãy để Bo tìm ra các việc phù hợp để làm.

Dọn phòng	3h
Rửa chén	0.5h
Sửa bàn làm việc	2h
Giặt đồ	1.5h
Thu dọn quần áo	1h

Mục tiêu: Càng nhiều việc càng tốt

Mục tiêu: Càng nhiều việc càng tốt

- Sắp xếp mảng thời gian A tăng dần: A=[0.5,1,1.5,2,3]
- Maxtime=5, Time=0, Work=0,i=0
- While Time <= Maxtime:
    Time+=A[i]
    Work+=1
    i+=1
- Xuất ra Work -1 là số việc nhiều nhất có thể hoàn thành.

Kỳ nghỉ của Bo

Do Bo làm việc siêng năng và hiệu quả nên quản lý đã cho Bo nghỉ ngơi một tuần liền .Trong kỳ nghỉ, Bo đã lên kế hoạch thực hiện một số hoạt động nghỉ ngơi, thư giãn cho mình. Chẳng may thay, Bo liệt kê ra quá nhiều hoạt động nên cũng không biết chọn như thế nào cho phù hợp. Cũng như lần trước, hãy giúp Bo lên lịch trình với nhiều hoạt động nhất.

ID	Hoạt động	Thời gian
1	Xem phim	22:00 thứ 2 - 01:00 thứ 3
2	Du lịch Nha Trang	06:00 thứ 3 - 22:00 thứ 7
3	Tham gia game show	11:00 thứ 3 - 21:00 thứ 3
4	Nghe nhạc	19:00 thứ 3 - 23:00 thứ 3
5	Chơi game	15:00 thứ 4 - 15:00 thứ 5
6	Đá bóng	10:00 thứ 5 - 16:00 thứ 5
7	Đi cà phê cóc	12:00 thứ 7 - 14:00 thứ 7
8	Xông hơi	20:30 thứ 7 - 20:45 thứ 7
9	Ngủ	21:00 thứ 7 - 06:00 CN
10	Đọc truyện tranh	21:01 thứ 7 - 23:59 thứ 7

Mục tiêu: Càng nhiều hoạt động càng tốt

Mục tiêu: Càng nhiều hoạt động càng tốt

Tiếp cận 1

- Chọn những hoạt động có thời gian thực hiện ngắn nhất
- Nếu hoạt động tiếp theo được chọn bị trùng với các hoạt động trước đó thì bỏ qua.

Kết quả

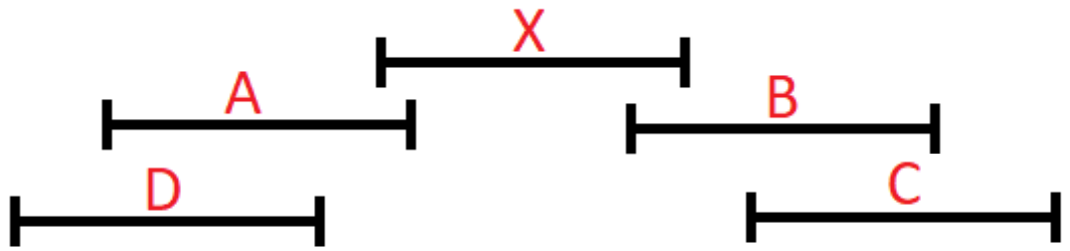
8	Xông hơi	20:30 thứ 7 - 20:45 thứ 7
7	Đi cà phê cóc	12:00 thứ 7 - 14:00 thứ 7
10	Đọc truyện tranh	21:01 thứ 7 - 23:59 thứ 7
1	Xem phim	22:00 thứ 2 - 01:00 thứ 3
4	Nghe nhạc	19:00 thứ 3 - 23:00 thứ 3
6	Đá bóng	10:00 thứ 5 - 16:00 thứ 5

Thật may mắn, 6 hoạt động cũng chính là kết quả tối ưu trong bài toán này

Mục tiêu: Càng nhiều địa điểm càng tốt

Tiếp cận 2

- Chọn hoạt động ít trùng với các hoạt động khác nhất.
- Giả sử ta chọn X, tìm xem có A và B cùng trùng với X nhưng không trùng nhau hay không



Kết quả: 2

Tối ưu: 3

Mục tiêu: Càng nhiều địa điểm càng tốt

Tiếp cận 3

- Chọn hoạt động có kết thúc sớm nhất, thời gian còn lại dành để chọn cho các hoạt động khác.

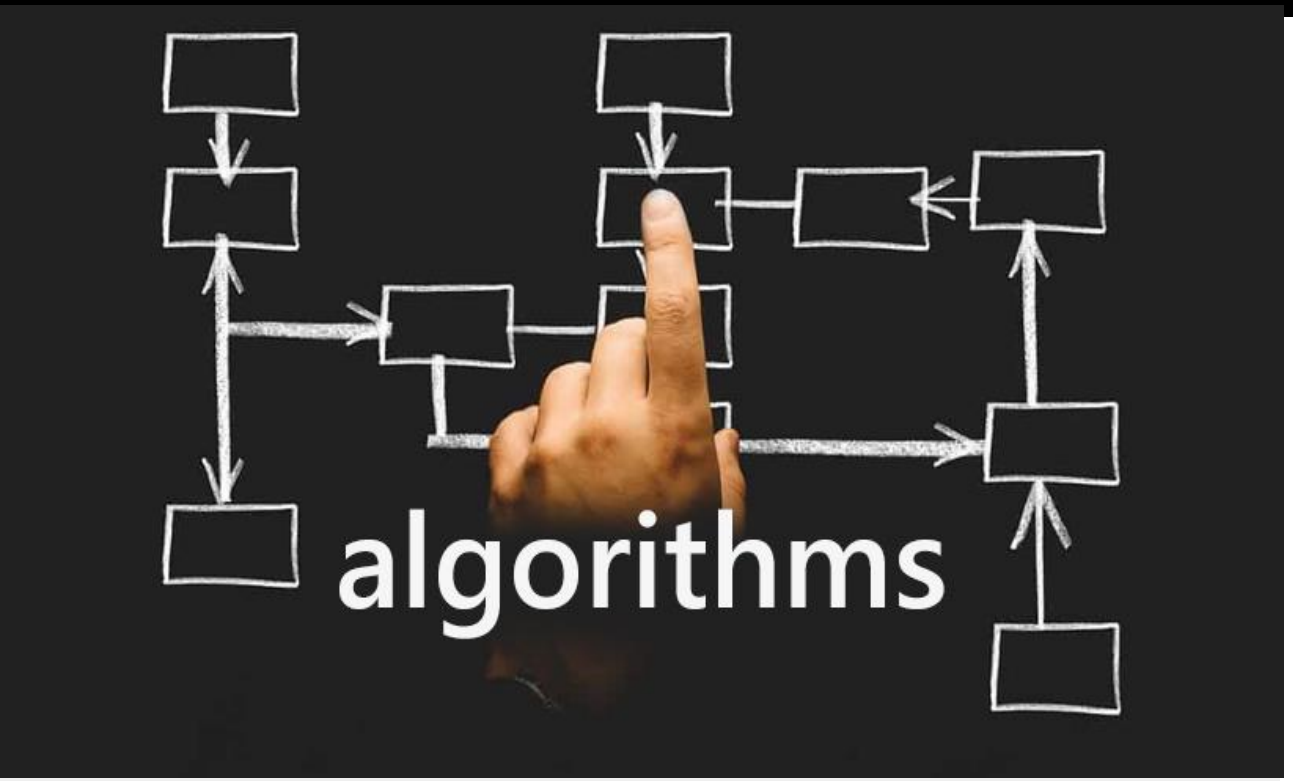
Kết quả

1	Xem phim	22:00 thứ 2 - 01:00 thứ 3
3	Tham gia game show	11:00 thứ 3 - 21:00 thứ 3
5	Chơi game	15:00 thứ 4 - 15:00 thứ 5
7	Đi cà phê cóc	12:00 thứ 7 - 14:00 thứ 7
8	Xông hơi	20:30 thứ 7 - 20:45 thứ 7
10	Đọc truyện tranh	21:01 thứ 7 - 23:59 thứ 7

- Sắp xếp các hoạt động theo thứ tự tăng dần của thời gian kết thúc - N
- A={1}
- j=1
- For i =2 to N

```
    If S[i] >= F[j]
            A+=i
    j=i
```

2. Giới thiệu



Giải thuật tham lam là gì ?

- Là giải thuật tìm kiếm, lựa chọn giải pháp tối ưu cục bộ ở mỗi bước với hy vọng tìm được giải pháp tối ưu toàn cục.
- Là một kỹ thuật để giải quyết bài toán tối ưu hóa.
- Một số thuật toán mang ý tưởng tham lam như: Prim, Kruskal, Dijkstra,...

2. Giới thiệu

Các thành phần Greedy Approach:

- Một tập hợp các ứng viên (candidate), để từ đó tạo ra lời giải
- Một hàm lựa chọn, để theo đó lựa chọn ứng viên tốt nhất để bổ sung vào lời giải
- Một hàm khả thi (feasibility), dùng để quyết định nếu một ứng viên có thể được dùng để xây dựng lời giải
- Một hàm mục tiêu, ấn định giá trị của lời giải hoặc một lời giải chưa hoàn chỉnh
- Một hàm đánh giá, chỉ ra khi nào ta tìm ra một lời giải hoàn chỉnh.

```
getOptimal(Item, arr[], int n)
  1) Initialize empty result : result = {}
  2) While (All items are not considered)

      // We make a greedy choice to select
      // an item.
      i = SelectAnItem()

      // If i is feasible, add i to the
      // result
      if (feasible(i))
        result = result U i
  3) return result
```

2. Giới thiệu

Có hai yếu tố ảnh hưởng nhất đến quyết định của Greedy Approach đó là:

- Tính chất lựa chọn tham lam.
- Cấu trúc con tối ưu.

Độ phức tạp của thuật toán: Độ phức tạp của Greedy Approach khá dễ để tính toán.

2. Giới thiệu

Ưu điểm:

- Dễ tiếp cận với các bài toán và đặc biệt là các bài toán đồ thị.
- Độ phức tạp khá rõ ràng.
- Chạy nhanh hơn các thuật toán khác (quy hoạch động)

Nhược điểm:

- Với một số bài toán, giải thuật này có thể đưa ra kết quả không chính xác.

Khi nào dùng Greedy Approach ?

Khi bài toán thuộc lớp các bài toán tối ưu tổ hợp (là một trường hợp riêng của bài toán tối ưu)

Các bài toán con tối ưu có dạng như sau:

- Hàm f(x) được gọi là hàm mục tiêu, xác định trên một tập hữu hạn các phần tử D.
- Mỗi phần tử x thuộc D có dạng x = (x1,x2,x3,...xn) được gọi là một phương án.
- Tìm một phương án x0 thuộc D sao cho f(x) đạt max hoặc min trên D, x0 chính là phương án tối ưu.
- Tập D gọi là tập các phương án của bài toán.

3. Các bài toán điển hình

Knapsack Problem

Cho danh sách n món đồ có trọng lượng và giá trị, 1 cái túi chứa tối đa được trọng lượng là M. Hãy cho vào cái túi những món đồ sao cho giá trị được cao nhất mà cái túi chứa được.

Thuật toán :

Bước 1: Sắp xếp mảng theo giá trị value/weight.

Bước 2: Gán remaining_weight = M, bag_value = 0

Bước 3: Duyệt mảng n phần tử:

```
While (i < n):
    If (weight[i] <= remaining_weight):
        remaining_weight -= weight[i]
        bag_value += value[i]
```

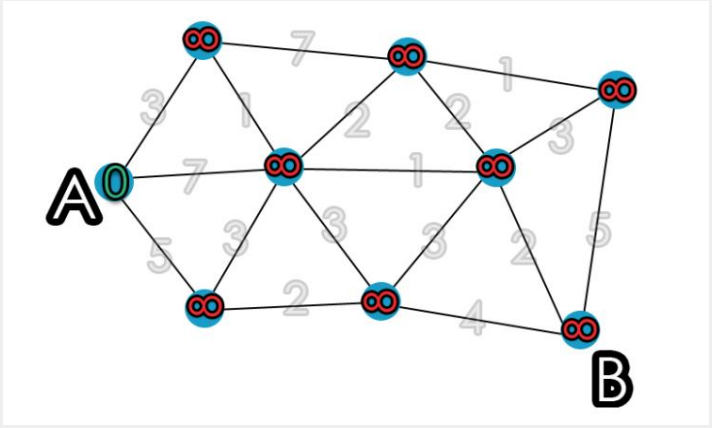
Bước 4: Khi không thể lấy thêm đồ nữa thì chương trình dừng lại và ta được kết quả là bag_value.

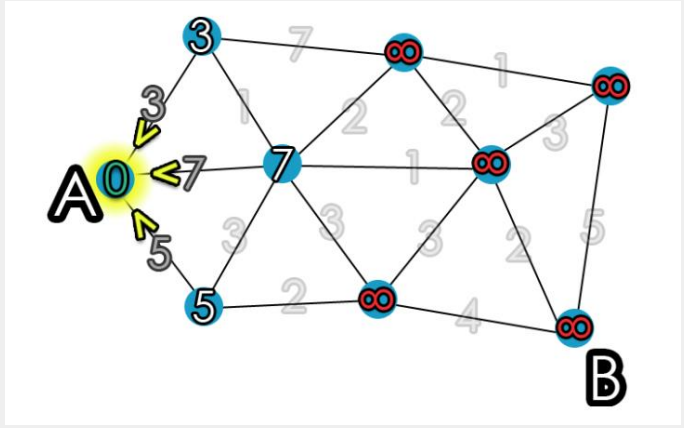
3. Các bài toán điển hình

Dijkstra's shortest path algorithm

Bước 1: Tạo một tập hợp sptSet để ghi lại các đỉnh có trong cây đường đi ngắn nhất. Ban đầu, tập hợp này là rỗng.

Bước 2: Gán giá trị khoảng cách cho tất cả các đỉnh trong đồ thị đầu vào. Khởi tạo tất cả các giá trị khoảng cách dưới dạng INFINITE. Gán giá trị khoảng cách là 0 cho đỉnh nguồn để nó được chọn đầu tiên.

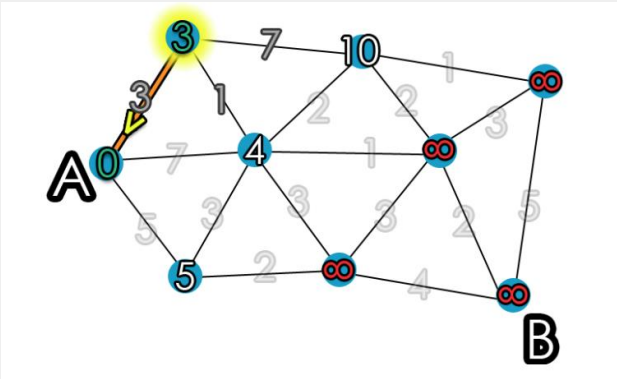


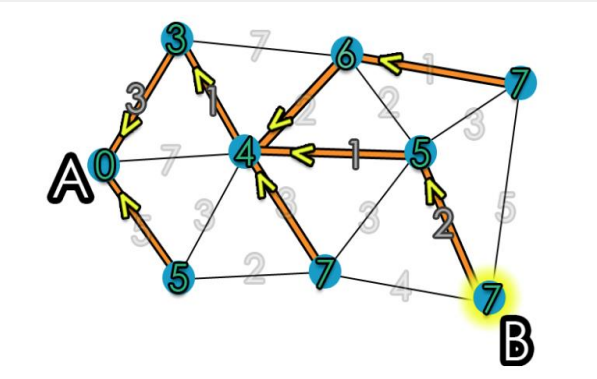


3. Các bài toán điển hình

Bước 3: Trong khi sptSet chưa bao gồm tất cả các đỉnh

- Chọn một đỉnh u không có trong sptSet và có giá trị khoảng cách nhỏ nhất.
- Bao gồm u vào sptSet.
- Cập nhật giá trị khoảng cách của tất cả các đỉnh liền kề của u. Để cập nhật các giá trị khoảng cách, hãy lặp lại qua tất cả các đỉnh liền kề. Đối với mọi đỉnh liền kề v, nếu tổng giá trị khoảng cách của u (từ nguồn) và trọng số của cạnh u-v, nhỏ hơn giá trị khoảng cách của v, thì cập nhật giá trị khoảng cách của v.





4. Bài tập về nhà

Knapsack Problem

Input:

- Dòng 1: số nguyên n là số món đồ có trong cửa hàng.
- Dòng 2: n số nguyên cách nhau bởi khoảng trắng lần lượt là giá trị của món đồ.
- Dòng 3: n số nguyên cách nhau bởi khoảng trắng lần lượt là cân nặng tương ứng.
- Dòng cuối cùng: một số nguyên là giới hạn cân nặng mà túi có thể chứa được

Output: số nguyên thể giá trị của các món đồ trong balo

Submit: 19522312@gm.uit.edu.vn

Deadline: 19/4

5. Tài liệu tham khảo

1. https://www.geeksforgeeks.org/greedy-algorithms/
2. https://text.123doc.net/document/3295824-tieu-luan-thuat-toan-tham-lam-greedy.htm
3. https://www.geeksforgeeks.org/greedy-algorithms-general-structure-and-applications/
4. https://vnoi.info/wiki/translate/topcoder/Greedy-is-Good.md

Thank you!

Do you have any questions?

19521956@gm.uit.edu.vn