TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Thuật toán ứng dụng

Nguyễn Khánh Phương

Computer Science department
School of Information and Communication technology
E-mail: phuongnk@soict.hust.edu.vn

Nội dung khóa học

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHƯƠNG 4

THUẬT TOÁN THAM LAM

TS. Nguyễn Khánh Phương
Trường Công nghệ thông tin và truyền thông

NỘI DUNG BÀI HỌC

1. Sơ đồ chung của thuật toán
2. Các ví dụ minh hoạ
   - 2.1. Bài toán đổi tiền
   - 2.2. Bài toán về các đoạn thẳng không giao nhau
   - 2.3. Bài toán cái túi
   - 2.4. Bài toán người du lịch
3. Chứng minh tính tối ưu của thuật toán

❑ NỘI DUNG BÀI HỌC

1. Sơ đồ chung của thuật toán

2. Các ví dụ minh hoạ

   2.1. Bài toán đổi tiền

   2.2. Bài toán về các đoạn thẳng không giao nhau

   2.3. Bài toán cái túi

   2.4. Bài toán người du lịch

3. Chứng minh tính tối ưu của thuật toán

❑ ĐẶC ĐIỂM CHUNG

Nói chung, các thuật toán tham lam và các bài toán có thể giải được nhờ chúng, có những đặc điểm chung sau:

- Ta phải giải quyết bài toán một cách tối ưu: Tìm lời giải với giá trị hàm mục tiêu lớn nhất (hoặc nhỏ nhất).
- Lời giải cần tìm có thể mô tả như là bộ gồm hữu hạn các thành phần thoả mãn các điều kiện nhất định.
- Có tập các ứng cử viên có thể lựa chọn làm các thành phần của lời giải.
- Xuất phát từ lời giải rỗng, thuật toán xây dựng lời giải của bài toán theo từng bước. Ở mỗi bước, sẽ chọn một phần tử từ tập ứng cử viên dựa trên tiêu chí "tham lam" đã xác định trước, và bổ sung nó vào lời giải đang có.

1. SƠ ĐỒ CHUNG

```
procedure Greedy( )
{  // Giả sử C là tập các ứng cử viên
    S = ∅;  // S lời giải xây dựng theo thuật toán
    while (C ≠ ∅ && !Solution(S)) {
      x ← Select(C);
      C = C \ x;
      if (Feasible (S ∪ x))  S = S ∪ x ;
    }
    if (Solution(S)) return S;
}
```

- Xuất phát từ lời giải rỗng, thuật toán xây dựng lời giải của bài toán theo từng bước, ở mỗi bước sẽ chọn một phần tử từ tập ứng cử viên và bổ sung vào lời giải hiện có.
- Hàm **Solution(S)** nhận biết tính chấp nhận được của lời giải S.
- Hàm **Select(C)** chọn từ tập C ứng cử viên có triển vọng nhất để bổ sung vào lời giải hiện có.
- Hàm **Feasible(S∪x)** kiểm tra tính chấp nhận được của lời giải bộ phận S ∪x.

NỘI DUNG

1. Sơ đồ chung của thuật toán

2. Các ví dụ minh hoạ

   2.1. Bài toán đổi tiền

   2.2. Bài toán về các đoạn thẳng không giao nhau

   2.3. Bài toán cái túi

   2.4. Bài toán người du lịch

3. Chứng minh tính tối ưu của thuật toán

2.1. BÀI TOÁN ĐỔI TIỀN

Cho các mệnh giá tiền xu: 1, 5, 10, 25, 100 cent.

Yêu cầu: hãy tìm cách trả ***X*** cent tiền thừa cho khách sao cho số lượng đồng xu đưa khách là ít nhất.

1 cent | 5 cent | 10 cent | 25 cent | 100 cent

2.1. BÀI TOÁN ĐỔI TIỀN: các mệnh giá tiền xu: 1, 5, 10, 25, 100 cent.

- **Thuật toán người thu ngân (Cashier algorithm)**
  - Ở mỗi bước lặp: thêm đồng tiền với mệnh giá lớn nhất mà không vượt quá số tiền phải trả

100 cent | 25 cent | 10 cent | 1 cent

Ví dụ: Trả khách hàng số tiền X = 289 cent

2.1. BÀI TOÁN ĐỔI TIỀN

- **Thuật toán người thu ngân (Cashier algorithm)**
  - Ở mỗi bước lặp: thêm đồng tiền với mệnh giá lớn nhất mà không vượt quá số tiền phải trả

```
procedure CASHIERS(X, c1, c2,..,cn)
{  //đổi X cent dùng các mệnh giá c1 < c2 < ..< cn
   S ← Ø //tập chứa các đồng xu đem trả cho khách
   while ( X > 0) {
      k ← đồng xu có mệnh giá lớn nhất ck sao cho ck ≤ X
      if (không tìm được k) return "no solution"
      else {
        X ←  X - ck
        S ←  S ∪ {k}
      }
   }
}
```

Thuật toán Cashier có phải tối ưu ?

2.1. BÀI TOÁN ĐỔI TIỀN: Tính chất của lời giải tối ưu

- **Tính chất 1: Số lượng đồng mệnh giá 1 cent trong lời giải tối ưu phải ≤ 4**

Chứng minh. Nếu có > 4 đồng mệnh giá 1 cent, thì cứ 5 đồng xu mệnh giá 1 cent ta sẽ thay bằng 1 đồng xu mệnh giá 5 cent, để thu được số lượng đồng xu ít hơn. Do đó, trong lời giải tối ưu, số lượng đồng xu mệnh giá 1 cent luôn **≤ 4**

- **Tính chất 2: Số lượng đồng mệnh giá 5 cent trong lời giải tối ưu phải ≤ 1**

- **Tính chất 3: Số lượng đồng mệnh giá 25 cent trong lời giải tối ưu phải ≤ 3**

2.1. BÀI TOÁN ĐỔI TIỀN: Chứng minh tính tối ưu

Tính chất 4: Tổng số đồng mệnh giá 5 cent và mệnh giá 10 cent trong lời giải tối ưu phải ≤ 2

Chứng minh. Phản chứng, giả sử có > 2 đồng xu

- 3 đồng mệnh giá 10 cent + 0 đồng mệnh giá 5 cent ➔ thay bằng 1 đồng mệnh giá 25 cent + 1 đồng mệnh giá 5 cent
- 2 đồng mệnh giá 10 cent + 1 đồng mệnh giá 5 cent ➔ thay bằng 1 đồng mệnh giá 25 cent
- 1 đồng mệnh giá 10 cent + 2 đồng mệnh giá 5 cent ➔ trường hợp này không xảy ra vì theo tính chất 2 (Số lượng đồng mệnh giá 5 cent ≤ 1)

➔ Điều giả sử là sai ➔ đpcm

2.1. BÀI TOÁN ĐỔI TIỀN: Tính chất của lời giải tối ưu

Mệnh đề: Thuật toán tham lam người thu ngân (NTN) là tối ưu cho bài toán đổi tiền với các mệnh giá 1, 5, 10, 25, 100

Chứng minh. (Quy nạp theo số tiền X cent)

- Trường hợp cơ sở: X = 1 cent : thuật toán NTN lấy 1 đồng xu mệnh giá 1 cent và đó cũng là lời giải tối ưu
- Giả sử thuật toán NTN cho lời giải tối ưu khi cần trả < X cent
- Cần chứng minh đúng với X cent:
  - Xét cách tối ưu để đổi $c_k \le X < c_{k+1}$: thuật toán NTN lấy đồng xu mệnh giá c_k
  - Ta khẳng định rằng lời giải tối ưu cũng phải lấy đồng xu mệnh giá c_k
    - Nếu không, ta cần chọn từ các đồng xu mệnh giá $c_1, c_2, \dots, c_{k-1}$ để lấy cho đủ X cent. Bảng dưới đây cho thấy không tìm được lời giải tối ưu nếu làm như vậy:

1 cent (P)
5 cent (N)
10 cent (D)
25 cent (Q)

k	C_k	Số lượng đồng xu trong lời giải tối ưu	Nếu chỉ dùng các đồng mệnh giá $c_1, c_2, .., c_{k-1}$ thì giá trị xu lớn nhất trong bất kỳ lời giải tối ưu đạt được là
1	1	P ≤ 4 (theo tính chất 1)	
2	5	N ≤ 1 (theo tính chất 2)	1 cent * 4 đồng xu = 4 cent < 5
3	10	N + D ≤ 2 (theo tính chất 4)	Nếu chỉ dùng các đồng 1 cent và 5 cent: 1 cent * 4 + 5 cent * 1 = 9 cent < 10
4	25	Q ≤ 3 (theo tính chất 3)	Nếu chỉ dùng các đồng 1 cent, 5 cent, 10 cent: 1 cent * 4 + 10cent*2 = 24 cent < 25
5	100	Không giới hạn số lượng	Nếu chỉ dùng các đồng 1 cent, 5 cent, 10 cent, 25 cent: 24 + 25 cent * 3 = 99 cent < 100

  - Bài toán X cent ban đầu tiếp tục được đưa về bài toán X- c_k cent, mà theo giả thiết quy nạp, ta đã có lời giải tối ưu bằng thuật toán NTN

2.1. BÀI TOÁN ĐỔI TIỀN: Chứng minh tính tối ưu

Câu hỏi. Thuật toán tham lam NTN có thể áp dụng cho bất kỳ tập mệnh giá nào?

Trả lời: KHÔNG

- Thuật toán tham lam NTN không cho lời giải tối ưu trên tập mệnh giá 1, 10, 21, 34, 70, 100, 350, 1225, 1500.
  - Ví dụ: đổi 140 cent
    - Thuật toán tham lam cho lời giải 140 = 100 + 34 + 1 + 1 + 1 + 1 +1 + 1
    - Tối ưu: 140 = 70 + 70
- Thuật toán tham lam NTN thậm chí không tìm được lời giải nếu $c_1 > 1$
  - Ví dụ: tập 3 mệnh giá 7, 8, 9 cent. Cần đổi 15 cent
    - Tham lam: 15 = 9 + ???
    - Tối ưu: 15 = 7 + 8

2.1. BÀI TOÁN ĐỔI TIỀN

Thuật toán người thu ngân (Cashier algorithm):

- Ở mỗi bước lặp: thêm đồng tiền với mệnh giá lớn nhất mà không vượt quá số tiền phải trả

```
/*Can doi X cent su dung cac dong xu menh gia c[0] < c[1]<...<c[n-1]
sao cho so luong dong xu dung it nhat
Mang S: phan tu S[i] = so luong dong xu menh gia c[i] se dung de tra X cent
*/
void Cashier(int X, int c[], int S[], int n)
{
   for (int i = n-1; i >=0; i--)
   {
        int soluong = X / c[i];
        if (soluong > 0) S[i] = soluong;
        X %= c[i];
   }
}
```

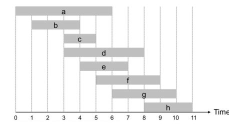
```
int main()
{
    int c[] = {1, 5, 10, 25, 100};
    int S[5] = {0};
    int n = 5;  int X = 383;
    Cashier(X,c,S,n);
    cout<<"Cach doi "<<X<< " cent: "<<endl;
    for (int i = 0; i<n;i++)
     if (S[i] > 0)
        cout<<S[i]<<" dong xu menh gia "<<c[i]<<endl;
    return 0;
}
```

❑ NỘI DUNG BÀI HỌC

2.2. BÀI TOÁN VỀ CÁC ĐOẠN THẲNG KHÔNG GIAO NHAU

- **Đầu vào:** Cho họ các đoạn thẳng:

  $C = \{(s_1, f_1), (s_2, f_2), \ldots, (s_n, f_n)\}$
- **Đầu ra:** Tập các đoạn không giao nhau có lực lượng lớn nhất



Ví dụ thực tế: Bài toán xếp thời gian biểu cho các hội thảo, Bài toán lựa chọn hành động (Activity Selection), Bài toán phục vụ khách hàng trên một máy...

➔ Bài toán xếp thời gian biểu:

- Có n cuộc họp, cuộc họp j bắt đầu (Start) từ s_j và kết thúc (Finish) ở f_j.
- Mục đích: Hãy xác định số lượng cuộc họp lớn nhất có thể lên lịch họp nếu chỉ có 1 phòng họp.

2.2. BÀI TOÁN VỀ CÁC ĐOẠN THẲNG KHÔNG GIAO NHAU

- **Thuật toán tham lam:**
    - Lần lượt xét các cuộc họp theo một trình tự nào đó.
    - Xếp cuộc họp vào phòng họp sao cho không có cuộc họp nào bị giao nhau.
- **4 qui tắc:**
    - [Bắt đầu sớm xét trước] Xét các cuộc họp theo thứ tự s_j tăng dần.
    - **[Kết thúc sớm xét trước] Xét các cuộc họp theo thứ tự f_j tăng dần.**
    - [Ngắn hơn xét trước] Xét các cuộc họp theo thứ tự $f_j - s_j$ tăng dần.
    - [Ít xung đột hơn xét trước] Với mỗi cuộc họp j, tính c_j là số lượng cuộc họp xung đột *(diễn ra cùng lúc)* với nó. Lập lịch theo thứ tự tăng dần của c_j.

Greedy1 [Bắt đầu sớm xét trước - Earliest start time]

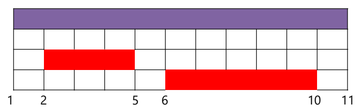
```
procedure Greedy( )
{
    S = ∅;  //S là tập các cuộc họp sẽ được chọn lên lịch họp trong phòng họp
    C = <Sắp xếp n cuộc họp đã cho theo thứ tự tăng dần của thời điểm bắt đầu cuộc họp>
    while (C ≠ ∅)
    {
        (s_c ,f_c ) ← cuộc họp đầu tiên có trong C;
         C = C \ (s_c ,f_c );
         if  <(s_c ,f_c ) không giao với bất cứ cuộc họp nào trong S>
              S = S ∪ (s_c , f_c );
    }
    <S là tập cần tìm>
}
```

2.2. BÀI TOÁN VỀ CÁC ĐOẠN THẲNG KHÔNG GIAO NHAU

Greedy1. [Bắt đầu sớm xét trước - Earliest start time]

- Sắp xếp các cuộc họp theo thứ tự tăng dần của thời điểm bắt đầu họp
- Bắt đầu từ tập *S* là tập rỗng, ta lần lượt xét các cuộc họp trong danh sách theo thứ tự đã sắp xếp và bổ sung cuộc họp đang xét vào *S* nếu nó không bị giao với bất kì cuộc họp nào có trong *S*.

Phản ví dụ cho Greedy1:



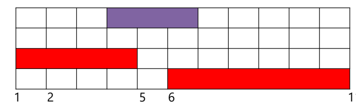
2.2. BÀI TOÁN VỀ CÁC ĐOẠN THẲNG KHÔNG GIAO NHAU

Greedy1. [Bắt đầu sớm xét trước - Earliest start time]

Greedy3. [Ngắn hơn xét trước - Shortest interval]

Khác với Greedy1, trong thuật toán Greedy3, ở mỗi bước ta chọn cuộc họp diễn ra trong khoảng thời gian ngắn nhất bổ sung vào *S*.

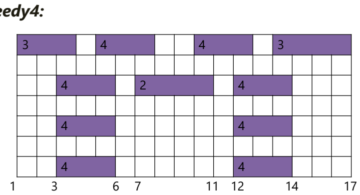
Phản ví dụ cho Greedy3:



2.2. BÀI TOÁN VỀ CÁC ĐOẠN THẲNG KHÔNG GIAO NHAU

Greedy4. [Ít xung đột hơn xét trước –Fewest conflict] Với mỗi cuộc họp *j*, tính c_j là số lượng cuộc họp **xung đột (diễn ra cùng lúc)** với nó. Lập lịch theo thứ tự tăng dần của c_j.

Phản ví dụ cho Greedy4:



2.2. BÀI TOÁN VỀ CÁC ĐOẠN THẲNG KHÔNG GIAO NHAU

- **Greedy1.** [Bắt đầu sớm xét trước] Tìm được phản ví dụ không cho lời giải tối ưu
- **Greedy3.** [Ngắn hơn xét trước] Tìm được phản ví dụ không cho lời giải tối ưu
- **Greedy4.** [Ít mâu thuẫn xét trước] Tìm được phản ví dụ không cho lời giải tối ưu
- **Greedy2.** [Kết thúc sớm xét trước – Earliest finish time]

Mệnh đề. *Thuật toán Greedy2 cho lời giải tối ưu của bài toán về các đoạn thẳng không giao nhau.*

Mệnh đề. ***Thuật toán Greedy2 cho lời giải tối ưu của bài toán về các đoạn thẳng không giao nhau.***

Chứng minh. Giả sử ngược lại, thuật toán Greedy2 không cho lời giải tối ưu. Khi đó cần tìm được bộ dữ liệu đầu vào C sao cho Greedy2 không cho lời giải tối ưu.

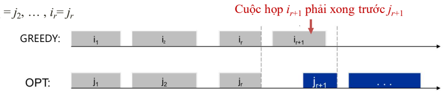
Gọi GREEDY = < $i_1, i_2, \ldots, i_k$ > là tập các cuộc họp của lời giải cho bởi Greedy2 khi chạy trên bộ dữ liệu đầu vào C.

Gọi OPT = < $j_1, j_2, \ldots, j_m$ > là tập các cuộc họp của lời giải tối ưu có số cuộc họp chung với GREEDY là lớn nhất.

Gọi i_{r+1} là cuộc họp đầu tiên trong GREEDY không có mặt trong OPT. Cuộc họp này là tồn tại, vì nếu trái lại thì:

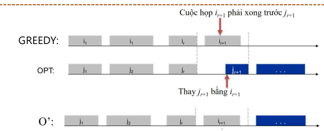
- hoặc GREEDY ≡ OPT (mâu thuẫn với tính không tối ưu của Greedy2)
- hoặc GREEDY là tập con thực sự của OPT (trái với hoạt động của thuật toán, bởi vì khi đó thuật toán phải chọn cả cuộc họp cuối cùng trong OPT).

Tức là ta có: $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$



Mệnh đề. ***Thuật toán Greedy2 cho lời giải tối ưu của bài toán về các đoạn thẳng không giao nhau.***

- Xét O′ = OPT \ { j_{r+1} } ∪ { i_{r+1} }.



Rõ ràng:

- O′ gồm các cuộc họp không giao nhau, bởi vì i_{r+1} không giao với bất cứ cuộc họp nào ở bên trái nó trong O′ cũng như không giao với bất cứ cuộc họp nào ở bên phải nó trong O′ (do mút phải của i_{r+1} nhỏ hơn mút phải của j_{r+1} và j_{r+1} không giao với bất cứ đoạn nào ở bên phải i_{r+1} trong O′).
- Do O′ có cùng lực lượng với OPT, nên O′ cũng là tối ưu.
- O′ giống với GREEDY hơn là OPT (O′ có nhiều đoạn chung với GREEDY hơn là OPT).

➔ Mâu thuẫn với giả thiết là OPT là lời giải tối ưu giống với GREEDY nhất

➔ đpcm

Mệnh đề. ***Thuật toán Greedy2 cho lời giải tối ưu của bài toán về các đoạn thẳng không giao nhau.***

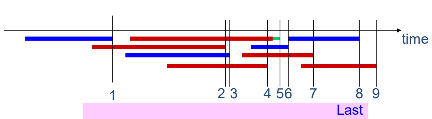
Greedy2 [Kết thúc sớm xét trước]

```
Algorithm Greedy2(C=(s1,f1), … , (sn,fn))
// Thời gian: O(n log n)
{
   Sắp xếp C theo thứ tự tăng dần của fi
   Không giảm tổng quát coi là  f1 ≤ f2 ≤… fn
   S ← ∅;
   Last ← -∞;
   for  k = 1 to n
     if  (Last ≤ sk)
     {
       S = S ∪ {(sk,fk)}
       Last ← fk
     }
   return S;
}
```

- **Greedy2.** [Kết thúc sớm xét trước]

Sắp xếp các cuộc họp theo thứ tự tăng dần của thời điểm kết thúc.

Bắt đầu từ tập *S* là tập rỗng, ta lần lượt xét các cuộc họp trong danh sách theo thứ tự đã sắp xếp và bổ sung cuộc họp đang xét vào *S* nếu nó không diễn ra cùng thời gian với bất cứ cuộc họp nào trong *S*.



❑ NỘI DUNG BÀI HỌC

1. Sơ đồ chung của thuật toán
2. Các ví dụ minh hoạ
   - 2.1. Bài toán đổi tiền
   - 2.2. Bài toán về các đoạn thẳng không giao nhau
   - **2.3. Bài toán cái túi**
   - 2.4. Bài toán người du lịch
3. Chứng minh tính tối ưu của thuật toán

2.3 BÀI TOÁN CÁI TÚI (Knap sack)

- Có n loại đồ vật.
- Đồ vật loại i có
  - trọng lượng w_i và
  - giá trị sử dụng là c_i (i = 1, 2,..., n) .

- Cần chất các đồ vật này vào một cái túi có trọng lượng là b sao cho tổng giá trị sử dụng của các đồ vật chất trong túi là lớn nhất.

Ký hiệu S = {1, 2, ..., n} tập chỉ số các đồ vật. Bài toán đặt ra là:

Tìm $I \subset S$ sao cho $\sum_{i \in I} w_i \le b, \quad \sum_{i \in I} c_i \to \max$

2.3 BÀI TOÁN CÁI TÚI (Knap sack)

Greedy 1 [Giá trị lớn hơn xét trước]:

- Sắp xếp các đồ vật theo thứ tự không tăng của giá trị.
- Lần lượt xét các đồ vật theo thứ tự đã sắp, và chất đồ vật đang xét vào túi nếu như dung lượng còn lại của cái túi đủ chứa nó (tức là tổng trọng lượng của các đồ vật đã xếp vào túi và trọng lượng của đồ vật đang xét là không vượt quá b).

- Phản ví dụ cho Greedy1 là bộ dữ liệu sau:
  - Trọng lượng cái túi b = 19.

Đồ vật	1	2	3
Giá trị	20	16	8
Trọng lượng	14	6	10

- Greedy1 cho lời giải: I_1 = {1} với giá trị 20.
- Trong khi đó, ta có lời giải tốt hơn là I* = {2,3} với giá trị 24

2.3 BÀI TOÁN CÁI TÚI (Knap sack)

Greedy 2 [Nhẹ hơn xét trước]:

- Sắp xếp các đồ vật theo thứ tự không giảm của trọng lượng.
- Lần lượt xét các đồ vật theo thứ tự đã sắp, và chất đồ vật đang xét vào túi nếu như dung lượng còn lại của cái túi đủ chứa nó.

- Phản ví dụ cho Greedy2 là bộ dữ liệu sau:
  - Trọng lượng cái túi b = 11.

Đồ vật	1	2	3
Giá trị	10	16	28
Trọng lượng	5	6	10

- Greedy2 cho lời giải: I_2 = {1, 2} với giá trị 26
- Trong khi đó, ta có lời giải tốt hơn là I* = {3} với giá trị 28

2.3 BÀI TOÁN CÁI TÚI (Knap sack)

Greedy 3 [Giá trị trên 1 đơn vị trọng lượng lớn hơn xét trước]:

- Sắp xếp các đồ vật theo thứ tự không tăng của giá trị một đơn vị trọng lượng (c_i /w_i), nghĩa là
$$\frac{c_{i_1}}{w_{i_1}} \ge \frac{c_{i_2}}{w_{i_2}} \ge \ldots \ge \frac{c_{i_n}}{w_{i_n}}$$
- Lần lượt xét các đồ vật theo thứ tự đã sắp, và chất đồ vật đang xét vào túi nếu như dung lượng còn lại của cái túi đủ chứa nó.

- Phản ví dụ cho Greedy3 là bộ dữ liệu sau:
  - Trọng lượng cái túi $b \ge 2$

Đồ vật	1	2
Giá trị	10	10b-1
Trọng lượng	1	b

$$\frac{c_1}{w_1} = \frac{10}{1} \ge \frac{10b-1}{b} = \frac{c_2}{w_2}$$

- Greedy3 cho lời giải: I_3 = {1} với giá trị 10
- Trong khi đó, ta có lời giải tốt hơn là I* = {2} với giá trị 10b-1 > 10

2.3 BÀI TOÁN CÁI TÚI (Knap sack)

Greedy 4:

- Gọi I_j là lời giải thu được theo thuật toán Greedy$j$, j =1, 2, 3. Gọi I_4 là lời giải đạt
$$\max\{\sum_{i\in I_1} c_i, \sum_{i\in I_2} c_i, \sum_{i\in I_3} c_i\}$$

- Phản ví dụ cho Greedy4 là bộ dữ liệu sau:
  - Trọng lượng cái túi b = 11

Đồ vật	1	2	3	4
Giá trị c	9	10	18	27
Trọng lượng w	4	5	6	10
c/w	2.25	2	3	2.7

Greedy1: 27

Greedy2: 19

Greedy3: 27

Better: 28 với I_0={2, 3}

Định lý. *Lời giải I_4 thoả mãn bất đẳng thức* $\sum_{i\in I_4} c_i \geq \frac{1}{2} OPT$

với OPT là lời giải tối ưu của bài toán

❑ NỘI DUNG BÀI HỌC

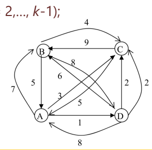
1. Sơ đồ chung của thuật toán
2. Các ví dụ minh hoạ
   - 2.1. Bài toán đổi tiền
   - 2.2. Bài toán về các đoạn thẳng không giao nhau
   - 2.3. Bài toán cái túi
   - **2.4. Bài toán người du lịch**
3. Chứng minh tính tối ưu của thuật toán

2.4 BÀI TOÁN NGƯỜI DU LỊCH(Traveling Salesman Problem – TSP)

- **Phát biểu bài toán**: Một người du lịch muốn đi tham quan n thành phố 1, 2, .., n. Xuất phát từ thành phố 1, người du lịch cần *đi qua tất cả các thành phố còn lại, mỗi thành phố đúng một lần, rồi quay trở lại thành phố xuất phát* 1. Biết c_{ij} là chi phí đi từ thành phố i đến thành phố j (i, j = 1, 2,..., n), tìm hành trình với tổng chi phí là nhỏ nhất.
- Greedy [Gần nhất thăm trước]: Hành trình người du lịch là: (1, a_2, ..., a_{n-1}, 1) trong đó $a_k \in \{2,\ldots,n-1\}$. Tại mỗi bước k (k = 2,...,n-1) ta cần chọn một đỉnh a_k:
  - a_k được chọn trong số các đỉnh chưa thăm (tức là $a_k \neq a_i$, i = 2,..., k-1);
  - a_k là đỉnh gần a_{k-1} nhất trong số các đỉnh chưa thăm.
- Phản ví dụ cho Greedy: Thành phố xuất phát là A,

Greedy cho lời giải là (A, D, C, B, A) chi phí = 17;

nhưng lời giải tối ưu là: (A, C, D, B, A) chi phí = 16



❑ NỘI DUNG BÀI HỌC

1. Sơ đồ chung của thuật toán
2. Các ví dụ minh hoạ
   - 2.1. Bài toán đổi tiền
   - 2.2. Bài toán về các đoạn thẳng không giao nhau
   - 2.3. Bài toán cái túi
   - 2.4. Bài toán người du lịch
3. **Chứng minh tính tối ưu của thuật toán**

3. CHỨNG MINH TÍNH TỐI ƯU CỦA THUẬT TOÁN THAM LAM

- Để chỉ ra thuật toán không cho lời giải tối ưu, ta chỉ cần đưa ra một phản ví dụ (Một bộ dữ liệu mà thuật toán không cho lời giải tối ưu).
- Chứng minh tính tối ưu của thuật toán khó hơn nhiều.

Lập luận biến đổi (Exchange Argument)

Giả sử cần chứng minh thuật toán A cho lời giải tối ưu.

- Gọi A(I) là lời giải tìm được bởi thuật toán A đối với bộ dữ liệu I, còn O là lời giải tối ưu của bài toán với bộ dữ liệu này.
- Ta cần tìm cách xây dựng phép biến đổi β để biến đổi lời giải tối ưu O thành lời giải O' sao cho:
  - O' cũng tốt không kém gì O (nghĩa là O' vẫn tối ưu).
  - O' **giống** với A(I) nhiều hơn O.
- Khi đó, sử dụng phép biến đổi β**,** ta có thể chứng minh tính tối ưu của thuật toán A bằng cách sử dụng một trong hai sơ đồ chứng minh sau:
  - Chứng minh bằng phản chứng
  - Chứng minh trực tiếp

Lập luận biến đổi: Chứng minh bằng phản chứng

Giả sử A không cho lời giải tối ưu. Khi đó, phải tồn tại bộ dữ liệu I sao cho A(I) khác với lời giải tối ưu của bài toán.

- Gọi O là lời giải tối ưu giống với A(I) nhất.
- Vì A(I) không phải là lời giải tối ưu nên A(I) ≠ O.
- Dùng phép biến đổi β, ta có thể biến đổi O thành O' sao cho: O' vẫn tối ưu và O' giống với A(I) hơn ➔ Mâu thuẫn giả thiết O là lời giải tối ưu giống với A(I) nhất.

Vậy điều giả sử A không cho lời giải tối ưu là sai ➔ thuật toán A cho lời giải tối ưu (đpcm).

Lập luận biến đổi: Chứng minh trực tiếp

Gọi O là lời giải tối ưu, sử dụng phép biến đổi β, ta biến đổi O thành lời giải tối ưu O' giống với A(I) hơn là O:

- Nếu O' =A(I) thì A(I) chính là phương án tối ưu (đpcm).
- Ngược lại, lặp lại phép biến đổi β đối với O' để thu được lời giải tối ưu O'' giống với A(I) hơn là O'.
- ….
- Cứ thế, ta thu được dãy O', O'', O''',… , và mỗi phần tử sau giống A(I) hơn phần tử trước nó

➔ dãy này phải kết thúc tại A(I). Vậy A(I) cũng là tối ưu, tức là thuật toán A cho lời giải tối ưu (đpcm).

KẾT LUẬN

- Thuật toán tham lam dễ đề xuất, và thông thường không đòi hỏi nhiều thời gian tính. Tuy nhiên, thuật toán dạng này thường không cho kết quả tối ưu.
- Với các bài toán tối ưu ứng dụng trong thực tế có độ khó cao, thuật toán tham lam được áp dụng để cho lời giải chấp nhận được.

Chúc các bạn học tốt !