THUẬT TOÁN THAM LAM (GREEDY)

- Giới thiệu
- Một số ví dụ
- Bài tập

Tìm giải pháp gần tối ưu

- Quay lui có thể giúp tìm các nghiệm tối ưu thông qua việc xét toàn bộ các phương án
- Nhánh cận giúp giảm bớt các phương án không cần thiết
- Đối với các bài toán phức tạp, quay lui/nhánh cận không thực hiện được hoặc rất lâu
  - Thuật toán tham ăn được sử dụng tìm nghiệm gần tối ưu
  - Thời gian thực hiện nhanh, độ phức tạp nhỏ

Tìm giải pháp gần tối ưu

- Các nghiệm của bài toán dạng 1 dãy các đối tượng được chọn dần từng bước (x_1, x_2, ..., x_i, ...)
  - x_i được chọn từ tập S_i
- Mỗi nghiệm được xác định độ tốt bằng hàm f(X)
  - Mục tiêu cần tìm nghiệm có giá trị hàm f(X) càng lớn (hoặc càng nhỏ) càng tốt.

Các bước thực hiện

- Xây dựng nghiệm X dần từng bước
  - Giả sử xây dựng được k-1 thành phần (x_1, x_2, ..., x_{k-1})
  - Tại bước k, chọn x_k "tốt nhất" trong tập ứng viên S_k
- Các bước lặp lại cho đến khi xây dựng xong hết n thành phần của nghiệm
  - Khái niệm "tốt nhất" trong bối cảnh cục bộ
  - Có thể cho nghiệm tối ưu hoặc không tìm ra nghiệm (mặc dù có nghiệm)

Cài đặt thuật toán

```
void Greedy(){
    X = {}
    i = 0;
    while (i<n){
          i = i + 1
          <Xác định tập ứng viên S_i>
          <Chọn ứng viên tốt nhất trong S_i>
          x_i = select(S_i)
     }
 }
```

VD 1: Bài toán người du lich

- N thành phố (1, 2, ..., N). Thành phố *i* nối với thành phố *j* bằng tuyến đường có khoảng cách *c[i,j] = c[j,i]*.
- Người du lịch xuất phát từ TP 1, muốn đi thăm tất cả TP (mỗi TP đúng 1 lần) và cuối cùng quay về TP 1.
- Tìm hành trình để người đó phải đi với tổng chi phí càng ít càng tốt.

Bài toán người du lịch: Greedy

- Hành trình cần tìm có dạng (x_1=1, x_2, ..., x_N, X_{N+1}=1): $x_i \neq x_j$ và (x_i, x_{i+1}) có đường đi trực tiếp
- Tại bước i, chọn x_i từ 1 trong các thành phố có thể tới từ x_{i-1} và có khoảng cách gần x_{i-1} nhất
  - Mảng used đánh dấu các thành phố các đi qua
  - Biến *b* và *xi* để lưu khoảng cách và thành phố gần nhất có thể đến từ x_{i-1}

Bài toán người du lịch: Greedy

```
void Greedy(){
      x[0] = 1; used[0] = 1;
      i = 0;
      while (i<n){
            i++;
            b = INT_MAX;
            for (j=1, j<n; j++)
                  if used[j]==0 and c[x[i-1],j]<b){
                        b =  c[x[i-1],j]; xi = j;
                  }
            x[i] = xi; used[xi]=1;
       }
```

VD 2: Rút tiền ATM

- Tìm phương án rút tiền với số tờ tiền ít
  - Giả sử đã xây dựng cách trả tiền đến bước k (x_1, x_2, ..., x_k), số tiền còn phải trả *sum*
  - Sắp xếp các tờ tiền theo giá trị giảm dần.
  - Xét các tờ tiền chưa sử dụng từ lớn đến nhỏ, nếu tờ nào có giá trị <= *sum* thì sử dụng luôn tờ tiền đó
  - Quá trình lặp lại cho đến khi số tiền còn phải trả = 0 hoặc không tìm được tờ tiền tiếp theo

Rút tiền ATM: Greedy

```
void Greedy(){
      i = 0; sum = S;
      while (i<n){
            if (t[i] <= sum){
                  x[i] = 1;//dùng tờ i
                  sum = sum - t[i];
            }
            if (sum == 0) break;
            else i++;
      }
}
```

Bài tập

- Viết chương trình thực hiện ví dụ:
  - Bài toán rút tiền ATM với số tờ ít
  - Bài toán người du lịch với tổng quãng đường đi nhỏ
- Sắp xếp công việc:
  - N hành động
  - <Si, Fi>: Thời gian bắt đầu/Thời gian kết thúc
  - Tìm phương án thực hiện nhiều nhất các hành động sao cho hệ không xảy ra mâu thuẫn