CHIẾN LƯỢC THIẾT KẾ TRỰC TIẾP VÀ VÉT CẠN

- Các đặc trưng cơ bản
- Các ví dụ minh họa

CÁC ĐẶC TRƯNG CƠ BẢN

- Giải thuật được thiết kế một cách trực tiếp (straightforward) dựa trên định nghĩa và các khái niệm liên quan của bài toán
- Là chiến lược dễ dàng áp dụng và được lựa chọn đầu tiên

CÁC ĐẶC TRƯNG CƠ BẢN

- Được áp dụng cho một lớp rất rộng các bài toán
- Chi phí thiết kế rẽ, thích hợp cho các bài toán kích thước nhỏ

CÁC ĐẶC TRƯNG CƠ BẢN

- Có thể sinh ra một số giải thuật có độ phức tạp khá lớn (hoặc rất lớn)
- Là cơ sở để đề xuất các giải thuật mới
- Chiến lược vét cạn (exhaustive) là trường hợp đặc biệt của chiến lược trực tiếp (brute force)

CÁC VÍ DỤ

- Bài toán tính tổng $S=1^2+2^2+...+n^2$
- Giải thuật sắp xếp chọn trực tiếp (Selection Sort)
- Giải thuật tìm kiếm tuần tự (Sequential Search)
- Bài toán so trùng mẫu của chuỗi ký tự (String Matching)
- Bài toán tìm cặp điểm gần nhất (Closest-Pair)
- Bài toán người đi du lịch (Traveling Salesman)

BÀI TOÁN TÍNH TỔNG $S=1^2+2^2+...+N^2$

- Chiến lược thiết kế giải thuật trực tiếp sử dụng kỹ thuật cộng dồn các bình phương của các số liên tiếp
- Giải thuật sử dụng một vòng lặp

BÀI TOÁN TÍNH TỔNG $S=1^2+2^2+\ldots+N^2$

ALGORITHM Sum(n)

```
1   S ←0
2   for i ←1 to n do
3         S ←S+i*i
4   return S
```

BÀI TOÁN TÍNH TỔNG $S=1^2+2^2+...+N^2$

- Phép toán cơ bản là nhân
- Gọi c là thời gian của phép toán cơ bản

$T(n) = nc = \Theta(n)$

Lưu ý có thể dùng chiến lược biến đổi để trị để đưa về $\Theta(1)$

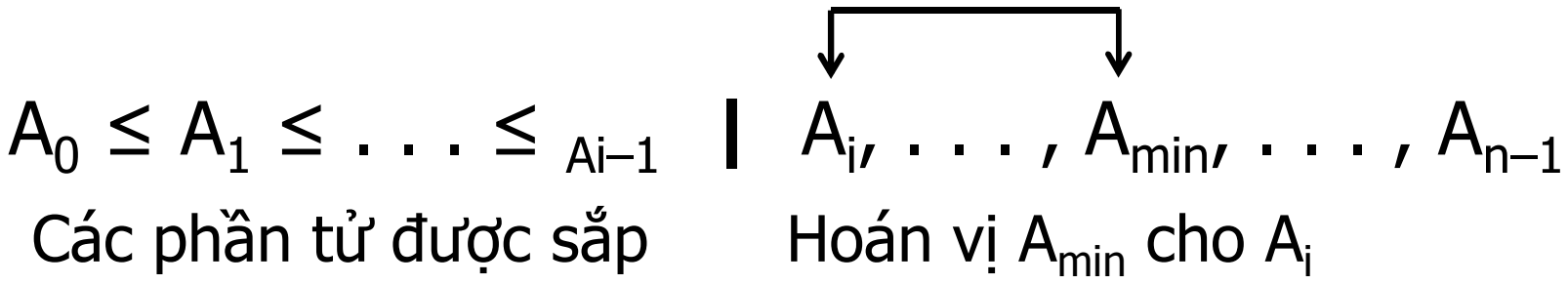
$S=1^2+2^2+...+n^2=n(n+1)(2n+1)/6$

SẮP XẾP CHỌN TRỰC TIẾP

- Chọn trực tiếp phần tử nhỏ nhất trong mảng và hoán đổi cho phần tử đầu tiên của mảng
- Thực hiện tương tự cho mảng có n-1 phần tử còn lại với chỉ số phần tử đầu tiên i=1

SẮP XẾP CHỌN TRỰC TIẾP

- Tiếp tục quá trình cho đến khi mảng cần hoán vị chỉ còn một phần tử



- Cần hai vòng lặp: Xác định phần tử thứ i của dãy chưa được sắp và tìm phần tử nhỏ nhất trong đó

SẮP XẾP CHỌN TRỰC TIẾP

ALGORITHM SelectionSort(A[0..n – 1])
//Input: An array A[0..n – 1] of orderable elements
//Output: Array A[0..n – 1] sorted in nondecreasing order
1 **for** i ←0 **to** n – 2 **do**
2 min←i
3 **for** j ←i + 1 **to** n – 1 **do**
4 **if** A[j]<A[min] min←j
5 swap A[i] and A[min]

SẮP XẾP CHỌN TRỰC TIẾP

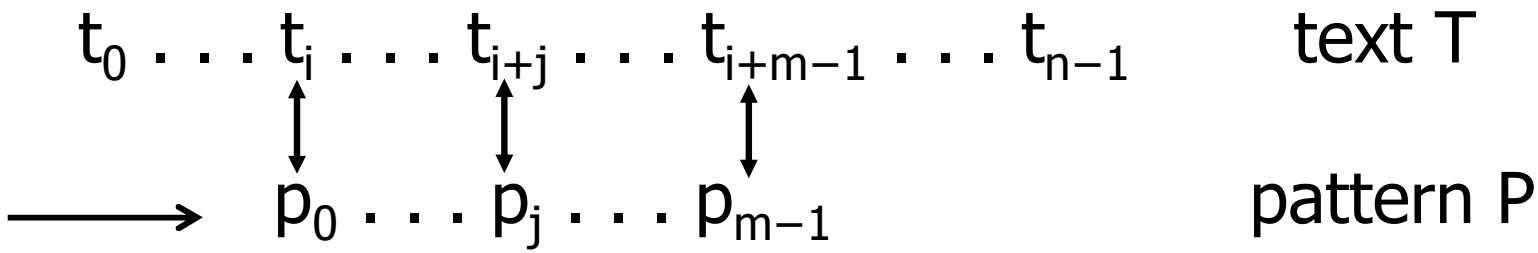
- Tính độ phức tạp thời gian
  - Kích thước đầu vào là n
  - Thao tác cơ bản là so sánh (giả sử thời gian là c)
  - $T(n) = (\Sigma_{i=0, n-2} \Sigma_{j=i+1, n-1} 1)c = (\Sigma_{i=0, n-2} (n-1-i))c = (n-1)nc/2$
  - $T(n)=O(n^2)$ hay $T(n) \in O(n^2)$

BÀI TOÁN SO TRÙNG MẪU CỦA CHUỖI KÝ TỰ

- Cho một chuỗi (xâu) n ký tự (gọi là văn bản-text) và một chuỗi m ký tự $m \leq n$ (gọi là mẩu - pattern), tìm một chuỗi con của văn bản trùng với mẩu
- Cụ thể, tìm chỉ số i của ký tự trái nhất của chuỗi con của văn bản mà so trùng với mẩu

$$t_i = p_0, \ldots, t_{i+j} = p_j, \ldots, t_{i+m-1} = p_{m-1}$$

BÀI TOÁN SO TRÙNG MẪU CỦA CHUỖI KÝ TỰ



- Giải thuật có 2 vòng lặp: Xác định vị trí bắt đầu (từ 0 cho đến n-m) để so sánh trong chuỗi t[0..n] và vòng lặp so trùng các cặp của t[0..n-1] và p[0..m-1]

BÀI TOÁN SO TRÙNG MẪU CỦA CHUỖI KÝ TỰ

```
ALGORITHM BruteForceStringMatch(T [0..n − 1], P[0..m − 1])
1    for i ←0 to n − m do
2        j ←0
3        while j <m and P[j ]= T [i + j ] do
4            j ←j + 1
5        if j = m return i
6    return −1
```

BÀI TOÁN SO TRÙNG MẪU CỦA CHUỖI KÝ TỰ

- Trường hợp tốt nhất T(n)=O(m)
- Trường hợp xấu nhất thực hiện n-m+1 chuyến so sánh, mỗi chuyến thực hiện m lần so các ký tự, T(n)=(n-m+1)mc=O(mn)
- Trung bình T(n)=$(\Sigma_{i=0,\ n-m}\ 1)mcp/(n-m+1)+(1-p)(n-m+1)mc$, trong đó c là thời gian so sánh, p là xác suất tìm thấy mẩu bắt đầu từ chỉ số i của chuỗi văn bản

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHẤT

- Trong mặt phẳng cho n điểm, hãy tìm cặp điểm có khoảng cách nhỏ nhất
- Cụ thể, cho hệ tọa độ xoy và n điểm $P_1(x_1, y_1)$, $P_2(x_2, y_2)$,..., $P_n(x_n, y_n)$, tìm cặp p_i, p_j mà $d(p_i, p_j)$ nhỏ nhất

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHẤT

- Giải thuật vét cạn cho bài toán này là tính khoảng cách của mọi cặp điểm, so sánh các khoảng cách và chọn cặp có khoảng cách bé nhất

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHẤT

ALGORITHM BruteForceClosestPair(P)

///Input: A list P of n (n ≥ 2) points $p_1(x_1, y_1), \ldots, p_n(x_n, y_n)$

//Output: The distance between the closest pair of points

1 d←∞

2 **for** i ←1 **to** n – 1 **do**

3 **for** j ←i + 1 **to** n **do**

4 d ←min(d, sqrt($(x_i - x_j)^2 + (y_i - y_j)^2$)) //sqrt is square root

5 **return** d

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHẤT

- Thao tác cơ bản là tính căn bậc 2
- Gọi thời gian của phép tính căn là c (hằng số)
  - $T(n) = (\Sigma_{i=1, n-1} \Sigma_{j=i+1, n} 2)c = 2(\Sigma_{i=1, n-1} (n-i))c = (n-1)nc$
  - $T(n)=O(n^2)$ hay $T(n) \in O(n^2)$
- Độ phức không lớn nhưng có giảm bằng cách dùng chiến lược chia để trị

BÀI TOÁN NGƯỜI ĐI DU LỊCH

- Cho n thành phố, mà từ mỗi thành phố có thể đi đến bất kỳ thành phố khác với một khoảng cách cho trước, hãy tìm đường đi ngắn nhất từ một thành phố và đi qua tất cả các thành phố khác đúng một lần rồi trở về thành phố xuất phát

BÀI TOÁN NGƯỜI ĐI DU LỊCH

- Thuật toán giải trực tiếp (vét cạn) xét tất các đường đi (mỗi đường đi là một dãy qua n thành phố khác nhau), tính tổng khoảng cách, so sánh để tìm đường đi ngắn nhất

BÀI TOÁN NGƯỜI ĐI DU LỊCH

- Gọi t_1, …, t_n là n thành phố, gọi p[1], p[2], …, p[n] là một hoán vị của n số 1, 2, …, n thì bài toán yêu cầu tìm một dãy các thành phố $t_{p[1]}$, $t_{p[2]}$, …, $t_{p[n]}$, $t_{p[1]}$ sao cho $d(t_{p[1]}, t_{p[2]}) + d(t_{p[2]}, t_{p[3]}) + \ldots + d(t_{p[n]}, t_{p[1]})$ là nhỏ nhất

BÀI TOÁN NGƯỜI ĐI DU LỊCH

- Mỗi một đường đi tương ứng với một hoán vị của n số 1, 2, …, n; có n! hoán vị của n số nên có n! đường đi
- Có thể cố định thành phố xuất phát trong mọi đường đi nên có (n-1)! đường đi

$$t_1, t_{p[2]}, \ldots, t_{p[n]}, t_1$$

BÀI TOÁN NGƯỜI ĐI DU LỊCH

ALGORITHM Traveling Salesman(t[1…n])

1 d←∞, π←∅

2 **for** i ←1 **to (**n-1)! **do**

3 compute (p[2], …, p[n]) // một hoán vị của 2, 3, …, n

4 min← $d(t_1, t_{p[2]})+ d(t_{p[2]}, t_{p[3]})+…+d(t_{p[n]}, t_1)$

5 if min<d

6 min ←d

7 $\pi \leftarrow t_1, t_{p[2]}, …, t_{p[n]}, t_1$

8 **return** π

BÀI TOÁN NGƯỜI ĐI DU LỊCH

- Thao tác cơ bản là xác định một hoán vị
- Gọi thời gian tính một hoán vị là c

$$T(n) = (n-1)!c = O((n-1)!)$$

- Lưu ý: Độ phức tạp quá lớn, tìm giải thuật xấp xỉ (chiến lược thiết kế "háu ăn" ("tham lam")

BÀI TẬP VỀ NHÀ

- Làm bài tập về nhà đã cho trong DS bài tập
- Bài tập thực hành: Hiện thực các giải thuật đã học (trong lý thuyết), tạo các bộ dữ liệu test (ngẫu nhiên), vẽ đồ thị biểu diễn thời gian chạy thông qua bộ đếm thời gian của máy