Chương 2
Phân tích độ phức tạp của một số giải thuật Vét Cạn

Nội dung

Chiến lược thiết kế giải thuật

Một số chiến lược cơ bản

Chiến lược thiết kế giải thuật

Một chiến lược thiết kế giải thuật là một cách tiếp cận tổng quát để giải quyết vấn đề bằng giải thuật có thể áp dụng cho các bài toán trong nhiều lĩnh vực khác nhau

Chiến lược thiết kế giải thuật

Lý do học những chiến lược này

- Hiểu được ý tưởng tổng quát giải quyết những bài toán kinh điển trong khoa học máy tính
- Vận dụng các chiến lược đã học để giải quyết những bài toán mới

Một số chiến lược cơ bản

Vét cạn

- Chiến lược cần nghĩ đến đầu tiên khi giải quyết các bài toán mới
- Với chiến lược này, chúng ta sẽ xem xét tất cả các ứng viên thuộc không gian lời giải của bài toán xem đó có phải là nghiệm của bài toán hay không

Một số chiến lược cơ bản

Vét cạn

- Cần có một hàm kiểm tra xem một ứng viên nào đó có phải là nghiệm của bài toán hay không
- Mặc dù dễ hiểu nhưng phương pháp này không hiệu quả đối với các bài toán mà kích thước dữ liệu nhập lớn

Một số chiến lược cơ bản

Vét cạn

- Ví dụ: Tìm mẫu có kích thước xác định xuất hiện nhiều nhất trong chuỗi thời gian
- Ý tưởng: Duyệt từ đầu đến cuối chuỗi để xét từng mẫu có kích thước như đã xác định, tiến hành tìm từ đầu đến cuối các mẫu khác trùng với nó và tổng kết số lượng

Một số chiến lược cơ bản

Phương pháp sinh

- Kiến thức đại số tổ hợp
- Chỉnh hợp lặp
- Chỉnh hợp không lặp
- Hoán vị
- Tổ hợp

Một số chiến lược cơ bản

Phương pháp sinh

- Có thể áp dụng để giải bài toán liệt kê tổ hợp đặt ra nếu hai điều kiện sau thoả mãn
- Có thể xác định được một thứ tự trên tập các cấu hình tổ hợp cần liệt kê. Từ đó biết được cấu hình đầu tiên và cấu hình cuối cùng trong thứ tự đó.
- Xây dựng được thuật toán từ một cấu hình chưa phải cấu hình cuối, sinh ra được cấu hình kế tiếp

Một số chiến lược cơ bản

Phương pháp sinh

- Mô tả

```
⟨Xây dựng cấu hình đầu tiên⟩;
repeat
    ⟨Đưa ra cấu hình đang có⟩;
    ⟨Từ cấu hình đang có sinh ra cấu hình
          kế tiếp nếu còn⟩;
until ⟨hết cấu hình⟩;
```

Một số chiến lược cơ bản

Phương pháp sinh

- Thứ tự từ điển
  - Xét a[0..n-1] và b[0..n-1] là hai dãy độ dài n. Khi đó a < b nếu như tồn tại một số nguyên dương k: $0 \le k < n-1$ để a[i] = b[i] với $\forall i: 0 \le i \le k$ và a[k+1] < b[k+1]
  - Khi độ dài a và b khác nhau, thêm vào cuối dãy a hoặc dãy b những phần tử đặc biệt gọi là phần tử Ø để độ dài của a và b bằng nhau (phần tử Ø nhỏ hơn tất cả các phần tử khác)

Một số chiến lược cơ bản

Phương pháp sinh

- Bài toán: Sinh một dãy nhị phân độ dài n x[0..n-1] trong đó x[i] ∈ {0, 1} (∀i : 0 ≤ i ≤ n-1).
- Phân tích
  - Khi n = 3, các dãy nhị phân độ dài 3 được liệt kê như sau

p(x)	0	1	2	3	4	5	6	7
x	000	001	010	011	100	101	110	111

Một số chiến lược cơ bản

Phương pháp sinh

- Phân tích
  - Dãy đầu tiên là 00…0 và dãy cuối cùng là 11…1
  - Nếu dãy x = [0..n-1] là dãy đang có và không phải dãy cuối cùng cần liệt kê thì dãy kế tiếp sẽ nhận được bằng cách cộng thêm 1 (theo cơ số 2 có nhớ) vào dãy hiện tại.

Một số chiến lược cơ bản

Phương pháp sinh

- Phân tích

Dãy đang có:	10010000	Dãy đang có:	10010111
cộng thêm 1:	+ 1	cộng thêm 1:	+ 1
Dãy mới:	10010001	Dãy mới:	10011000

Một số chiến lược cơ bản

Phương pháp sinh

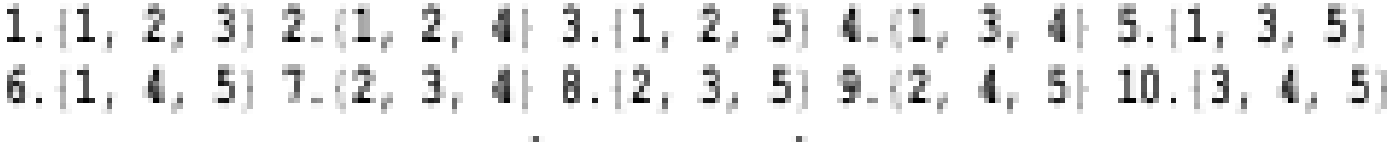
- Giải thuật

```
for i = 0 to n-1  {In ra cấu hình hiện tại}
    x[i] = 0
do
    for i = 0 to n-1
        write(x[i])
    i = n - 1 {x[i] là phần tử cuối dãy, lùi dần i cho tới khi gặp số 0 hoặc khi i < 0 thì dừng}
    while i >= 0 and x[i] == 1
        i = i - 1
    if i >= 0 then {Chưa gặp phải cấu hình 11…1}
        x[i] = 1
        for j = i+1 to n-1
            x[j] = 0
while i >= 0 {Còn cấu hình}
```

Một số chiến lược cơ bản

Phương pháp sinh

- Bài toán: Liệt kê các tập con k phần tử của tập {1, 2, …, n} theo thứ tự từ điển
- Phân tích
  - Với n = 5, k = 3, ta phải liệt kê đủ 10 tập con



  - Như vậy tập con đầu tiên là {1, 2, …, k}. Tập con cuối cùng là {n - k + 1, n - k + 2, …, n}.

Một số chiến lược cơ bản

Phương pháp sinh

- Phân tích
  - Biểu diễn mỗi tập con là một dãy trong đó phần tử đứng trước nhỏ hơn phần tử đứng sau
  - Giới hạn trên của x[k-1] là n, của x[k-2] là n - 1, của x[k-3] là n - 2…
  - Tổng quát: giới hạn trên của x[i] = n - k + i + 1, giới hạn dưới của x[i] = x[i-1]+1

Một số chiến lược cơ bản

Phương pháp sinh

- Phân tích
  - Ví dụ: n = 9, k = 6. Cấu hình đang có x = ⟨1, 2, 6, 7, 8, 9⟩ thì 2 cấu hình kế tiếp là x = ⟨1, 3, 4, 5, 6, 7⟩ và x = ⟨1, 3, 4, 5, 6, 8⟩
  - Kỹ thuật sinh cấu hình kế tiếp: Tìm từ cuối dãy lên đầu cho tới khi gặp một phần tử x[i] chưa đạt giới hạn trên n - k + i + 1
    - Nếu tìm thấy thì tăng x[i] đó lên 1, đặt tất cả các phần tử phía sau x[i] bằng giới hạn dưới
    - Nếu không tìm thấy tức là mọi phần tử đã đạt giới hạn trên, đây là cấu hình cuối cùng

Một số chiến lược cơ bản

Phương pháp sinh

- Giải thuật

```
for i = 0 to k-1
    x[i] = i + 1 {Khởi tạo x := (1, 2, …, k)}
do
    {In ra cấu hình hiện tại}
    for i = 0 to k-1
        write(x[i])
    {Sinh tiếp}
    i = k - 1 {Xét từ cuối dãy lên tìm x[i] chưa đạt giới hạn trên n - k + i + 1}
    while i >= 0 and x[i] = n - k + i + 1
        i = i - 1
    if i >= 0 then {Nếu chưa lùi đến dưới 0 có nghĩa là chưa phải cấu hình kết thúc}
        x[i] = x[i] + 1 {Tăng x[i] lên 1}
        for j = i+1 to k-1 {Đặt các phần tử đứng sau x[i] bằng giới hạn dưới của nó}
            x[j] := x[j - 1] + 1;
while i>= 0 {Lùi đến dưới 0 có nghĩa là tất cả các phần tử đã đạt giới hạn trên - hết cấu hình}
```

Một số chiến lược cơ bản

Phương pháp sinh

- Bài toán: Liệt kê các hoán vị của {1, 2, …, n} theo thứ tự từ điển
- Phân tích
  - Ví dụ với n = 4, ta phải liệt kê đủ 24 hoán vị

```
 1.1234  2.1243  3.1324  4.1342  5.1423  6.1432
 7.2134  8.2143  9.2314 10.2341 11.2413 12.2431
13.3124 14.3142 15.3214 16.3241 17.3412 18.3421
19.4123 20.4132 21.4213 22.4231 23.4312 24.4321
```

  - Hoán vị đầu tiên là ⟨1, 2, …, n⟩.
  - Hoán vị cuối cùng là ⟨n, n-1, …, 1⟩.

Một số chiến lược cơ bản

Phương pháp sinh

- Phân tích
  - Hoán vị sẽ sinh ra phải vừa đủ lớn hơn hoán vị hiện tại
  - Giả sử hoán vị hiện tại là x = ⟨3, 2, 6, 5, 4, 1⟩, hoán vị kế tiếp sẽ là ⟨3, 4, 1, 2, 5, 6⟩.
  - Đoạn cuối của hoán vị hiện tại được xếp giảm dần, số 4 là số nhỏ nhất trong đoạn cuối giảm dần thoả mãn điều kiện lớn hơn 2. Nếu đổi chỗ 4 và 2 thì ta sẽ được đoạn cuối vẫn được sắp xếp giảm dần. Khi đó muốn biểu diễn nhỏ nhất cho các giá trị trong đoạn cuối thì ta chỉ cần đảo ngược đoạn cuối.

Một số chiến lược cơ bản

Phương pháp sinh

- Phân tích
  - Kỹ thuật sinh cấu hình kế tiếp: Xác định đoạn cuối giảm dần dài nhất, tìm chỉ số i của phần tử x[i] đứng liền trước đoạn cuối đó, tức là tìm từ vị trí sát cuối dãy lên đầu, gặp chỉ số i đầu tiên thỏa mãn x[i] < x[i+1].
  - Nếu tìm thấy chỉ số i như trên thì tìm phần tử x[k] nhỏ nhất thoả mãn điều kiện x[k] > x[i] bằng cách tìm từ cuối dãy lên đầu gặp chỉ số k đầu tiên thoả mãn x[k] > x[i]. Đảo giá trị x[k] và x[i]. Lật ngược thứ tự đoạn cuối giảm dần (từ x[i+1] đến x[k]) trở thành tăng dần.
  - Nếu không tìm thấy tức là toàn dãy đã sắp giảm dần, đây là cấu hình cuối cùng

Một số chiến lược cơ bản

Phương pháp sinh

- Giải thuật

```
for i = 0 to n-1
    x[i] = i + 1 {Khởi tạo cấu hình đầu: x[0] := 1; x[1] := 2; …, x[n-1] := n}
do
    for i = 0 to n-1
        write(x[i]) {In ra cấu hình hoán vị hiện tại}
    i = n - 2
    while i >= 0 and x[i] > x[i + 1]
        i = i - 1
    if i >= 0 then {Chưa gặp phải hoán vị cuối (n, n-1, …, 1)}
        k = n - 1 {x[k] là phần tử cuối dãy}
        while x[k] < x[i]
            k = k - 1 {Lùi dần k để tìm gặp x[k] đầu tiên lớn hơn x[i]}
        Swap(x[k], x[i]) {Đổi chỗ x[k] và x[i]}
        a = i + 1
        b = n - 1 {Lật ngược đoạn cuối giảm dần, a: đầu đoạn, b: cuối đoạn}
        while a < b
            Swap(x[a], x[b]) {Đảo giá trị x[a] và x[b]}
            a = a + 1 {Tiến a và lùi b, tiếp tục cho tới khi a, b chạm nhau}
            b = b - 1
while i >= 0 {Toàn dãy là dãy giảm dần - không sinh tiếp được - hết cấu hình}
```

Thank you