



Algorithm Paradigms

Các dạng thuật toán

Nội dung

- Vét cạn – Complete Search/Brute-force Search
- Chia để trị – Divide and Conquer
- Tham lam – Greedy
- Qui hoạch động – Dynamic Programming

Bài toán 1

Tìm vị trí số x trên dãy a gồm N số thực.

- Input: dãy (a, N) – dãy gồm N số thực, số x – số cần tìm
- Output: số nguyên – vị trí của x trên a (-1 nếu a không có x)

Thuật toán vét cạn

- Ý tưởng: Thử tìm x tại từng vị trí của a , nếu tìm thấy thì ngừng và báo vị trí. Nếu đã thử hết các vị trí mà vẫn không thấy x thì báo -1
- Thuật toán:
 1. For $pos = 0 \div N-1$
 1. If $(a[pos] = x)$
 1. Return pos
 - EndFor
 2. Return -1

Vét cạn – dạng thức chung

Tìm lời giải cho bài toán P

1. $s \leftarrow \text{first}(P)$
2. While ($c \neq \phi$)
 1. If $\text{correct}(P, c)$ Return c ;
 2. $c \leftarrow \text{next}(P, c)$;
3. Return NULL; //không có lời giải

Dãy tăng (giảm) dần: chia để trị - tìm nhị phân

- Ý tưởng: Thử tìm x tại vị trí giữa (mid) của $(a, left, right)$, nếu $x = a[mid]$ thì ngừng và báo vị trí; nếu $x < a[mid]$ thì tìm x ở đoạn bên trái mid ; ngược lại tìm x ở đoạn bên phải mid
- Thuật toán:
 1. $left = 0; right = N-1$
 2. While ($left \leq right$)
 1. $mid = left + (right - left)/2$
 2. If ($x = a[mid]$) Return mid ;
 3. If ($x < a[mid]$) $right = mid - 1$
 4. Else $left = mid + 1$
 - EndWhile
 3. Return -1

Cài đặt bằng C/C++

```
1  int BiSearch(double a[], int n, double x)
2  {
3      int left = 0, right = n-1;
4      while (left <= right)
5      {
6          mid = left + (right-left)/2;
7          if (x == a[mid])      return mid;
8          if (x < a[mid])      right = mid - 1;
9          else                left = mid + 1;
10     }
11     return -1;
12 }
```

Bài toán 2

Tính lũy thừa bậc N (nguyên không âm) của số thực x

- Input: x – số thực, N – số nguyên không âm
- Output: số thực – x^N

Cách tiếp cận

- “ngây thơ – naïve”: nhân tích lũy N giá trị x sẽ thu được x^N , cần thực hiện N phép nhân
- Chia để trị

Lũy thừa nhanh – chia để trị

- $x^{13} = x * x * \dots * x$: cần 12 phép nhân
- $x^{13} = x * x^4 * x^8$: chỉ cần 5 phép nhân
- Ý tưởng: chia để trị, giảm kích thước bài toán
 - Nếu N chẵn: $x^N = x^{\frac{N}{2}} * x^{\frac{N}{2}} = \left(x^{\frac{N}{2}}\right)^2$
 - Nếu N lẻ $x^N = x * x^{N-1}$

Cài đặt đệ qui

```
1. double FastPower(double x, unsigned short N)
2. {
3.     if (!N) //N == 0
4.         return 1;
5.     if (N & 1) //N % 2 == 1
6.         return x * FastPower(x, N-1);
7.     double y = FastPower(x, N/2);
8.     return y*y;
9. }
```

Cài đặt không đệ qui

```
1. double FastPower(double x, unsigned short N)
2. {
3.     double ans = 1;
4.     while (N) {
5.         if (N&1) ans *= x;
6.         x = x*x;
7.         N >>= 1; //N /= 2
8.     }
9.     return ans;
10. }
```

Bài toán 3

Tìm tập con có tích lớn nhất của dãy a có N phần tử

- Input: N – số nguyên không âm, a – dãy gồm N số thực,
- Output: số thực – tích lớn nhất

Cách tiếp cận:

- “ngây thơ – naïve”: phát sinh tất cả 2^N tập con, từ đó chỉ ra tập con có tích lớn nhất – vét cạn
- tham lam

Tập con có tích lớn nhất: thuật toán tham lam

► Ý tưởng:

1. Nếu dãy không có số 0 và có số số âm là chẵn: kết quả là tích toàn bộ các số của dãy
2. Nếu dãy chỉ có <1 số âm và các số khác đều bằng 0: kết quả là 0
3. Trường hợp còn lại: số số âm là lẻ và có số 0: tích các số khác không ngoại trừ số âm có giá trị lớn nhất.

► Thuật toán:

1. Xác định số lượng số 0 (`count_0`) và số lượng số âm (`count_neg`), số âm lớn nhất (`max_neg`), tích các số khác không (`product`)
2. If (`count_0 = N`) or ((`count_neg = 1`) and (`count_0 = N-1`)) Return 0;
3. If (`count_0 % 2 = 1`) `product /= max_neg;`
4. Return `product;`

```
1. int MaxProduct (int a[], int N)
2. {
3.     int count_0=0, count_neg=0, max_neg=INT_MIN, product=1;
4.     for (int i=0; i<N; i++) {
5.         if (!a[i])        count_0 ++;
6.         else {
7.             product *= a[i];
8.             if (a[i] < 0)
9.                 count_neg ++, max_neg = max(max_neg, a[i]);
10.        }
11.    }
12.    if ((!count_0) || ((count_neg==1) && (count_0==N-1)))
13.        return 0;
14.    if (count_neg & 1)        product /= max_neg;
15.    return product;
16.}
```

Bài toán 4

Tìm số Fibonacci thứ N , nhắc lại:

$$\begin{cases} F_0 = F_1 = 1 \\ F_N = F_{N-1} + F_{N-2} \end{cases} \quad \text{với } N \geq 2$$

- Input: N – số nguyên không âm
- Output: F_N – số Fibonacci thứ N

Cách tiếp cận:

- “ngây thơ – naïve”: theo đúng công thức đệ qui
- Qui hoạch động:

Tính số Fibonacci: thuật toán qui hoạch động

- Ý tưởng: Tính các giá trị từ $F_0, F_1 \dots$ dần về đến F_N
- Thuật toán:
 1. $F_N = F_{N1} = F_{N2} = 1$
 2. For $i = 2 \div N$
 1. $F_N = F_{N1} + F_{N2}$
 2. $F_{N1} = F_N$
 3. $F_{N2} = F_{N1}$EndFor
 3. Return F_N ;

Cài đặt bằng C/C++

```
1. unsigned long Fibo(unsigned short N)
2. {
3.     unsigned long FN, FN1, FN2;
4.     FN = FN1 = FN2 = 1;
5.     for (unsigned short i=2; i<=N; i++)
6.     {
7.         FN = FN1 + FN2;
8.         FN1 = FN;
9.         FN2 = FN1;
10.    }
11.    return FN;
12. }
```

Bài tập

1. Thiết kế thuật toán dạng vét cạn để tìm tập con có tích lớn nhất của dãy a , cài đặt chương trình bằng C/C++/Python
2. Tìm hiểu thuật toán MergeSort, hãy cho biết đây là dạng nào trong các loại: vét cạn/chia để trị/tham lam/qui hoạch động
3. Bài toán đổi tiền: Có M loại tiền mệnh giá S_1, S_2, \dots, S_M ; số lượng mỗi loại không hạn chế. Cần xác định số cách đổi số tiền N đồng thành các tờ tiền trong M loại đã cho.

Ví dụ: $N=4, M=3$ và $S = \{1, 2, 3\}$. Có 4 cách đổi tiền: 4 tờ 1; 2 tờ 1 - 1 tờ 2; hai tờ 2; 1 tờ 1 - 1 tờ 3.

Hãy lựa chọn dạng thuật toán thích hợp để giải quyết bài toán. Giải thích lý do chọn.