Nội dung

1.1. Ví dụ mở đầu

1.2. Thuật toán và độ phức tạp

1.3. Kí hiệu tiệm cận

1.3. Một số kĩ thuật phân tích thuật toán

Ví dụ: Tìm dãy con lớn nhất (The maximum subarray problem)

- Cho dãy số gồm n số:

  $a_1, a_2, \ldots, a_n$

  Dãy gồm liên tiếp các số $a_i, a_{i+1}, \ldots, a_j$ với $1 \le i \le j \le n$ được gọi là **dãy con** của dãy đã cho và $\sum_{k=i}^{j} a_k$ *được gọi là trọng lượng của dãy con này*

  Bài toán đặt ra là: Hãy tìm trọng lượng lớn nhất của các dãy con, tức là tìm cực đại giá trị $\sum_{k=i}^{j} a_k$. Ta gọi dãy con có trọng lượng lớn nhất là **dãy con lớn nhất**.

Ví dụ: Cho dãy số -2, **11, -4, 13**, -5, 2 thì cần đưa ra câu trả lời là 20 (dãy con lớn nhất là 11, -4, 13 với giá trị = 11+ (-4)+13 =20)

➔ Để giải bài toán này, ta có thể dùng nhiều cách khác nhau, ví du duyệt toàn bộ (brute force), chia để chị (divide and conquer), quy hoạch động (dynamic programming), ...

6

Ví dụ: Tìm dãy con lớn nhất (The maximum subarray problem)

1.1.1. Duyệt toàn bộ (Brute force)

1.1.2. Duyệt toàn bộ có cải tiến

1.1.3. Thuật toán quy hoạch động (Dynamic programming)

Ví dụ: Tìm dãy con lớn nhất (The maximum subarray problem)

1.1.1. Duyệt toàn bộ (Brute force)

1.1.2. Duyệt toàn bộ có cải tiến

1.1.3. Thuật toán quy hoạch động (Dynamic programming)

1.1.1. Thuật toán duyệt toàn bộ giải bài toán dãy con lớn nhất

- Thuật toán đơn giản đầu tiên có thể nghĩ để giải bài toán đặt ra là: Duyệt tất cả các dãy con có thể :

$$a_i, a_{i+1}, \ldots, a_j \quad \text{với} \quad 1 \le i \le j \le n,$$

và tính tổng của mỗi dãy con để tìm ra trọng lượng lớn nhất.

- Trước hết nhận thấy rằng, tổng số các dãy con có thể của dãy đã cho là:

$$C(n, 1) + C(n, 2) = n^2/2 + n/2$$

Thuật toán duyệt toàn bộ: duyệt tất cả các dãy con

Chỉ số i	0	1	2	3	4	5
a[i]	-2	11	-4	13	-5	2

```
i = 0: (-2), (-2, 11), (-2,11, -4), (-2,11,-4,13), (-2,11,-
   4,13,-5), (-2,11,-4,13,-5,2)
i = 1: (11), (11, -4), (11, -4, 13), (11, -4, 13, -5), (11, -4,
   13, -5, 2)
i = 2: (-4), (-4, 13), (-4, 13, -5), (-4,13,-5,2)
i = 3: (13), (13,-5), (13, -5,2)
i = 4: (-5), (-5, 2)
i = 5: (2)
```

Với mỗi chỉ số i : duyệt tất cả các dãy con bắt đầu từ phần tử a[i] có 1 phần tử, có 2 phần tử, … :

- i = 0: tất cả các dãy con bắt đầu từ a[0] có 1 phần tử, 2 phần tử, … 6 phần tử
- i = 1: tất cả các dãy con bắt đầu từ a[1] có 1 phần tử, 2 phần tử, … 5 phần tử
- …
- i = 5: tất cả các dãy con bắt đầu từ a[5] có 1 phần tử

```
int maxSum = -∞;
for (int i=0; i<n; i++) {
   for (int j=i; j<n; j++) {
      int sum = 0;
      for (int k=i; k<=j; k++)
         sum += a[k];
      if (sum > maxSum)
         maxSum = sum;
   }
}
```

10

Thuật toán duyệt toàn bộ: duyệt tất cả các dãy con

- **Phân tích thuật toán:** Ta sẽ tính số lượng phép cộng mà thuật toán phải thực hiện, tức là đếm xem dòng lệnh

```
sum += a[k]
```

phải thực hiện bao nhiêu lần. Số lượng phép cộng là:

$$\sum_{i=0}^{n-1}\sum_{j=i}^{n-1}(j-i+1)=\sum_{i=0}^{n-1}(1+2+...+(n-i))=\sum_{i=0}^{n-1}\frac{(n-i)(n-i+1)}{2}$$

$$=\frac{1}{2}\sum_{k=1}^{n}k(k+1)=\frac{1}{2}\left[\sum_{k=1}^{n}k^2+\sum_{k=1}^{n}k\right]=\frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6}+\frac{n(n+1)}{2}\right]$$

$$=\frac{n^3}{6}+\frac{n^2}{2}+\frac{n}{3}$$

```
int maxSum = -∞;
for (int i=0; i<n; i++) {
   for (int j=i; j<n; j++) {
      int sum = 0;
      for (int k=i; k<=j; k++)
          sum += a[k];
      if (sum > maxSum)
          maxSum = sum;
   }
}
```

11

Ví dụ: Tìm dãy con lớn nhất (The maximum subarray problem)

1.1.1. Duyệt toàn bộ (Brute force)

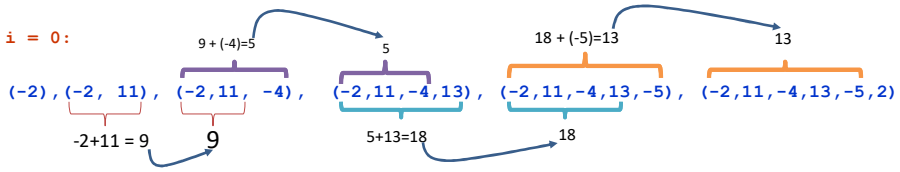
1.1.2. Duyệt toàn bộ có cải tiến

1.1.3. Thuật toán quy hoạch động (Dynamic programming)

1.1.2. Duyệt toàn bộ có cải tiến

Thuật toán duyệt toàn bộ: duyệt tất cả các dãy con

Index i	0	1	2	3	4	5
a[i]	-2	11	-4	13	-5	2



- **Cài đặt cải tiến:**

Nhận thấy, ta có thể tính tổng các phần tử từ vị trí *i* đến *j* từ tổng của các phần tử từ *i* đến *j*-1 chỉ bằng 1 phép cộng:

$$\sum_{k=i}^{j} a[k] = a[j] + \sum_{k=i}^{j-1} a[k]$$

Tổng các phần tử từ *i* đến *j* — Tổng các phần tử từ *i* đến *j*-1

13

1.1.2. Duyệt toàn bộ có cải tiến

Thuật toán duyệt toàn bộ: duyệt tất cả các dãy con

- **Cài đặt cải tiến:**

Nhận thấy, ta có thể tính tổng các phần tử từ vị trí *i* đến *j* từ tổng của các phần tử từ *i* đến *j*-1 chỉ bằng 1 phép cộng:

$$\sum_{k=i}^{j} a[k] = a[j] + \sum_{k=i}^{j-1} a[k]$$

Tổng các phần tử từ *i* đến *j* — Tổng các phần tử từ *i* đến *j*-1

```
int maxSum = -∞;
for (int i=0; i<n; i++) {
   for (int j=i; j<n; j++) {
      int sum = 0;
      for (int k=i; k<=j; k++)
          sum += a[k];
      if (sum > maxSum)
          maxSum = sum;
   }
}
```

```
int maxSum = a[0];
for (int i=0; i<n; i++) {
   int sum = 0;
   for (int j=i; j<n; j++) {
      sum += a[j];
      if (sum > maxSum)
          maxSum = sum;
   }
}
```

14

1.1.2. Duyệt toàn bộ có cải tiến

Thuật toán duyệt toàn bộ: duyệt tất cả các dãy con

- **Phân tích thuật toán.** Ta lại tính số lần thực hiện phép cộng:

```
Sum += a[j]
```

Và thu được kết quả:

$$\sum_{i=0}^{n-1}(n-i) = n + (n-1) + \ldots + 1 = \frac{n^2}{2} + \frac{n}{2}$$

- Để ý rằng số này là đúng bằng số lượng dãy con. Dường như thuật toán thu được là rất tốt, vì ta phải xét mỗi dãy con đúng 1 lần.

```
int maxSum = a[0];
for (int i=0; i<n; i++) {
   int sum = 0;
   for (int j=i; j<n; j++) {
      sum += a[j];
      if (sum > maxSum)
          maxSum = sum;
   }
}
```

15

Bài toán dãy con lớn nhất: so sánh thời gian tính của các thuật toán

Số lương phép cộng mà mỗi thuật toán cần thực hiện là:

1.1.1. Duyệt toàn bộ $\frac{n^3}{6} + \frac{n^2}{2} + \frac{n}{3}$

1.1.2. Duyệt toàn bộ có cải tiến $\frac{n^2}{2} + \frac{n}{2}$

➔ Cùng một bài toán, ta đã đề xuất 2 thuật toán đòi hỏi số lượng phép toán khác nhau, và vì thế sẽ đòi hỏi thời gian tính khác nhau.

Bảng dưới đây cho thấy thời gian tính của 2 thuật toán trên, với giả thiết: máy tính có thể thực hiện10^8 phép cộng trong một giây

Độ phức tạp	n=10	Thời gian	n=100	Thời gian	$n=10^4$	Thời gian	$n=10^6$	Thời gian
n^3	10^3	10^{-5} giây	10^6	10^{-2} giây	10^{12}	2.7 hours	10^{18}	115 ngày
n^2	100	10^{-6}giây	10000	10^{-4} giây	10^8	1 giây	10^{12}	2.7 giờ

Bài toán dãy con lớn nhất: so sánh thời gian tính của các thuật toán

Độ phức tạp	n=10	Thời gian	n=100	Thời gian	$n=10^4$	Thời gian	$n=10^6$	Thời gian
n^3	10^3	10^{-5} giây	10^6	10^{-2} giây	10^{12}	2.7 hours	10^{18}	115 ngày
n^2	100	10^{-6}giây	10000	10^{-4} giây	10^8	1 giây	10^{12}	2.7 giờ

- Với n nhỏ thời gian tính là không đáng kể.
- Vấn đề trở nên nghiêm trọng hơn khi $n > 10^6$.
- Còn có thể làm tốt hơn nữa không?
  - Yes! Có thể đề xuất thuật toán chỉ đòi hỏi n phép cộng!

Ví dụ: Tìm dãy con lớn nhất (The maximum subarray problem)

1.1.1. Duyệt toàn bộ (Brute force)

$$\frac{n^3}{6}+\frac{n^2}{2}+\frac{n}{3}$$

1.1.2. Duyệt toàn bộ có cải tiến

$$\frac{n^2}{2}+\frac{n}{2}$$

1.1.3. Thuật toán Quy hoạch động (Dynamic programming)

n

1.1.3. Thuật toán quy hoạch động giải bài toán dãy con lớn nhất

Thuật toán quy hoạch động được chia làm 3 giai đoạn:

1. Phân rã (Divide): chia bài toàn cần giải thành những bài toán con

(Bài toán con (Subproblem): là bài toán có cùng dạng với bài toán đã cho, nhưng kích thước nhỏ hơn)

2. Ghi nhận lời giải: lưu trữ lời giải của các bài toán con vào 1 bảng

3. Tổng hợp lời giải: Lần lượt từ lời giải của các bài toán con kích thước nhỏ hơn tìm cách xây dựng lời giải của bài toán kích thước lớn hơn, cho đến khi thu được lời giải của bài toán xuất phát (là bài toán con có kích thước lớn nhất).

1.1.3. Thuật toán quy hoạch động giải bài toán dãy con lớn nhất

Thuật toán quy hoạch động được chia làm 3 giai đoạn:

1. Phân rã:

- Gọi s_i là trọng lượng của dãy con lớn nhất của dãy $a_1, a_2, ..., a_i$, $i = 1, 2, ..., n$.
- Rõ ràng, s_n là giá trị cần tìm (lời giải của bài toán).

3. Tổng hợp lời giải:

- $s_1 = a_1$
- Giả sử $i > 1$ và ta đã biết giá trị s_k với $k = 1, 2, ..., i-1$. Ta cần tính giá trị s_i là trọng lượng của dãy con lớn nhất của dãy:

  $a_1, a_2, ..., a_{i-1}, a_i$.
- Nhận thấy rằng: dãy con lớn nhất của dãy $a_1, a_2, ..., a_{i-1}, a_i$ có thể hoặc bao gồm phần tử a_i hoặc không bao gồm phần tử a_i ➔ do đó, dãy con lớn nhất của dãy $a_1, a_2, ..., a_{i-1}, a_i$ chỉ có thể là một trọng 2 dãy sau:
  - Dãy con lớn nhất của dãy $a_1, a_2, ..., a_{i-1}$
  - Dãy con lớn nhất của dãy $a_1, a_2, ..., a_i$, *và dãy con này kết thúc tại phần tử* a_i.

➔ Do đó, ta có $s_i = max \{s_{i-1}, e_i\}$, $i = 2, ..., n$.

với e_i là trọng lượng của dãy con lớn nhất $a_1, a_2, ..., a_i$ và dãy con này kết thúc tại a_i.

Để tính e_i, *ta xây dựng công thức đệ quy*:

- $e_1 = a_1$;
- $e_i = \max \{a_i, e_{i-1} + a_i\}$, $i = 2, ..., n$.

20

1.1.3. Thuật toán quy hoạch động giải bài toán dãy con lớn nhất

Thuật toán quy hoạch động được chia làm 3 giai đoạn:

1. Phân rã:

- Gọi s_i là trọng lượng của dãy con lớn nhất của dãy $a_1, a_2, ..., a_i$, $i = 1, 2, ..., n$.
- Rõ ràng, s_n là giá trị cần tìm (lời giải của bài toán).

3. Tổng hợp lời giải:

- $s_1 = a_1$, $s_2 = \max \{a_1, a_2, a_1+a_2\}$
- Giả sử $i > 1$ và ta đã biết giá trị s_k với $k = 1, 2, ..., i-1$. Ta cần tính giá trị s_i là trọng lượng của dãy con lớn nhất của dãy:

  $a_1, a_2, ..., a_{i-1}, a_i$.
- Nhận thấy rằng: dãy con lớn nhất của dãy $a_1, a_2, ..., a_{i-1}, a_i$ có thể hoặc bao gồm phần tử a_i hoặc không bao gồm phần tử a_i ➔ do đó, dãy con lớn nhất của dãy $a_1, a_2, ..., a_{i-1}, a_i$ chỉ có thể là một trọng 2 dãy sau:
  - Dãy con lớn nhất của dãy $a_1, a_2, ..., a_{i-1}$
  - Dãy con lớn nhất của dãy $a_1, a_2, ..., a_i$, *và dãy con này kết thúc tại phần tử* a_i.

➔ Do đó, ta có $s_i = max \{s_{i-1}, e_i\}$, $i = 2, ..., n$.

với e_i là trọng lượng của dãy con lớn nhất $a_1, a_2, ..., a_i$ và dãy con này kết thúc tại a_i.

Để tính e_i, *ta xây dựng công thức đệ quy*:

- $e_1 = a_1$;
- $e_i = \max \{a_i, e_{i-1} + a_i\}$, $i = 2, ..., n$.

```
MaxSub(a)
{
    smax = a[1];          // smax : trọng lượng của dãy con lớn nhất
    ei   = a[1];          // ei: trọng lượng của dãy con lớn nhất kết thúc tại phần tử a[i]
    for i = 2 to n {
          ei = max {a[i], ei + a[i] };
          smax =  max {smax, ei};
    }
}
```

1.1.4. Thuật toán quy hoạch động giải bài toán dãy con lớn nhất

```
MaxSub(a)
{
    smax = a[1];          // smax : trọng lượng của dãy con lớn nhất
    ei   = a[1];          // ei: trọng lượng của dãy con lớn nhất kết thúc tại phần tử a[i]
    imax = 1;             // imax : chỉ số của phần tử cuối cùng thuộc dãy con lớn nhất
    for i = 2 to n {
          u = ei + a[i];
          v = a[i];
          if (u > v) ei = u
           else ei = v;
          if (ei > smax) {
              smax := ei;
              imax := i;
          }
    }
}
```

```
MaxSub(a)
{
    smax = a[1];          // smax : trọng lượng của dãy con lớn nhất
    ei   = a[1];          // ei: trọng lượng của dãy con lớn nhất kết thúc tại phần tử a[i]
    for i = 2 to n {
          ei = max {a[i], ei + a[i] };
          smax =  max {smax, ei};
    }
}
```

Phân tích thuật toán:

Số phép cộng phải thực hiện trong thuật toán
= Số lần thực hiện câu lệnh u = ei + a[i]
= n

So sánh 3 thuật toán

- Bảng sau cho thấy ước tính thời gian tính của 3 thuật toán đề xuất ở trên (với giả thiết máy tính có thể thực hiện 10^8 phép cộng trong 1 giây)

Thuật toán	Độ phức tạp	n=10^4	Thời gian	n=10^6	Thời gian
Duyệt toàn bộ	n^3	10^{12}	2.7 giờ	10^{18}	115 ngày
Duyệt toàn bộ có cải tiến	n^2	10^8	1 giây	10^{12}	2.7 giờ
Quy hoạch động (Dynamic programming)	n	10^4	10^{-4} giây	10^6	$2*10^{-2}$ giây

- Ví dụ này cho ta thấy việc phát triển được thuật toán hiệu quả có thể giảm bớt được chi phí thời gian một cách đáng kể như thế nào.

Nội dung

1.1. Ví dụ mở đầu

1.2. Thuật toán và độ phức tạp

1.3. Kí hiệu tiệm cận

1.4. Một số kĩ thuật phân tích thuật toán