Thuật toán ứng dụng

Giảng viên : Đỗ Tuấn Anh, Lê Quốc Trung
TA: Trần Thanh Tùng

Viện Công Nghệ Thông Tin và Truyền Thông
Trường Đại học Bách Khoa Hà Nội

Tháng 3, năm 2020

Mục lục

ADD

Cho hai số a và b, hãy viết chương trình bằng C/C++ tính số $c = a + b$

Lưu ý giới hạn : $a, b < 10^{19}$ dẫn đến c có thể vượt quá khai báo `long long`

Thuật toán

Chỉ cần khai báo a, b, c kiểu `unsigned long long`, trường hợp tràn số chỉ xảy ra khi a, b có 19 chữ số và c có 20 chữ số

1. Tách $a = a_1 \times 10 + a_0$
2. Tách $b = b_1 \times 10 + b_0$
3. Tách $a_0 + b_0 = c_1 \times 10 + c_0$
4. In ra liên tiếp $a_1 + b_1 + c_1$ và c_0

Code

```
1  int main() {
2      unsigned long long a,b,c;
3      cin >> a>>b;
4
5      unsigned long long a0 = a % 10;
6      unsigned long long a1 = (a-a0) / 10;
7      unsigned long long b0 = b % 10;
8      unsigned long long b1 = (b-b0) /10;
9      unsigned long long c0 = (a0+b0) % 10;
10     unsigned long long c1 = (a0+b0-c0) / 10;
11     c1 = a1 + b1 + c1;
12     if (c1>0) cout << c1;
13     cout << c0;
14     return 0;
15 }
```

SUBSEQMAX

- Cho dãy số $s = \langle a_1, \ldots, a_n \rangle$
- một dãy con từ i đến j là $s(i,j) = \langle a_i, \ldots, a_j \rangle$, $1 \leq i \leq j \leq n$
- với trọng số $w(s(i,j)) = \sum_{k=i}^{j} a_k$
- Yêu cầu : tìm dãy con có trọng số lớn nhất

Ví dụ

- dãy số : -2, 11, -4, 13, -5, 2
- Dãy con có trọng số cực đại là 11, -4, 13 có trọng số 20

Có bao nhiêu dãy con ?

- Số lượng cặp (i,j) với $1 \leq i \leq j \leq n$
- $\binom{n}{2} + n$
- Thuật toán trực tiếp !

Thuật toán trực tiếp — $\mathcal{O}(n^3)$

- Duyệt qua tất cả $\binom{n}{2} + n = \frac{n^2+n}{2}$ dãy con
- Với mỗi dãy con ta tính tổng của dãy
- Lưu lại tổng lớn nhất

```
1  public long long algo1(int[] a) {
2    int n = a.length;
3    long max = a[0];
4    for(int i = 0; i < n; i++){
5      for(int j = i; j < n; j++){
6        int s = 0;
7        for(int k = i; k <= j; k++)
8          s = s + a[k];
9        max = max < s ? s : max;
10     }
11   }
12   return max;
13 }
```

Thuật toán tốt hơn — $\mathcal{O}(n^2)$

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

```
1  public long algo2(int[] a){
2    int n = a.length;
3    long max = a[0];
4    for(int i = 0; i < n; i++){
5      int s = 0;
6      for(int j = i; j < n; j++){
7        s = s + a[j];
8        max = max < s ? s : max;
9      }
10   }
11   return max;
12 }
```

Thuật toán Quy hoạch động

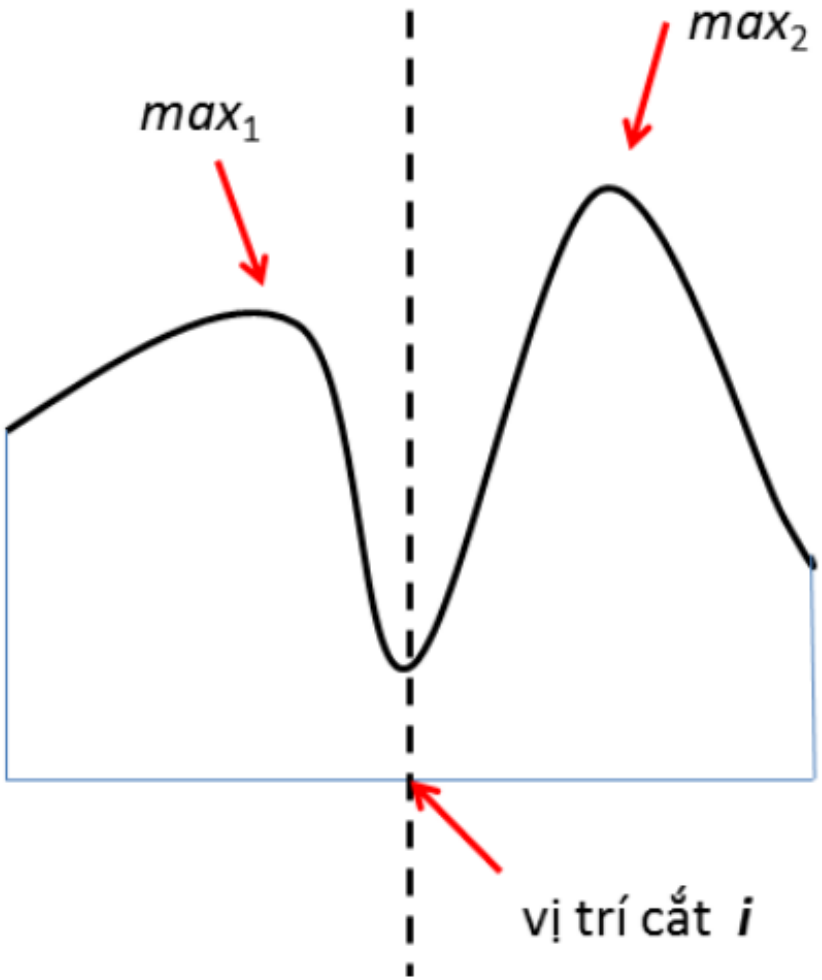
- Thiết kế hàm tối ưu :
  - Đặt s_i là trọng số của dãy con có trọng số cực đại của dãy $a_1, \ldots, a_i$ mà kết thúc tại a_i
- Công thức Quy hoạch động :
  - $s_1 = a_1$
  - $s_i = \max\{s_{i-1} + a_i, a_i\}, \forall i = 2, \ldots, n$
  - Đáp án là $\max\{s_1, \ldots, s_n\}$
- Độ phức tạp thuật toán là $O(n)$ (thuật toán tốt nhất !)

Quy hoạch động — $\mathcal{O}(n)$

```
1  public long algo4(int[] a){
2    int n = a.length;
3    long max = a[0];
4    int[] s = new int[n];
5    s[0] = a[0];
6    max = s[0];
7    for(int i = 1; i < n; i++) {
8      // tinh s[i]
9      if(s[i-1] > 0) s[i] = s[i-1] + a[i];
10     else s[i] = a[i];
11     max = max > s[i] ? max : s[i];
12   }
13   return max;
14 }
```

SIGNAL

- Cho một dãy tín hiệu độ dài n có độ lớn lần lượt là $a_1, a_2, ..., a_n$ và một giá trị phân tách b.
- Một tín hiệu được gọi là phân tách được khi tồn tại một vị trí i $(1 < i < n)$ sao cho $max\{a_1, .., a_{i-1}\} - a_i \geq b$ và $max\{a_{i+1}, .., a_n\} - a_i \geq b$
- Tìm vị trí i phân tách được sao cho $max\{a_1, .., a_{i-1}\} - a_i + max\{a_{i+1}, .., a_n\} - a_i$ đạt giá trị lớn nhất.
- In ra giá trị lớn nhất đó. Nếu không tồn tại vị trí phân tách được thì in ra giá trị -1.


max_1
max_2
vị trí cắt i

Thuật toán

- Chuẩn bị mảng $maxPrefix[i] = max\{a_1, .., a_i\}$.
- Chuẩn bị mảng $maxSuffix[i] = max\{a_i, .., a_n\}$
- Duyệt qua hết tất cả các vị trí i ($1 < i < n$). Với mỗi vị trí kiểm tra xem liệu đó có phải là vị trí phân tách được hay không bằng cách kiểm tra
  $maxPrefix[i-1] - a[i] >= b$ và
  $maxSuffix[i+1] - a[i] >= b$.
- Lấy max của giá trị
  $maxPrefix[i-1] - a_i + maxSuffix[i+1] - a_i$ tại các vị trí i thoả mãn.
- Độ phức tạp thuật toán O(n).

Cài đặt

```
16      // Nhap du lieu
17      Nhap n, b
18      Nhap day so a
19
20      // tinh max_prefix, min_prefix
21      Duyet qua tat ca vi tri i
22          // tinh max_prefix va max_suffix bang cong thuc
23          max_prefix[i] = max(max_prefix[i - 1], a[i]);
24          max_suffix[i] = max(max_suffix[i + 1], a[i]);
25
26
27      // tinh ket qua
28      Duyet qua tat ca vi tri i
29          // Kiem tra dieu kien vi tri i
30          Neu (max_prefix[i - 1] - a[i] >= b &&
31              max_suffix[i + 1] - a[i] >= b)
32              // Neu thoa man thi lay ket qua
33              ans = max(ans, max_prefix[i - 1] - a[i] +
34                                 max_suffix[i + 1] - a[i]);
35
36      // dua ra ket qua
37      In ra ans
```

REROAD

- Cho N đoạn đường, đoạn thứ i có loại nhựa đường là t_i.
- Định nghĩa một phần đường là một dãy liên tục các đoạn đường được phủ cùng loại nhựa phủ t_k và bên trái và bên phải phần đường đó là các đoạn đường (nếu tồn tại) được phủ loại nhựa khác.
- Độ gập ghềnh của đường bằng tổng số lượng phần đường.
- Mỗi thông báo bao gồm 2 số là số thứ tự đoạn đường được sửa và mã loại nhựa được phủ mới.
- Sau mỗi thông báo, cần tính độ gập ghềnh của mặt đường hiện tại.

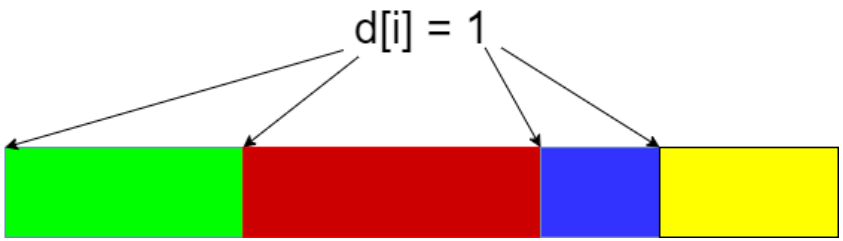
Ví dụ

Đoạn đường ban đầu với độ gập ghềnh là 4

Đoạn đường sau khi update với độ gập ghềnh là 6

Thuật toán

- Gọi d[i] là mảng nhận giá trị 1 nếu $a[i] \neq a[i-1]$ và giá trị 0 trong trường hợp ngược lại
- Nhận thấy mỗi phần đường có một và chỉ một phần tử bắt đầu, số lượng phần đường (hay độ gập ghềnh) chính là số lượng phần tử bắt đầu.
- Nói cách khác thì độ gập ghềnh = $\sum_{i=1}^{n} d[i]$
- Nhận thấy với mỗi lần đổi 1 phần tử i trong mảng a thì ta chỉ thay đổi giá trị của nhiều nhất là 2 phần tử trong mảng d đó là $d[i]$ và $d[i+1]$



Code

```
39  // ham tinh phan tu u cua mang d
40  int d(int u) {
41      if (u == 1) return 1;
42      return a[u] != a[u - 1];
43  }
44
45
46  int main() {
47      Nhap so luong phan tu n
48      Nhap day so a
49      anwer = 0;
50      Duyet vi tri i tu 1 den n
51          answer += d(i);
52      Nhap so luong truy van q
53      Duyet het q truy van
54          Nhap vi tri thay doi p va gia tri thay doi c
55
56          // Tru di gia tri d truoc do
57          answer -= d(p);
58          Neu p khong phai phan tu n
59              answer -= d(p + 1);
60
61          // Cong vao cac gia tri d moi
62          a[p] = c;
63          answer += d(p);
64          Neu p khong phai phan tu n
65              answer += d(p + 1);
66          In ra answer
67  }
```