Bài toán tìm dãy con lớn nhất

Trong buổi học này, ta xem xét bài toán sau.

- **Input:** Dãy số $a[0],\ a[1],\ \ldots,\ a[n-1]$.
- **Output:** Dãy con liên tiếp $a[i],\ a[i+1],\ \ldots,\ a[j]$ có tổng lớn nhất.

Ví dụ: Dãy $-2, 11, -4, 13, -5, 2$ có tổng lớn nhất là 20, là tổng của dãy con $11, -4, 13, -5$.

Thuật toán trực tiếp

Ý tưởng: Duyệt mọi dãy con có thể và tính tổng mỗi dãy con để tìm ra dãy có tổng lớn nhất.

Câu hỏi: Có bao nhiêu dãy con có thể?

```
// Tìm số lớn nhất trong hai số
int max (int x, int y) {
    if (x > y) return x;
    else       return y;
}
// Thuật toán đơn giản
int maxSub1 (int a[], int N)
{
    int maxsum = INT32_MIN;  //âm vô cùng
    int sum = 0;
    for (int i = 0; i < N; i++) {
        for (int j = i; j < N; j++){
            sum = 0;
            for (int k = i; k <= j; k++) {
                sum = sum + a[k];
            }
            maxsum = max (sum, maxsum);
        }
    }
    return maxsum;
}
```

Câu hỏi: Thuật toán này dùng bao nhiêu phép cộng?

Sinh dữ liệu để test và chương trình chính

Bạn có thể sử dụng mã nguồn dưới đây để test thời gian chạy của hàm `maxsub1` ở trên.

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;
// Sinh dữ liệu ngẫu nhiên để test
```

```
void genRandom (int a[], int N) {
    int r = 0;
    srand (time(NULL));
    for (int i = 0; i < N; i++) {
        r = rand() % 100;
        if (r > 70) a[i] = -r;    // 30% là số âm
        else a[i] = r;            // 70% là số dương
    }
}

#define N 100000000
int a[N]; // = {-2, 11, -4, 13, -5, 2};

// Chương trình chính
int main(int argc, char const *argv[]) {
    genRandom(a,N);
    cout <<maxsub1(a, N)<<endl;
    return 0;
}
```

Khi thực hiện chương trình, bạn có thể thêm lệnh `time` phía trước chương trình chạy để có thông tin thời gian chạy của chương trình. Các lệnh dưới đây để biên dịch và chạy chương trình.

```
$ g++ s1.cc -o maxsub
$ time ./maxsub
19162787
./maxsub  2.25s user 0.10s system 99% cpu 2.349 total
```

Thuật toán nhanh hơn

Cải tiến thuật toán trước dựa trên công thức

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

```
// Thuật toán đơn giản cải tiến
int maxSub2 (int a[], int N) {
    int maxsum = INT32_MIN;  //âm vô cùng
    int sum = 0;
    for (int i = 0; i < N; i++) {
        sum = 0;
        for (int j = i; j < N; j++) {
           sum = sum + a[j];
            maxsum = max (sum, maxsum);
        }
    }
    return maxsum;
}
```

Câu hỏi: Thuật toán này dùng bao nhiêu phép cộng?

Thuật toán đệ quy

Ý tưởng: Dùng kỹ thuật chia để trị.

Ta chia dãy thành hai dãy dùng phần tử ở giữa và thu được hai dãy (gọi là dãy bên trái và dãy bên phải) với độ dài giảm đi một nửa. Để tổ hợp lời giải, ta nhận thấy rằng chỉ có thể xảy ra một trong ba trường hợp:

1. Dãy con lớn nhất nằm ở nửa bên trái;
2. Dãy con lớn nhất nằm ở nửa bên phải; và
3. Dãy con lớn nhất bắt đầu ở nửa trái và kết thúc ở nửa phải.

Vì vậy nếu ký hiệu w_L, w_R và w_M, tương ứng, là tổng dãy con lớn nhất bên trái, bên phải và ở giữa thì tổng dãy con lớn nhất của dãy ban đầu sẽ là $\max\{w_L, w_R, w_M\}$.

- Việc tìm dãy con lớn nhất bên trái và dãy con lớn nhất bên phải có thể thực hiện một cách đệ quy.
- Để tìm tổng của dãy con ở giữa ta thực hiện như sau:
  - Tính tổng của dãy con lớn nhất trong nửa trái từ $a[i]$ đến $a[j]$ và kết thúc ở phần tử $a[j]$. Để tính tổng này ta dùng hàm `maxleft(a, i, j)`.
  - Tính tổng của dãy con lớn nhất trong nửa phải từ $a[i]$ đến $a[j]$ bắt đầu ở phần tử $a[i]$. Để tính tổng này ta dùng hàm `maxright(a, i, j)`.

```
/* Cài đặt dùng đệ quy, chia để trị */
// Tìm dãy con lớn nhất trong đoạn a[i],...,a[j] và kết thúc tại a[j]
int maxleft (int a[], int i, int j)
{
    int maxsum = INT32_MIN;
    int sum = 0;
    for (int k = j; k >= i; k--){
        sum = sum + a[k];
        maxsum = max(sum, maxsum);
    }
    return maxsum;
}
// Tìm dãy con lớn nhất trong đoạn a[i],..., a[j] và bắt đầu tại a[i]
int maxright (int a[], int i, int j){
    int maxsum = INT32_MIN;
    int sum = 0;
    for (int k = i; k <= j; k++){
        sum = sum + a[k];
        maxsum = max(sum, maxsum);
    }
    return maxsum;
}
// Tìm dãy con lớn nhất trong đoạn a[i], a[i+1], ..., a[j]
int maxsub(int a[], int i, int j){
    if (i == j) return a[i];
    int m = (i + j)/2;
    int wL = maxsub (a, i, m);
    int wR = maxsub (a, m+1, j);
    int wM = maxleft(a, i, m) + maxright(a, m+1, j);
    return max (max(wL, wR), wM);
```

```
}
// Dùng đệ quy, chia để trị
int maxsub3 (int a[], int N) {
    return maxsub (a, 0, N-1);
}
```

Câu hỏi:** Thuật toán này dùng bao nhiêu phép cộng?

Thuật toán Quy hoạch động

Ta ký hiệu:

- e_i là dãy con lớn nhất của dãy $a[0], a[1], \ldots, a[n-1]$ **kết thúc** tại $a[i]$.

Do dãy con có tổng lớn nhất phải kết thúc ở $a[i]$ nào đó nên `maxsum` có thể tính bởi công thức:

$$\text{maxsum} = \max\{e_i \mid i = 0, 1, \ldots, n-1\}$$

Bây giờ ta xem xét cách để tính các e_i từ e_{i-1}. Ta chia hai trường hợp: Dãy con lớn nhất của dãy $a[0], a[1], \ldots, a[n-1]$ kết thúc tại $a[i]$ thỏa mãn:

1. dãy **có ít nhất hai phần tử**, tức là có chứa cả $a[i-1]$ và $a[i]$. Khi đó, theo định nghĩa $e_i = e_{i-1} + a[i]$;
2. dãy **chỉ** có một phần tử $a[i]$. Khi đó $e_i = a[i]$.

Từ hai trường hợp này, ta có công thức truy hồi sau cho phép tính e_i từ e_{i-1}:

$$\begin{cases} e_0 = a[0] \\ e_i = \max\{a[i],\ e_{i-1} + a[i]\} \end{cases}$$

Với các nhận xét trên, ta dễ dàng đi tới chương trình.

```
//Dùng quy hoạch động
int maxsub4 (int a[], int N) {
    int maxsum = a[0];
    int e = a[0];
    for (int i = 1; i < N; i++) {
        e = max(a[i], e + a[i]);
        maxsum = max (maxsum, e);
    }
    return maxsum;
}
```

Câu hỏi: Thuật toán này dùng bao nhiêu phép cộng?

Bài tập về nhà

Các cài đặt trên chỉ cho phép ta tính tổng của dãy con lớn nhất mà không chỉ rõ đoạn nào của dãy. Hãy sửa đổi các thủ tục trên để hiện ra màn hình cặp giá trị (i, j). Cặp này chỉ ra dãy $a[i], a[i+1], \ldots, a[j]$ là dãy có tổng lớn nhất.

Bạn phải viết một báo cáo ngắn khoảng 1-2 trang mô tả các sửa đổi mã nguồn của bạn cho từng thuật toán.