Dự tuyển olympic tin học

Dãy số Fibonacci

Bài toán LÁT GẠCH

Một sân chơi của trẻ em có dạng hình chữ nhật kích thước 2 x N ($N < 10^{10}$).

Hãy tính xem có bao nhiêu cách khác nhau để lát sân chơi bởi các viên gạch cỡ 1 x 2 sao cho không có phần gạch nhỏ nào thừa ra ngoài và không có vùng diện tích nào của sân chơi không được lát.

Nếu số lượng cách lát quá nhiều, hãy in ra 9 chữ số cuối của số cách đếm được.

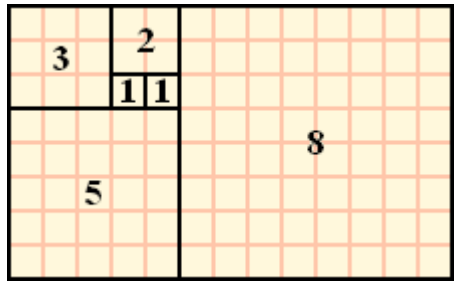
Dãy số fibonacci

- Nhắc đến trong sách của Leonardo Fibonacci năm 1202
  - Về sau mới được Lucas gọi là dãy số fibonacci (thế kỷ 19)
  - Trước đó một số nhà toán học Ấn Độ cũng đã nói đến dãy này
- Xuất phát từ quan sát trong tự nhiên
- Dãy: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,…
- Định nghĩa:

  F(0) = 0

  F(1) = 1

  F(n) = F(n-1) + F(n-2)



Tính số fibonacci thứ n

```
int F1(int n) {
     if (n > 1) return F1(n-1) + F1(n-2);
     return n;
}

int F2(int n) {
     return (n > 1) ? F2(n-1) + F2(n-2) : n;
}

int F3(int n) {
     int a[n+1] = {0, 1};
     for (int i = 2; i <= n; i++)
          a[i] = a[i-1] + a[i-2];
     return a[n];
}
```

Tính số fibonacci thứ n

```
int F4(int n) {
	if (n < 2) return n;
	int a = 0, b = 1, c;
	for (int i = 2; i <= n; i++) {
		c = a + b; a = b; b = c;
	}
	return b;
}

int F(int n) {
	if (n < 2) return n;
	int a = 0, b = 1;
	for (int i = 2; i <= n; i++) {
		b = a + b; a = b - a;
	}
	return b;
}
```

Tính chất của dãy số fibonacci

$F(n + 1) = F(n) + F(n - 1)$

$F(0) + F(1) + F(2) + \ldots + F(n) = F(n + 2) - 1$

$F(1) + 2\,F(2) + 3\,F(3) + \ldots + n\,F(n) = n\,F(n + 2) - F(n + 3) + 2$

$F(n+1)\,F(n-1) - F(n)^2 = (-1)^n$

$F(n+k) = F(n)\,F(k-1) + F(n+1)\,F(k)$

$GCD(F(m), F(n)) = F(GCD(m,n))$

$$F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

$$F_n = \left[\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n}{\sqrt{5}}\right]$$

Tính nhanh số fibonacci thứ n

- Dựa trên tính chất: $F(n+k) = F(n)\,F(k-1) + F(n+1)\,F(k)$
- Nếu n = k:
  - $F(2k) = F(k)\,(F(k+1) + F(k-1))$
  - $F(2k+1) = F(k+1)^2 + F(k)^2$
- Như vậy để tính F(n) ta cần tính F(k+1), F(k) và F(k-1)
- Ta có thể sử dụng kĩ thuật đệ quy có nhớ để tính theo phương pháp này
- Lợi:
  - Số lượng giá trị cần nhớ giảm theo $\log_2 n$
  - Số bước tính toán cũng là $O(\log_2 n)$

Tính nhanh số fibonacci thứ n

```
map<int, int> C;

C[0] = 0, C[1] = 1, C[2] = 1

int F(int n) {
    if (C.count(n) > 0) return C[n];
    int k = n / 2;
    if (n % 2 == 1)
        return C[n] = F(k+1) * F(k+1) + F(k) * F(k);
    else
        return C[n] = F(k) * (F(k + 1) + F(k-1));
}
```

Tính nhanh số fibonacci thứ n hơn nữa!

- Dựa trên tính chất: $F(n+k) = F(n)\,F(k-1) + F(n+1)\,F(k)$
- Nếu n = k:
  - $F(2k) = F(k)\,(F(k+1) + F(k-1))$
  - $F(2k+1) = F(k+1)^2 + F(k)^2$
- Tiếp tục biến đổi, nếu n = k:
  - $F(2k) = F(k)\,(F(k+1) + F(k-1)) = F(k)\,(2F(k+1) - F(k))$
  - $F(2k+1) = F(k+1)^2 + F(k)^2$
- Như vậy ta chỉ cần tính F(k) và F(k+1)
  - Sửa đổi: Mỗi lần tính toán ta tính một lượt cả F(n) và F(n+1)
  - Không sử dụng cache nữa
- Độ phức tạp vẫn cỡ $O(\log_2 n)$ nhưng nhanh hơn đáng kể

Tính nhanh số fibonacci thứ n hơn nữa!

```
pair<int, int> F(int n) {
    if (n == 0) return {0, 1};

    auto fk = F(n / 2);
    int a = fk.first * (2 * fk.second - fk.first);
    int b = fk.first * fk.first + fk.second * fk.second;
    if (n % 2)
        return {b, a + b};
    else
        return {a, b};
}
```

Số fibonacci thứ n dưới góc nhìn ma trận

- Dễ chứng minh công thức

$$(F_{n-1} \quad F_n) = (F_{n-2} \quad F_{n-1}) \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

- Đặt biến phụ $P \equiv \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$

- Ta viết công thức trên thành $(F_n \quad F_{n+1}) = (F_0 \quad F_1) \cdot P^n$
- Về bản chất việc tính mũ của ma trận P có thể tính nhanh tương tự như tính mũ số nguyên A^n
  - Cách làm trước (ở slide 10) bản chất là phiên bản tinh tế của kĩ thuật nhân ma trận
  - Kĩ thuật này còn sử dụng trọng nhiều bài khác