NHẬP MÔN PHÂN TÍCH ĐỘ PHỨC TẠP THUẬT TOÁN

Bài tập 2

Viết bài thu hoạch phân tích thuật toán kiểm tra một dãy phần tử có phải là dãy tăng hay không.

Nguyễn Minh Đức

1712358

I. Trình bày đề bài và thuật toán

Đề bài: Cho mảng A[n], với n phần tử ($n \geq 2$) trong A lấy ngẫu nhiên không hoàn lại từ tập B có m phần tử khác nhau (m >= n), giả sử rằng các phần tử trong B đều có tính thứ tự, mỗi phần tử chỉ có thể nhỏ hơn hoặc lớn hơn khi so sánh với phần tử khác trong B. Viết một thuật toán để kiểm tra xem A có phải là dãy tăng hay không. Tức kiểm tra điều kiện

$$A[1] < A[2] < \cdots < A[n]$$

có đúng hay không.

Thuật toán Alg giải quyết bài toán trên trình bày như sau:

```
Alg(A, n):
    for i from 1 to n-1:
        if A[i] > A[i+1] then
            return false
    return true
```

Dễ thấy thuật toán có độ phức tạp thời gian $\theta(n)$ bởi vòng lặp có thể lặp tối đa n-1 lần (khi mảng được sắp tăng sẵn hoặc sai 1 vị trí cuối cùng), và trong vòng lặp for thì ta chỉ thực hiện 1 lượng constant các lệnh gồm so sánh và return.

Nhưng trong thực tế độ phức tạp trung bình của thuật toán có lẽ không hẳn là $\theta(n)$. Để ý thấy độ phức tạp của Alg có cùng độ tăng với số vòng lặp for, có cùng độ tăng với số lần so sánh A[i] A[i+1], từ đó liên quan mật thiết tới thứ tự của dữ liệu đầu vào.

Vì vậy mục đích của bài thu hoạch này là tìm ra độ phức tạp trung bình của thuật toán kiểm tra dãy tăng (tức kiểm tra số lần lặp, kiểm tra số phép so sánh) với điều kiện dữ liệu được lấy ngẫu nhiên như trong đề bài.

II. Phân tích lý thuyết

Giả sử số lần lặp của vòng lặp for là $f(n)$, với c là số phép toán trong mỗi lần lặp ($c \leq 2$), suy ra thuật toán có độ phức tạp $\theta(cf(n)) = \theta(f(n))$. Vậy chỉ cần tìm ra số lần lặp là ta sẽ suy ra được độ phức tạp của thuật toán.

Alg sẽ ngừng ở $1 \leq k \leq n-1$ vòng lặp khi gặp biến cố

$$A[1] < A[2] < \cdots < A[k] \cap A[k] > A[k+1]\ (*)$$

Ngoài ra khi Alg ngừng ở $n-1$ vòng lặp có thể do gặp được biến cố (*) hoặc gặp biến cố A là mảng tăng.

Từ đây nếu đặt $P_{k,n}$ là xác suất xảy ra biến cố (*) và P_n là xác suất để A là mảng tăng (ở đây không xuất hiện tham số m, sẽ giải thích ở bổ đề 1 dưới đây) thì số vòng lặp trung bình của thuật toán là:

$$f(n) = \left[\sum_{k=1}^{n-1} P_{k,n} * k\right] + P_n * (n-1) \quad (1)$$

Bổ đề 1:

Hình thành một mảng A bằng cách rút ngẫu nhiên không hoàn lại n phần tử từ tập B có m phần tử khác nhau (m >= n), khi đó các xác suất $P_{k,n}, P_n$ xét trên mảng A không phụ thuộc vào m.

Chứng minh

Gọi X là biến cố rút được một mảng A n phần tử có thứ tự ngẫu nhiên từ tập B. Khi đó không gian mẫu cho X có kích cỡ P_m^n (chỉnh hợp chập n của m). Nhưng $P_m^n = C_m^n * n!$, tức ta có thể hình thành mảng A theo một cách tương đương: Chọn ra n phần tử từ tập B trước rồi sau đó hoán vị n phần này sau. Từ ý này ta có thể chia không gian mẫu thành C_m^n tập con T rời nhau, mỗi tập con T chứa $n!$ hoán vị của một bộ phần tử nào đó lấy từ B.

Vì các xác suất $P_{k,n}, P_n$ chỉ phụ thuộc vào thứ tự lớn nhỏ giữa các phần tử mà không phải là bản thân giá trị phần tử đó nên $P_{k,n}, P_n$ sẽ không đổi khi tính trên mọi bộ n phần tử khác nhau bất kỳ. Nên miễn là T còn đủ $n!$ hoán vị thì $P_{k,n}, P_n$ cũng sẽ không đổi. Và cũng bởi vì $P_{k,n}, P_n$ không đổi trên mọi tập con rời nhau của không gian mẫu nên $P_{k,n}, P_n$ cũng sẽ không đổi khi tính trên không gian mẫu đó. Từ đó suy ra điều phải chứng minh.

Suy ra ta có thể tính $P_{k,n}, P_n$ qua việc hoán vị ngẫu nhiên một mảng A cố định bất kỳ nào đó.

Bổ đề 2:

Khi xáo trộn ngẫu nhiên một mảng A có n phần tử khác nhau thì xác suất để chỉ $k \le n-1$ phần tử đầu tiên được sắp xếp tăng dần là:

$$P_{k,n} = \frac{k}{(k+1)!}$$

Chứng minh

Chọn ra $k+1$ phần tử làm các phần tử đầu cho A, suy ra phần sau của A có $n-k-1$ phần tử. Suy ra có C_n^{k+1} cách chọn.

Xét dãy $k+1$ phần tử đầu tiên này, sẽ luôn có một cách xếp duy nhất sao cho các phần tử tăng dần, với k phần tử cũng vậy. Vậy nếu ta lấy ra một phần tử không phải là lớn nhất xếp vào vị trí $k+1$ rồi xếp k phần tử đầu tiên tăng dần thì ta sẽ được một dãy tăng chỉ dài k phần

tử liên tục (do phần tử lớn nhất nằm ở vị trí k sẽ lớn hơn phần tử ở vị trí $k+1$). Mà có k cách chọn phần tử cho vị trí $k+1$ nên cũng có k cách chọn $k+1$ đầu cho A thỏa mãn.

Lại có $(n-k-1)!$ hoán vị cho phần sau A và A có $n!$ hoán vị tất cả nên tổng kết lại xác suất để chỉ k phần tử đầu tiên của A tăng dần là:

$$P_{k,n} = \frac{C_n^{k+1} * k * (n-k-1)!}{n!}$$

$$P_{k,n} = \frac{\frac{n!}{(k+1)!\,(n-k-1)!} * k * (n-k-1)!}{n!}$$

Rút gọn lại ta được

$$P_{k,n} = \frac{k}{(k+1)!}$$

Từ bổ đề 1, ta có thể tính $P_{k,n}, P_n$ độc lập với m qua việc hoán vị ngẫu nhiên một mảng cố định có n phần tử khác nhau. Khi này chỉ có một hoán vị duy nhất trong số $n!$ hoán vị là được sắp xếp tăng dần. Suy ra

$$P_n = \frac{1}{n!}$$

Từ bổ đề 2 ta cũng tính được $P_{k,n}$ độc lập với m, thay vào (1) ta có số lần lặp trung bình:

$$f(n) = \left[\sum_{k=1}^{n-1} \frac{k}{(k+1)!} * k\right] + \frac{1}{n!} * (n-1)$$

Phân tích một chút ta sẽ thấy f(n) bị chặn trên bởi một số hữu hạn

$$f(n) = \left[\sum_{k=1}^{n-1} \frac{k(k+1) - k - 1 + 1}{(k+1)!}\right] + \frac{n-1}{n!}$$

$$f(n) = \sum_{k=1}^{n-1} \frac{k(k+1)}{(k+1)!} - \sum_{k=1}^{n-1} \frac{k+1}{(k+1)!} + \sum_{k=1}^{n-1} \frac{1}{(k+1)!} + \frac{n}{n!} - \frac{1}{n!}$$

$$f(n) = \sum_{k=1}^{n-1} \frac{1}{(k-1)!} - \sum_{k=1}^{n-1} \frac{1}{k!} + \sum_{k=1}^{n-1} \frac{1}{(k+1)!} + \frac{1}{(n-1)!} - \frac{1}{n!}$$

Để ý thấy

$$\sum_{k=1}^{n-1} \frac{1}{(k-1)!} - \sum_{k=1}^{n-1} \frac{1}{k!} = \frac{1}{0!} + \frac{1}{1!} + \cdots + \frac{1}{(n-2)!} - \frac{1}{1!} - \frac{1}{2!} - \cdots - \frac{1}{(n-1)!} = 1 - \frac{1}{(n-1)!}$$

Từ đó ta tính tiếp f(n)

$$f(n) = 1 - \frac{1}{(n-1)!} + \sum_{k=1}^{n-1} \frac{1}{(k+1)!} + \frac{1}{(n-1)!} - \frac{1}{n!}$$

$$f(n) = 1 + \sum_{k=1}^{n-1} \frac{1}{(k+1)!} - \frac{1}{n!}$$

$$f(n) = 1 + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!} - \frac{1}{n!}$$

$$f(n) = \sum_{k=1}^{n-1} \frac{1}{k!} = \left(\sum_{k=0}^{n-1} \frac{1}{k!}\right) - 1$$

Ta biết rằng $\sum_{k=0}^{\infty} \frac{1}{k!} = e$ nên suy ra

$$f(n) < e - 1$$

Tức $f(n)$ bị chặn trên bởi $e - 1 \approx 1.71828$, trong khi đó Alg phải thực hiện ít nhất một vòng lặp ban đầu, suy ra

$$1 \leq f(n) < e - 1$$

Vậy thuật toán Alg có độ phức tạp trung bình hằng số $\theta(1)$

Thật ngạc nhiên, tới đây độ phức tạp trung bình lại là một con số khá nhỏ và không phụ thuộc vào n. Điều này có lẽ chỉ đúng với dữ liệu hoàn toàn ngẫu nhiên như trên, khi ấy $A[1] > A[2]$ đã chiếm hết một nửa các trường hợp, xác suất để k phần tử đầu tiên tăng dần giảm rất nhanh khi k lớn dần, nên dù n có lớn tới vô cùng thì trung bình số lần lặp cũng chỉ tiệm cận $e - 1$.

III. Kết quả chạy thực nghiệm

Hàm Alg() được viết bằng C, kiểm tra một mảng số nguyên A, n phần tử có phải là mảng tăng hay không

```
bool Alg(int *A, int n, int & SS, int & DPT)
{
    DPT++;
    for (int i = 0; DPT++,i < n - 1;DPT++,i++) {
        if (SS += 1, DPT += 1, A[i] > A[i + 1]) {
            DPT++; return false;
        }
    }
    DPT++;
    return true;
}
```

Các biến SS và DPT được đưa vào hàm để đếm số phép toán và số vòng lặp.

Kết quả chạy hàm Alg() dưới đây được đo trên 1000 mảng A ngẫu nhiên, 100 phần tử

Số vòng lặp	Số lượng phép toán	Số lượng mảng A
1	4	501
2	7	347
3	10	117

4	13	24
5	16	10
7	22	1

Ví dụ kết quả dòng đầu tiên trong bảng thể hiện rằng trong 1000 lần chạy Alg() thì có 501 lần hàm chỉ thực hiện đúng một vòng lặp (so sánh đúng một lần), số phép toán thực hiện là 4.

Từ đó ta tính được số vòng lặp trung bình là 1.699 và số phép toán trung bình là 6.097

Kết quả này rất trùng với tính toán lý thuyết, em đã chạy ngẫu nhiên Alg() rất nhiều lần và thu được số vòng lặp trung bình thường dao động quanh 1.7

Thử chạy lại Alg() 1000 lần với các mảng A chỉ có 10 phần tử, được kết quả như sau:

Số vòng lặp	Số lượng phép toán	Số lượng mảng A
1	4	513
2	7	322
3	10	121
4	13	41
5	16	5

Số vòng lặp trung bình là 1.709, số phép toán trung bình đã thực hiện là 6.127.

Kết quả của mảng A 10 phần tử không khác gì so với kết quả ở mảng 100 phần tử, em đã thử với mảng 1000 phần tử và mảng 10000 phần tử, số phép so sánh luôn dao động rất nhỏ quanh con số 1.7. Chỉ ngoại trừ ở các mảng nhỏ có chưa tới 10 phần tử sẽ có số vòng lặp trung bình dao động trong khoảng 1.5 1.6. Từ đó cho thấy độ phức trung bình của Alg() trên các bộ số ngẫu nhiên thực sự là hằng số.