MỘT SỐ PHƯƠNG PHÁP TỐI ƯU THUẬT TOÁN

ThS. Nguyễn Vân Giang

I. ĐẶT VẤN ĐỀ

Chúng ta đều biết là thuật toán giữ vai trò trung tâm trong cuộc Cách mạng công nghiệp 4.0, đặc biệt là đối với các bài toán phức tạp hay dữ liệu lớn. Vì thế để đáp ứng được nhu cầu về nguồn nhân lực chất lượng cao, có trình độ về thuật toán thì các trường và giảng viên phải quan tâm đến việc dạy cho SV khả năng tư duy thuật toán và các phương pháp tối ưu thuật toán trong lập trình.

Thực tế cho thấy nhiều GV quan niệm là bậc trung cấp và cao đẳng thì chỉ cần dạy SV biết lập trình nên không chú ý rèn SV thói quen cải tiến hay tối ưu chương trình. Nhưng như thế sẽ làm triệt tiêu tư duy sáng tạo và thói quen tối ưu chương trình. Đây chính là các yêu tố cần thiết để trở thành lập trình viên chuyên nghiệp.

Vì tầm quan trọng của thuật toán và tối ưu thuật toán mà hầu hết các cuộc thi về lập trình trên thế giới hiện nay đều thi về thuật toán. Các công ty lớn luôn có bài test về khả năng tư duy thuật toán khi tuyển lập trình viên. Ví dụ công ty Samsung cũng tổ chức cuộc thi về thuật toán hàng năm với quy mô toàn cầu (Samsung SCPC) để thu hút các lập trình viên giỏi làm việc cho mình.

II. MỘT SỐ PHƯƠNG PHÁP TỐI ƯU THUẬT TOÁN

Với mỗi bài toán thường có nhiều thuật toán để đạt được kết quả cuối cùng. Tối ưu thuật toán là tìm ra thuật toán tối ưu hơn về thời gian hay bộ nhớ để giải quyết bài toán đã cho.

Có nhiều phương pháp để tối ưu thuật toán. Với trình độ Cao đẳng thì ngoài cách dùng quy hoach động hoặc dùng công thức toán học để giảm số vòng lặp,

trong khuôn khổ bài tham luận này tôi xin trình bày một số kinh ngiệm tối ưu thuật toán đơn giản của bản thân tôi qua quá trình giảng dạy và bồi dưỡng các sinh viên giỏi tham dự kỳ thi Olympic Tin học Sinh viên Toàn quốc và cuộc thi lập trình ACM/ICPC. Đây là những phương pháp thường được sử dụng trong các kỳ thi lập trình và có thể áp dụng trong quá trình giảng dạy các môn Lập trình của khoa.

Trong các kỳ thi lập trình, nếu SV sử dụng những thuật toán không tối ưu thường chỉ đạt khoảng 30% số điểm vì chương trình sẽ chạy quá thời gian quy định (thường là 1s) hoặc tràn bộ nhớ với những bộ test có dữ liệu lớn.

1. Lưu trữ vào mảng theo giá trị số để tránh thao tác sắp xếp

Nếu mảng gồm toàn số nguyên và giá trị lưu trữ không lớn ta có thể dung cách lưu trữ theo giá trị để giảm thời gian xử lý.

Ví dụ 1: Cần sắp xếp dãy số nguyên dương a_0, a_1,….,a_{N-1} tăng dần với $N \le 10^5$, $a[i] \le 1000$

Cách 1: Dùng QuickSort, có độ phức tạp O(nlogn).

- Nếu $N \ge 10^4$ chương trình chạy quá TG quy định.
- Nếu đề mở rộng cho N lớn hơn, ví dụ $N=10^9$ thì cách này sẽ phá sản vì không thể khai báo mảng

Cách 2: Lưu vào mảng theo giá trị số, có độ phức tạp O(n).

Dãy 2, 4, 3, 1, 4,… sẽ được lưu như sau:

0	1	2	3	4	…	1000
0	1	1	1	2	…	

Lưu ý:

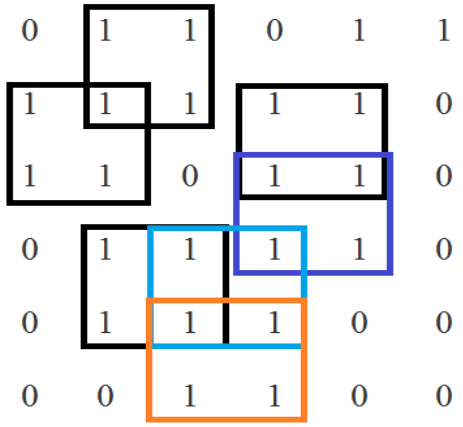
- Nếu a[i]<0 vẫn dùng được cách này nhưng phải tịnh tiến vị trí lưu trữ

2. Dùng mảng 1 chiều thay vì 2 chiều, dùng biến tạm thay vì dùng mảng.

Rất nhiều bài toán xử lý trên mảng 2 chiều nhưng thực tế không cần lưu trữ trên mảng 2 chiều mà chỉ cần một mảng tạm 1 chiều để xử lý. Tương tự với mảng 1 chiều đôi khi có thể được thay thế bằng một biến tạm. Phương pháp này vừa giúp tiết kiệm bộ nhớ, vừa giúp giảm thời gian xử lý dữ liệu.

Ví dụ 2: Cho ma trận kích thước N x N ($10 \le N \le 10^4$) với mỗi ô (I,J) ($1 \le I, J \le N$) có giá trị 0 hoặc 1. Đếm số hình vuông kích thước K x K ($1 \le K \le 40$) mà tất cả các giá trị trong hình vuông đều phải khác 0.

Ví dụ với N=6, K=2 ta đếm được 7 hình vuông



<u>Cách 1</u>: Độ phức tạp $O(N^2xK^2) \approx O(N^3)$

Duyệt từng phần tử của ma trận, với mỗi phần tử dương kiểm tra xem có tồn tại hình vuông KxK bắt đầu từ điểm này hay không ?

Cách này sẽ phá sản nếu số N lớn hơn hoặc bộ nhớ không đủ để cấp phát mảng 6.400.000

<u>Cách 2</u>: Độ phức tạp $O(N^2)$. Cách nhanh nhất cho bài này vì phải đọc NxN giá trị các phần tử của ma trận.

Không lưu dữ liệu vào mảng 2 chiều mà chỉ dùng mảng 1 chiều N phần tử để lưu giá trị tính toán trung gian: int a[N] ={0}

Với mỗi dòng dữ liệu dung biến d để đếm xem có bao nhiêu số 1 liên tục nhau.

Nếu tại vị trí i mà d ≥ K thì tăng a[i] lên 1 đơn vị và kiểm tra : nếu a[i]>=K thì tăng biến KQ đếm số hình vuông lên 1, ngược lại cho a[i]=0.

Ví dụ: Bảng sau mô tả giá trị của mảng a[] sau khi đọc dữ liệu của từng dòng

Dòng	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
0	0 (d=0)	0 (d=1)	1 (d=2)	0 (d=0)	0 (d=1)	1 (d=2)
1	0 (d=1)	1 (d=2)	2 (d=3, KQ=1)	1 (d=3)	1 (d=4)	0 (d=0)
2	0 (d=1)	2 (d=2, KQ=2)	0 (d=0)	0 (d=1)	2 (d=2, KQ=3)	0 (d=0)
3	0 (d=0)	0 (d=1)	1 (d=2)	1 (d=3)	3 (d=4, KQ=4)	0 (d=0)
4	0 (d=0)	0 (d=1)	2 (d=2, KQ=5)	2 (d=3, KQ=6)	0 (d=0)	0 (d=0)
5	0 (d=0)	0 (d=0)	0 (d=1)	3 (d=2, KQ=7)	0 (d=0)	0 (d=0)

3. Đơn giản hóa bài toán để tính trực tiếp kết quả

Khi gặp các bài toán thực tế người lập trình thường hay bị rối bởi những mô tả có vẻ phức tạp của bài toán. Đôi khi chỉ cần suy nghĩ một cách đơn giản để nắm bắt được bản chất của bài toán thì ta có thể dễ dàng tìm được lời giải.

Ví dụ: Xét bài 3 đề Olympic Tin học không chuyên 2007

Nhật kí hành trình

Đoàn thám hiểm sa mạc Gô bi xuất phát từ điểm có tọa độ (XS,YS) sau N ngày khảo sát sa mạc đã hoàn thành xuất sắc các nhiệm vụ đề ra và về tới đích an toàn ở điểm có tọa độ(XD,YD). Đầu mỗi ngày trong cuộc hành trình, khi mặt trời còn chưa kịp tỏa ánh nắng chói chang như muốn thiêu đốt mọi sinh vật trên sa mạc, đoàn thám hiểm di chuyển tới điểm khảo sát mới, cách điểm hiện tại một đơn vị độ dài và đi theo một trong số 4 hướng: Đông (E), Bắc (N), Tây (W) hoặc Nam (S). Như vậy, nếu ban đầu trước khi lên đường vị trí của đoàn ở tọa độ (X, Y) thì vị trí mới nơi đoàn thực hiện các khảo sát và ngủ qua đêm sẽ là như sau:

Hướng đi Tọa độ X mới Tọa độ Y mới

E: X+1 Y

N: X Y+1

W:X-1 Y

S: X Y-1

Đường đi được ghi lại trong nhật kí công tác dưới dạng xâu kí tự T chỉ chứa các kí tự thuộc tập {E, N, W, S}.

Ví dụ, từ điểm xuất phát XS = 1, YS = 2, với hành trình T = ’ENWNEEESESWWSW’, điểm đích của chuyến khảo sát sẽ là XD = 2, YD = 1.

Tùy theo yêu cầu thực tế, một điểm có thể được quay lại khảo sát nhiều lần. Mọi việc dường như vô cùng tốt đẹp nếu như không có một sự cố nhỏ: báo cáo công tác (in trên máy vi tính) bị trả về để bổ sung chỉnh l. vì trong xâu T xác định hành trình có một số kí tự lạ không chỉ hướng đi, tức là không thuộc tập {E, N,W, S}!

Ví dụ, hành trình trên có thể bị gõ nhầm vào máy vi tính thành T = 'ENWNEYESEZWWSW'.

Cần phải sửa lại các kí tự sai để nhận được xâu đảm bảo từ điểm xuất phát (XS,YS) hành trình sẽ kết thúc ở điểm (XD,YD). Hai cách sửa được gọi là khác nhau nếu cho các xâu mô tả hành trình khác nhau. Yêu cầu: hãy giúp xác định có bao nhiêu cách sửa các kí tự sai để có một xâu hành trình đúng và đưa ra xâu T có thứ tự từ điển nhỏ nhất.

Dữ liệu: vào từ file văn bản TRACE.INP gồm 2 dòng:

- Dòng thứ nhất ghi 4 số nguyên XS, YS, XD, YD, có giá trị tuyệt đối không quá 106, giữa các số cách nhau một dấu cách.

- Dòng thứ hai ghi xâu T có độ dài không quá 255 chứa không quá 10 kí tự lạ.

Kết quả: ghi ra file văn bản TRACE.OUT:

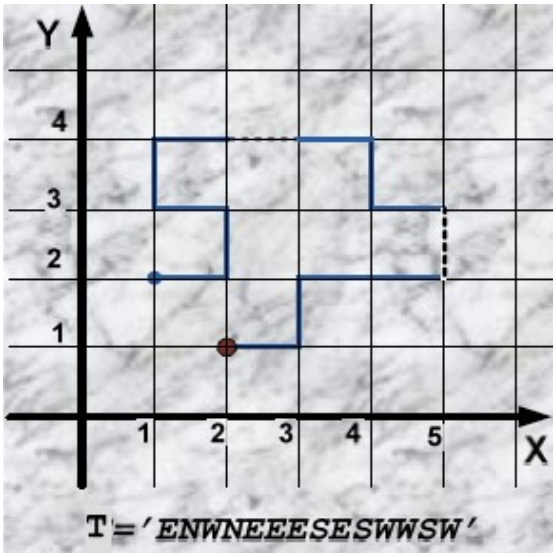
- Dòng thứ nhất chứa một số nguyên là số cách sửa;

- Dòng thứ hai chứa xâu T đã chỉnh l. có thứ tự từ điển nhỏ nhất.

Ví dụ:

TRACE.INP
1 2 2 1 ENWNEYESEZWWSW

TRACE.OUT
2 ENWNEEESESWWSW



Cách 1: Độ phức tạp $O(4^N xM)$ với N là số ký tự lạ, M là độ dài chuỗi

Vét cạn các trường hợp (Có 4^{10} =1.048.576 trường hợp):

- Dùng phương pháp đệ quy quay lui để sinh ra các trường hợp điền ký tự. Với mỗi trường hợp sẽ đi thử, nếu vị trí kết thúc đúng như đề bài thì tăng biến đếm số cách sửa và đồng thời so sánh chuỗi đang thử hiện với chuỗi min hiện thời để tìm chuỗi nhỏ nhất.

Cách 2: Độ phức tạp O(M) với M là độ dài chuỗi

(Đây là cách tính trực tiếp đáp số dựa vào độ lêch tọa độ giữa điểm đi và điểm đến)

- Duyệt chuỗi dữ liệu để cập nhật (XS,YS) nếu gặp ký tự lạ lưu vị trí ký tự lạ vào mảng a[10] và đếm số ký tự lạ d.
- So sánh XD với YS, XD với YD để biết số số ký tự E,W, N và S cần điền (lưu vào biến E, W, N, S)

  Ví dụ XD-XS=2 có nghĩa là số ký tự E lớn hơn W là 2. Lưu biến E=2.

- Nếu m = d - (E+W+S+N) > 0 có nghĩa là trong các ký tự lạ có cặp ký tự E và W hay N và S cùng tồn tại. Chú ý m luôn là số chẵn.

  Ví dụ nếu m>0 có nghĩa là có thêm m/2 cặp ký tự ngược hướng , ví dụ E,W hoặc N,S. Nhưng chuỗi kết quả sẽ chọn cách 1 để được chuỗi nhỏ hơn theo thứ tự từ điển)

- Lần lượt điền các ký tự vào chuỗi ban đầu tại các vị trí đã lưu.

  Ví dụ: Nếu E=1, W=0, S=1, N=0, m=2.

  Thì sẽ điền theo thứ tự lần lượt như sau: E-E-S-W

  Để tính số cách sửa ta dùng công thức chỉnh hợp lặp.

  Nếu m=0 : Kq= n!/E!W!S!N!

  Ngược lại:

  $$Kq= n!/((E+m/2)!(W+m/2)!S!N) + \ n!/(E!W!(S+m/2)+!(N+m/2)!)$$

III. KẾT LUẬN

Với bậc đào tạo Trung cấp và Cao đẳng thì để rèn luyện khả năng tư duy thuật toán và kỹ năng tối ưu chương trình cho SV không nhất thiết là phải dạy những giải thuật phức tạp mà thông qua những bài giảng và bài tập trong môn dạy về lập trình của mình, GV cần ý thức dạy cho SV cách tư duy để tìm ra thuật toán cũng như yêu cầu phải cải tiến và hoàn thiện chương trình tùy theo khả năng của từng đối lượng SV. Bên cạnh đó mỗi GV cũng cần quan tâm học hỏi và tự bồi dưỡng để góp phần nâng cao chất lượng đào tạo và truyền ngọn lửa đam mê lập trình cho các SV của mình.