Phân tích về QHĐ - Thầy Lê Minh Hoàng

Phân tích về QHĐ - Thầy Lê Minh Hoàng

Nguồn: Facebook VNOI

Khi có 1 bạn hỏi Quy hoạch động trạng thái là gì trong bài QBSELECT , thầy Lê Minh Hoàng có trả lời như sau:

Bạn cứ hiểu quy hoạch động là quy hoạch động. Đừng quan tâm đến cách thức lưu trữ (một chiều, hai chiều, rời rạc...), cũng đừng quan tâm tới thứ tự tính toán (quét ngang, băm chéo, trên cây, trên DAG, trên đường, dưới ao...) thì sẽ thấy vấn đề là do dân competitive programmers tự thêm thuật ngữ mới vào, chứ bản chất rất đơn giản:

Dynamic Programming = Solving Recurrence + Memoization

Ví dụ như cái bài QBSELECT, chắc bạn tiếp cận và dựng công thức truy hồi như thế này: cách chọn trong cái bảng gồm n cột nó bao hàm cách chọn trong $n-1$ cột. Suy nghĩ tự nhiên là ta thử mọi cách chọn trong cột cuối, với mỗi cách đó thử khớp với cách chọn tối ưu nhất trong $n-1$ cột đầu mà không xung đột với cột cuối => Với mỗi cách chọn trong $n-1$ cột đầu, ta phải phân loại theo cách chọn cột $n-1$ để khớp với cách chọn trong cột n sau này. Bài toán trở thành dựng một hàm mục tiêu $f(x)$ trong đó biến x xác định bài toán.

Vậy để xác định bài toán, biến x cần chứa đủ thông tin: Ta đang tính tới cột nào và ở cột cuối cùng ta chọn các ô như thế nào. Cái biến x này người ta gọi là tham số xác định bài toán hay dân Việt gọi là "Trạng Thái". Tức là biến x = một bộ {i, a, b, c, d} cho biết tính đến cột i, a = có chọn (1, i) không; b = có chọn ô (2, i) không, c = có chọn ô (3, i) không và d = có chọn ô (4, i) không.

Bây giờ mới nghĩ về cách thức lưu trữ và tính toán. Ta biết rằng hàm mục tiêu $f(x)$ xác định bởi giá trị $x=(i,a,b,c,d)$. Trong đó a, b, c, d chỉ là 0 hoặc 1 => có thể nén 4 tham số này vào 4 bit của một biến nguyên. Bạn cũng có thể để rời và lưu trữ hàm mục tiêu dưới dạng mảng 5 chiều `f[x] = f[i][a][b][c][d]...` Nhưng tùy vào cách thức triển khai tính toán mà bạn tìm cách mã hóa x theo cách nào hợp lý nhất. nhằm thuận lợi cho các thao tác:

- Xác định xem một bài toán đã giải chưa, nếu giải rồi thì giá trị hàm mục tiêu = mấy?
- Giải công thức truy hồi và lưu trữ lời giải

Chẳng hạn nếu bạn giải kiểu quét ngang bảng phương án (for i = 1....n) thì không nên mã hóa cái i vào cùng với a, b, c, d. Nếu bạn giải bằng TOP-DOWN + Memoization (mà ta thường gọi là đệ quy có nhớ) thì chẳng cần mã hóa. Cứ giải xong `f(i, a, b, c, d)` nào lưu nó vào một tập hợp S, muốn xem một bài giải chưa thì tìm nó trong S....
Tập S có thể cài bằng Hash-Map (Pascal) hoặc Unordered-Map(C++11) có tốc độ lưu trữ và truy vấn O(1)...

Nói như vậy mọi bài QHĐ đều phải tìm cách xác định bài toán, tức là không gian các x = các trạng thái của nó. Hay nói cách khác QHĐ là QHĐ trạng thái và ngược lại.

Những chuyện xử lý bit, nén, ... là encoding, nó là kỹ thuật cơ bản không liên quan gì tới QHĐ hết mà chỉ dùng để mã hóa bài toán. Những chuyện DFS, Topological Sorting cũng là để xác định thứ tự tính, chứ không phải đặc trưng của QHĐ. Bạn nên coi các cái này là kỹ thuật bổ trợ nhằm tăng tính hiệu quả của phép cài đặt mà thôi.

Còn về căn bản như bài toán ở trên, mọi ct truy hồi ta có thể giải kiểu dùng hash-map, lúc đó ta chẳng cần quan tâm tới encoding và thứ tự tính nữa.