Một kĩ thuật tính tổng tích lũy đơn giản mà hiệu quả

Ngày hôm nay là một ngày đẹp trời ! Mình đã được truyền đạt lại một kĩ thuật khá đơn giản nhưng rất hiệu quả để tính tổng tích lũy. Kĩ thuật này đối với mình thì rất là xa lạ, nhưng thật ra nó khá phổ biến đối với những học sinh chuyên Tin ở những nơi khác. Kệ, học được bao nhiêu tốt bấy nhiêu ^.^

Vấn đề đặt ra như sau: Cho một dãy số a[1..n] và danh sách m lệnh có dạng (x, y, k) - tăng tất cả các phần tử a[x..y] lên k đơn vị (k có thể là số âm, 0 hoặc dương). Yêu cầu in ra dãy a[1..n] sau m lệnh đó.

Trước giờ mình chỉ nghĩ tới việc sử dụng Brute-Force: "Với từng lệnh, xét từng phần tử thuộc đoạn a[x..y] rồi đem cộng với k" mà thôi (Độ phức tạp O(n2)), hoặc cùng lắm, dùng cây chứa khoảng với độ phức tạp O(nlogn) là nhanh lắm rồi. Bởi thế, tri thức mới này làm cho mình có phần sốc khi thực sự hiểu ra, và đương nhiên, rất vui ^.^

Kĩ thuật này như sau: Xét từng lệnh trong m lệnh đó, giả sử đang xét tới lệnh (i, j, t), ta sẽ gán lại các giá trị a[i] và a[j + 1] như sau:

a[i] := a[i] + k;

a[j + 1] := a[j + 1] - k;

Sau đó, ta chỉ cần:

for i := 2 to n do a[i] := a[i - 1] + a[i];

Thì dãy a[1..n] cuối cùng sẽ là dãy kết quả cần tìm !! Thời gian chính xác là O(2m + n) !!

Lí giải tại sao, ta chỉ cần chú ý rằng các lệnh (x, y, k) là riêng biệt, tức là thứ tự thực hiện các lệnh không ảnh hưởng gì tới kết quả, và với mỗi đoạn (i, j), bởi vì phần tử đầu tiên (a[i]) được cộng thêm t, do đó trong bước cộng dồn vào các phần tử sau, đương nhiên các phần tử phía sau đều được cộng thêm một lượng là t (đúng!), còn tới phần tử a[j + 1] vì đã trừ mất t rồi, nên cộng t vào sẽ không thay đổi nữa, do đó, làm theo các bước trên rõ ràng sẽ cho ta kết quả.

Và tới giờ này mình vẫn luôn cố gắng tìm được một con đường để giúp cái đầu mình thông minh hơn. Tại sao mình chưa bao giờ từng nghĩ ra điều này nhỉ ? Tại vì trong đầu mình luôn luôn thường trực một tư tưởng sắt thép rằng: "Vấn đề này không thể giải quyết trong thời gian tuyến tính được đâu, quá rõ ràng mà!" ... và giờ niềm tin này đã nát vụn. Bởi thế, mình (và mong các bạn, những ai cũng giống mình lúc này) sẽ luôn luôn thay đổi cách suy nghĩ để tìm tới những điều tốt đẹp hơn. "Không đúng đâu, chưa tối ưu đâu, có thể nhanh hơn nữa, có thể tốt hơn nữa"