Bài 1: Tích chập

Cách 1: Giả sử chênh lệch vị trí chọn giữa hai dãy là d, xây dựng dãy c, với $c_i = a_i * b_{(i+d)}$. Bài toán đưa về tìm đoạn con có tổng lớn nhất của dãy c

Cách 2: Gọi f[i][j] là tích chập lớn nhất kết thúc tại a_i và b_j.

Khi đó f[i][j] = max(a[i]*b[j] + f[i-1][j-1], a[i]*b[j])

Bài 2: Chia kẹo

Gọi w là tổng 1 phân (w=tổng n gói / k)

Sub1: Duyệt tất cả các hoán vị của 1, 2, ..., n. Một hoán vị được gọi là thỏa mãn và tương ứng với duy nhất một cách chia thỏa mãn nếu có thể chia hoán vị thành k đoạn, các đoạn có tổng bằng nhau,bằng w. Việc chia này đc thực hiện bằng cách đi từ đầu hoán vị đến cuối, lần lượt lấy từng gói kẹo cho đến khi đúng đủ w thì tiếp tục lấy phần tiếp theo, ...

Sub2: Sử dụng ý tưởng hoán vị ở sub1, quy hoạch động bitmask, f(t) = true/false có nghĩa là có thể xếp các gói thuộc tập t thành một đoạn đầu trong hoán thỏa mãn hay không?

Từ f(t)=true, nếu gói i (không thuộc t) có thể xếp vào cuối đoạn hoán vị đang xây dựng tổng các gói kẹo thuộc phần hiện tại cộng với gói thứ i <=w thì f(t|2^i) = true

Sub3: Quy hoạch động f(i, a, b) = true/false có nghĩa là xét các gói từ 1 đến i, có thể lấy một số gói kẹo cho vào phần 1 có tổng là a, phần 2 có tổng là b, tổng của phần 3 là tổng các gói kẹo (từ 1 đến i) trừ đi a + b.

F(i, a, b) = true, thì f(i+1, a + g[i], b) – lấy vào phần 1, f(i+1, a, b+g[i]) – lấy vào phần 2, f(i+1, a, b) – lấy vào phần 3, ba giá trị này đều sẽ nhận bằng true.

Sub4, 5: lời giải với n, được truy hồi về (n-2k), các gói n-2k+1 đến n chia làm k phần, mỗi phần gồm 2 gói có tổng bằng nhau (gói n-2k+1 với n một phần, n-2k+2 với n-1,...). Truy hồi về n <=50 thì dùng duyệt hoặc tham xếp nốt.

Bài 3: Truy vấn

Sub1: Với mỗi truy vấn, sắp xếp lại các số trong S và tính tổng bình thường. Độ phức tạp Q^2*log(Q)

Sub2: Xem mỗi số nguyên trong S là một xâu nhị phân gồm 18 bit. Dùng một cây trie để quản lý tập xâu này. Việc chèn xóa vào S chính là chèn xóa xâu trên trie. Mỗi nút của trie lưu thông tin về tổng của các số nằm trong cây con đó. Truy vấn tổng sẽ đi dọc trên cây từ nút gốc đến nút lá có thứ tự k, và tính tổng các nhánh nằm bên trái. Độ phức tạp Q*log(x)

Sub3: Dùng thêm một biến X để lưu tổng xor của các truy vấn loại 2. Khi thêm x vào S, thay vì chèn x thì ta chèn x xor X vào trie. Tương tự với phép xóa x. Cấu trúc gồm trie và X cho phép tìm số lớn thứ i trong độ phức tạp log(x), bằng cách đi từ gốc xuống nút lá, ở độ sâu thứ t, tùy thuộc vào bit thứ t của X là 0 / 1 mà ta ưu tiên nhánh trái / nhánh phải trước. Cho i chạy từ 1 đến k ta tính được tổng của k số nhỏ nhất. Đô phức tạp Q*k*log(x)

Sub4: Mỗi nút của trie lưu 18 thông tin, thông tin thứ t là số lượng phần tử nằm trong nút đó mà có bit thứ t bật. Lúc này có thể tính được tổng của các phần tử nằm trong một nút bằng cách tính trên từng bit. Độ phức tạp Q*log^2(x)