Bài 3:

Tập xor:

Tính chất x+y = x^y tương đương với việc x và y không có bit 1 nào trùng nhau

Subtask 1 (1 <= n,k,a\_i <= 100):

Gọi dp(i, j, mask) là số cách chọn thêm j phần tử từ {a\_i, ..., a\_n} sao cho các phần tử không trùng bit 1 với nhau hay với mask. Khi đó dp(i,j,mask) = dp(i+1,j,mask) + [a\_i+mask == a\_i^mask]dp(i+1,j-1,mask|a\_i). Từ việc đếm được các nghiệm ta cũng sẽ chuyển sang tính tổng các nghiệm được.

Subtask 2 (1 <= n,k,a\_i <= 1000):

Việc bổ sung a\_i vào tập khiến số bit 1 của mask tăng lên, nên chỉ có thể bổ sung không quá log(a) phần tử. Do đó chỉ cần xét đến j <= log(a) và xử lý tương tự subtask 1

Subtask 3 (1 <= n,k <= 10000, a\_i=i):

Nếu các phần tử thoả mãn tính chất về tổng xor thì chúng khác nhau đôi một. Do đó ta có thể bỏ qua thứ tự lấy: Gọi dp(j, mask) là số cách lấy j phần tử từ {1, 2, ..., n} sao cho bit 1 không trùng nhau hay trùng mask. Chuyển nhãn sẽ cần xét hết các số có bit 1 không trùng với mask, tổng độ phức tạp để chuyển nhãn là 3^log2(n) ~ n căn n. Kết quả sẽ cần chia cho j! do bị trùng lặp

Subtask 4 (1 <= n, k, a\_i <= 10000):

Xử lý tương tự subtask 3 nhưng cần nhân hệ số chuyển nhãn với số lần xuất hiện ở trong dãy ban đầu của phần tử sẽ lấy

Subtask 5 (1 <= n, k <= 10^1000, a\_i = i):

Ta sẽ xây dựng các bộ (x\_1, x\_2, ..., x\_t) với ràng buộc n >= x\_1 > x\_2 ... > x\_t >= 1 và t <= k. Việc xây dựng sẽ thực hiện trên từng bit, mỗi lần sẽ mở rộng thêm 1 bit của tất cả các số cùng một lúc.

Khi xây dựng đến bit thứ i, ta cần quan tâm số lượng t các phần tử đã lấy, và kiểm soát x\_1 <= n. Bộ tham số để thoả mãn ràng buộc là (i, t, ok) nên độ phức tạp là O(log(n)^2). Để chuyển nhãn, ta có thể thêm bit 0 vào sau tất cả các x\_, hoặc thêm đúng 1 bit 1 vào, hoặc thêm một phần tử x\_{t+1} = 2^i. Do đó chi phí chuyển nhãn là hằng số