Week1:

Binarygen,combgen,permgen - .\_. …

Permutationlist:

Gọi F[i] là số lượng xâu có hoán vị có độ dài I (cthức n!) F[i] = F[i-1]\*I;

Gọi mảng Fre[i] là cái số nó có xuật hiện trong xâu nghiệm hay chưa , res[i] là xâu nghiệm tại cấu hình K mang giá trị tại vị trí I là res[i]

Thì việc còn lại for i=1->n;

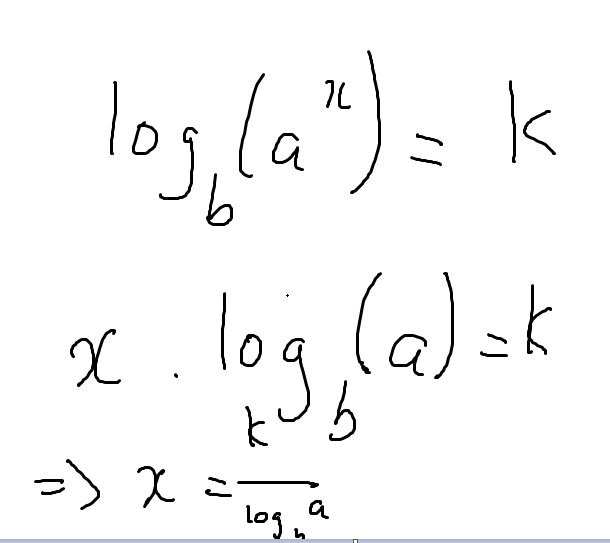
Nếu mà tại vị trí thứ I, ta for j = 1->N nếu K>F[n-i] thì trừ đi, việc ta phải làm như vậy là vì, với mỗi giá trị I, thì seẽ có F[n-i] bộ nghiệm thỏa , ta trừ đi như vậy sẽ tìm được cái thằng mình cần tìm

Sau khi biết vị trí thằng ko trừ dc nữa ( đến đó là K ko thuộc) thì gán res[i] = j; rồi fre[j] = false;

CombList: Tương tự, đổi mảng F thành Tam giác pascal + điều kiện so sánh Comb(n-j,k-1)

Fibval: Sinh ra từ từ thấy nó loop tại vị trí thứ 15, -> Chu trình

Bài LOG



Bài C(K,N) % M

Phân tích n! thành tsnt,k! thành tsnt, (n-k)! thành tsnt rồi lấy trên trừ dưới -> quy về phép nhân rồi đồng dư dpt nlog1e6

Week2:

Blackbox: Dựa trên tư tưởng là mình luôn luôn duy trì 1 dãy tăng dần để tìm thằng K cho dễ, thì em dùng vector, chặt nhị phân tìm vị trí rồi nhét vào, thằng thứ K thì O(1) do mảng đã sorted.Em thử sinh max test máy em nó vẫn chạy được .\_. nên e nghĩ thuật OK

Connect: Dựng lại đồ thị với mỗi đỉnh là điểm đề cho trước, trọng là khoảng cách rồi tìm cây khủng (em dùng Kruskal)

Đẳng cấu: sau khi xem hint và nháp 1 hồi, em nhận thấy là cái số lượng xâu thỏa là 1 hằng số dựa trên cái số lần xuất hiện (e ngồi code trâu sinh ra từ từ .-. ) …

DemBoTu:

Do A[i],B[j],C[k],D[z] <= |1e6| nên em sinh tổng 2 thằng đầu ra, tổng 2 thàng sau ra(sum A+B), (Sum C+D) ném tụi nó vào 2 mảng lùa rồi 1 for từ -1e6->1e6 vét

TourDB: Liệt kê TPLT Mạnh, với mỗi TPLT Mạnh thì số chu trình đơn là m-n+1(m sốcạnh,nsố đỉnh)

FROG: Stack, đi từ cuối dãy vào, Với thằng Thứ I, ta xét coi có bao nhiêu thằng <= A[i] pop hết ra khỏi stack,vị trí của con ếch sau khi nhảy chính là vị trí sau khi nhảy trong stack, push A[i] vào stack và lặp lại quá trình

Dzi:

Đầu tiên xét theo hàng, gọi là mảng H, nếu A[i][j] = A[i-1][j] = 0 thì H++ ko thì H=1

Gọi L[i] là phần từ trái nhất mà A[L[i]] >= A[i]

R[i] là ptử phải nhất mà A[r[i]] >= A[i] -> tại mỗi vị trí I ta tính dc HCN là , H[i] \* (R[i] – L[i]+1) -> cập nhập res

RGD: Gọi F[i][j] là giá trị thu được khi rút gọn dãy từ vị trí I->J một cách tối ưu

Ta sẽ đi vét K trong [I,J] để tối ưu hóa cho F[i][j], F[i][j] = min(F[i][j],F[i][k-1]+F[k+1][j]+a[k-1]\*a[k]\*a[K+1]);

STONE: Gọi F[i] là số sỏi tối thiểu khi đi hết gốc I, ta tiến hành DFS(v) với v kề I để tính các F[v] lân cận,

Thì cách chọn tối ưu là mình sort tăng dần theo các F[v]. Có 2 biến F[i] = 0; thua = 0;

Thì với mỗi lần mình duyệt qua F[v] đã sorted thì sẽ có 2 trường hợp , nếu thua < f[i] nghĩa là không đủ để phân phối -> f[u] += f[i] – thua; và thua = f[i] -1; (đề bài kêu bỏ i-1 vào trong túi ban đầu) còn không thì thua—( do rich as fuck rồi đủ để cho các nút con)

Sum2arr: sort a,sort b, dựng 1 heapmin,push toàn bộ a[1] + b[j] vào heap. Với mỗi lần póp ra,giả sử pop 1 thằng X,có vị trí trong mảng a là I, trong mảng B là j thì push(a[i+1]+b[j]) vào

Làm đến khi nào ko dc nữa

Week 3:

Ladder: Gọi L[i] là vị trí phải nhất mà bắt đầu từ vị trí I tới vị trí phải nhất đấy là 1 dãy ko giảm

R[i] là vị trí trái nhất mà từ I tới vị trí trái nhất tạo nên 1 dãy ko giảm

Tính mảng L,R bằng deque while(a[l[i]] <= a[l[i]+1])l[i] = l[l[i]+1]; cái R tương tự

Kingpath. BFS + Map

Đầu tiên với mỗi đoạn thẳng, nhét vào Map cái vị trí của nó (hàng cột), Push đỉnh (x,y) vào queue. Với mỗi lần pop ra, ta kiểm tra xem các đỉnh kề với (x,y) theo 8 hướng có trong Map hay ko , nếu có thì đẩy vào queue và bỏ mấy đỉnh đó ra khỏi map. Lặp lại quá trình đến khi đến được (x0,y0) nếu không đến được in ra -1

Digit:1->9 mỗi lần độ dài +1, 10->99 + 2, 100->999 +3, cứ thế mà vét …

Domino Dựng lại đồ thị dựa theo các quân domino đã cho, rồi tiến hành tìm đường đi euler, nếu tồn tại 1 đường đi thì in ra đường đi đấy ko thì in ra -1

Tomau: Đầu tiên giả sử ta có hàm DFS(u,bl,wh,last) với ý nghĩa, u đỉnh đang xét, bl số đỉnh đen đã dc tô, wh số đỉnh trắng đã dc tô last đỉnh cuối dc tô là màu gì. Tại đỉnh u, ta xét nếu u là màu đen thì dfs(v,bl,wh+1,trắng), nếu là đen thì ngược lại, và đặt cận nếu số đỉnh chưa tô + số đỉnh bl <= max đen tìm dc thì loại, bl+white = n thì cập nhập bblack

Dua do. Loại toàn bộ những đoạn có l<r (vì ta tiện đường công tác đi từ 1->m luôn nên skip tụi nó)

Việc còn lại sẽ còn lại những đoạn [l,r] với R>l, ta đảo ngược tụi nó lại, sort tăng dần theo x, nếu x bằng nhau tăng dần theo y, ta lần lượt xét lại những đoạn đó, gọi left , right là vị trí trái nhất và phải nhất mà ta có thể mở rộng dc, nếu mà đoạn left,right giao với đoạn thứ i, ta mở rộng tụi nó ra, tới khi nào ko giao nữa thì ta cộng res = (right-left) \* 2; (đi từ đầu này về đầu kia + ngược lại) . xong hết mọi thứ ta cộng thêm res 1 đoạn = m(do đi từ 1->m) res là đáp án bài toán :D . tư tưỡng tham

Scale: Nếu giới hạn 10^13 thì chỉ việc sinh nhị phân ... còn 10^100 em chưa nghĩ

Thu âm: Sparse table min,max :<