**HƯỚNG DẪN GIẢI MỘT SỐ CÂU**

**TRONG ĐỀ THI CHUNG KẾT ICPC PTIT NĂM 2024**

**BÀI A. BIỂU THỨC TOÁN HỌC**

* Ta cần đặt N – 1 phép toán vào giữa N số
* N = 10 🡪 9 vị trí. Mỗi vị trí có 3 cách đặt toán tử, do đó, số trường hợp cần xét là 3⁹ =19683 ~2e4. Giới hạn bé như vậy thì đây là một bài sinh kế tiếp/quay lui cơ bản. (Thực chất là sinh chỉnh hợp lặp chập 3 của (N - 1) phần tử)
* Tuy nhiên: a[i] lên đến 10⁹, chỉ cần nhân liên tục 3 số đã lên đến 1027, vượt quá giới hạn kiểu dữ liệu long long trong C++. Nên tính đến đâu phải mod đến đấy. Chú ý rằng phép cộng và phép nhân cũng có tính phân phối trong modulo.
* Sau khi ghép xong dấu vào N – 1 vị trí rồi, ta sẽ được một biểu thức trung tố như dạng hay viết hàng ngày. Nhưng máy tính không thể hiểu được biểu thức trung tố, bắt buộc ta phải chuyển về biểu thức tiền tố hoặc hậu tố rồi dùng stack để giải
* Về nguyên tắc chung là như vậy, nhưng rất may đây là biểu thức không có dấu ngoặc, nên ta chỉ cần nhân trước, cộng trừ sau. Khi tính tay, ta sẽ thực hiện hết mọi phép nhân, được bao nhiêu cộng dồn lại với nhau. Dựa trên việc này, không cần dùng stack xử lý, ta sẽ làm như sau:
* Biến res lưu kết quả cuối cùng
* Biến tmp: Lưu kết quả của phép nhân gần nhất
* Cách thực hiện: Ta sẽ duyệt qua từng phép toán trong dãy phép toán +, -, \* mà ta đã sinh/quay lui (Tương đương với việc duyệt các phần tử a[2] … a[N] cùng với dấu đứng trước nó).
* Nếu trước a[i] là phép nhân thì nhân tiếp a[i] vào tmp
* Nếu trước a[i] là phép toán + hoặc -: Lúc này, dãy nhân đã bị gãy, nên ta cộng dồn kết quả phép nhân gần nhất vào res đồng thời reset lại phép nhân thành +a[i], nếu trước a[i] là dấu +, hoặc -a[i] nếu trước a[i] là dấu -
* Thuật toán chuyển từ trung tố về hậu tố hoặc tiền tố khá dài, nhờ việc xử lý khéo léo ta đã tránh được việc này.

**BÀI B. SỐ MAY MẮN ĐẶC BIỆT**

* Khi so sánh 2 số dưới dạng xâu (Trường hợp không có số 0 thừa ở đầu):
* Số nào ít chữ số hơn thì số đó là số nhỏ hơn
* 2 số cùng số chữ số, số nào có thứ tự từ điển đứng trước thì nhỏ hơn
* Ý tưởng: Ta sẽ duyệt theo độ dài, cố gắng độ dài ngắn nhất có thể. Trong mỗi độ dài ta xét, cố gắng số chữ số 6 nhiều nhất có thể
* Chuẩn bị trước 2 mảng: pow[i] = 10i % m, one[i] = (11…1 🡪 i số 1) % m
* Giả sử số tìm được gồm a số 8, b số 6. Khi đó:
* Giá trị của a số 8 là A = 8 \* one[a], giá trị của b số 6 là B = 6 \* one[b]
* Khi ghép lại thành số hoàn chỉnh, a số 8 phải thêm b số 0 ở cuối để nhường chỗ cho b số 6. Mà thêm b số 0 ở cuối chính là nhân với 10b.
* Như vậy, số hoàn chỉnh khi lấy dư cho m sẽ là:

R = A \* 10b + B = 8 \* one[a] \* pow[b] + 6 \* one[b]

* Nếu R = 0, đây là số cần tìm
* Triển khai thuật toán:
* Tiền xử lý mảng one và pow như trên; Duyệt độ dài từ 1 đến 200
* Với mỗi độ dài, duyệt số chữ số 8 = a từ 0 đến 200 (Ưu tiên a ít để b = len – a nhiều để số có thứ tự từ điển nhỏ)
* Tính A, B, R như trên. Nếu thoả mãn thì in a số 8 và b số 6 liền nhau.
* Tổng số trường hợp tối đa cần xét với toàn bài:

100 (Số Test) \* 200 (Số cách chọn độ dài) \* 200 (Số cách chọn số chữ số 8) = 4.106, đủ nhanh trong 1s

**BÀI C. TÍCH CỦA DÃY SỐ**

**Yêu cầu:** Tìm số lượng dãy số có N phần tử mà tích cả dãy đúng bằng M

* M = p1x1 \* p2x2 \* … \* ptxt
* Ta cũng phân tích từng số trong dãy ra thừa số nguyên tố, từ đó biểu diễn theo các thừa số của M như sau:
* a[1] = p1a11 \* p2a12 \* … \* pta1t
* a[2] = p1a21 \* p2a22 \* … \* pta2t
* …
* a[N] = p1aN1 \* p2aN2 \* … \* ptaNt
* Như vậy, tích các phần tử trong dãy là:

S = a[1] \* a[2] \* … \* a[N]

= (p1a11 \* p2a12 \* … \* pta1t) \* (p1a21 \* p2a22 \* … \* pta2t) \* … \* (p1aN1 \* p2aN2 \* … \* ptaNt)

* Rút gọn tích các luỹ thừa cùng cơ số với nhau, ta có:

S = p1a11 + a21 + … + aN1 \* p2a12 + a22 + … + aN2 \* … \* pta1t + a2t + … + aNt

* Ta cần S = M, đồng nhất số mũ ta có:
* a11 + a21 + … + aN1 = x1 (1)
* a12 + a22 + … + aN2 = x2 (2)
* …
* a1t + a2t + … + aNt = xt (t)

Nghĩa là ta cần đếm số nghiệm nguyên không âm của t phương trình này

* ***Xét một số bài toán cơ bản:***

**Bài toán 1:** Phương trình x1 + x2 + … + xk = N có bao nhiêu nghiệm nguyên dương

**Giải:** Ta viết số N thành N số 1 trên cùng 1 hàng: 1 1 … 1 1 1

Cần chia N số 1 này thành k phần, mỗi phần có ít nhất 1 số 1

Giữa N số 1 có N – 1 khe trống, ta cần chọn k khe trống để đặt thanh chắn, nhằm chia thành k phần 🡪

**Bài toán 2:** Phương trình x1 + x2 + … + xk = N có bao nhiêu nghiệm nguyên không âm?

**Giải:**

+) Ta sẽ biến đổi các số hạng vế trái thành số dương để quy về bài toán 1:

x1 ≥ 0 🡪 x1 + 1 ≥ 1 > 0.

+) Ta cộng vào mỗi số hạng vế trái thêm 1, phương trình trở thành:

(x1 + 1) + (x2 + 1) + … + (xk + 1) = N + k

**+)** Đặt x1 + 1 = y1, x2 + 1 = y2, … , xk + 1 = yk 🡪 y1 + y2 + … + yk = N + k (\*)

+) Bài toán trở thành đếm số nghiệm nguyên dương của phương trình (\*), áp dụng bài toán 1, ta có đáp số là:

* Áp dụng kết quả của bài toán 2, ta có số nghiệm nguyên của các phương trình (1), (2), …, (t) lần lượt là: , , …,
* Vì việc chọn các bộ số cho mỗi thừa số phải được thực hiện đầy đủ, độc lập, không phụ thuộc thứ tự, không phụ thuộc nhau nên đáp số bài toán chính là tích đáp số của t bài toán nêu trên

**Hướng giải (Tóm gọn):**

**1.** Phân tích M = p1x1 \* p2x2 \* … \* ptxt

**2.** Đáp số bài toán chính là:

**Vấn đề về cài đặt tổ hợp C(k, n):**

* Ta biết rằng, ab = c, với c không đổi, nếu a giảm thì b phải tăng và ngược lại

Mà M = 109, hỏi rằng số mũ to nhất trong phân tích thừa số nguyên tố của M là bao nhiêu. Số mũ to nhất, khi cơ số nhỏ nhất. Mà cơ số nhỏ nhất chính là 2

Ta giải phương trình 2x = 109 🡪 x = log2(109) = 29.89. Nên số mũ tối đa sẽ không đến 30, tức là x[i] ≤ 30, với mọi i

* N + x[i] – 1 = 105 + 30. Như vậy, không thể xử lý mảng C(k, n) 2D được
* Ta biết rằng = . Như đã phân tích ở trên, Nmax = 105, kmax = 30.
* Tiền xử lý N! mod (MOD = 109 + 7): Gọi fac[i] = i!.

Đương nhiên 0! = 1, N! = (N – 1)! \* N. 🡪fac[i] = (fac[i – 1] \* i) % MOD;

Như vậy là đã xong giai thừa trên tử số

* Xử lý giai thừa dưới mẫu số:

**Định lý Fermat nhỏ:**

* p là số nguyên tố, a không chia hết cho p thì ap – 1 ≡ 1 (mod p)
* Nhân cả 2 vế đồng dư thức với a-1, ta được: ap – 2 ≡ a-1 (mod p)

**Áp dụng:**

* Gọi ivFac[i] = (i!)-1 =
* Áp dụng định lý Fermat nhỏ, ta có: = (MAX!)MOD – 2. Mà (MAX!) theo modulo ta đã tính từ trước đó (fact[MAX]).

Vì số mũ rất lớn nên chỗ này sẽ sử dụng luỹ thừa nhị phân để tính

* Từ đó, ta tính giật ngược nghịch đảo giai thừa về trước theo tính chất phân phối phép nhân theo modulo:

🡪invFac[i] = [invFac[i + 1] \* (i + 1)] % MOD;

* Cuối cùng: = fac[n] \* ivFac[k] % MOD \* ivFac[n - k]

**BÀI D. ĐƯỜNG TRÒN**

* Cho đường tròn C tâm I(X, Y), bán kính R.
* Một điểm P(x, y) nằm trong C thì IP ≤ R 🡪(x – X)2 + (y – Y)2 ≤ R2
* Như vậy, các điểm x khả dĩ là: (x – X)2 ≤ R2 🡪 distX = |x – X| ≤ R

🡪-R ≤ x – X ≤ R 🡪X - R ≤ x ≤ X + R. Đây chính là khoảng mà ta sẽ lặp X

* Với mỗi X như vậy, ta có |y – Y| ≤ = distY

🡪-distY ≤ y - Y ≤ distY 🡪Y – distY ≤ y ≤ Y + distY.

* Cận trên luôn làm tròn lên, cận dưới luôn làm tròn xuống. Ta được Min, Max. Đáp án sẽ là: Max – Min + 1
* **Xử lý sai số: Đọc thêm bài viết:** [**https://www.facebook.com/share/p/17DGNmMpa2/**](https://www.facebook.com/share/p/17DGNmMpa2/)

**BÀI E. NỐI ĐIỂM**

* Về bản chất, bài toán này chính là tìm cây khung cực tiểu.
* ***Nhắc lại về thuật toán Kruskal cơ bản:***

1. Sắp xếp tâp cạnh E theo trọng số tăng dần. Khởi tạo MST\_w = 0, cnt = 0, với cnt là số cạnh cây khung
2. Duyệt từng cạnh (u, v, w) trong E. Nếu nối (u, v) không tạo thành chu trình:

* Nối (u, v) vào cây khung
* MST\_w += w, cnt++. Khi cnt = Số đỉnh - 1 thì dừng
* Các bạn đã học qua môn DSA ở PTIT chắc cũng biết đây là bài toán khó hơn của bài DSA10016 – Nối điểm. Bây giờ, ta sẽ cùng xét bài toán này
* Đối với bước kiểm tra chu trình, để cho thuận tiện thì ta sẽ sử dụng các hàm find và Union của CTDL DSU:

int find(int x){

if(x == par[x]) return x;

return par[x] = find(par[x]);

}

bool Union(int a, int b) {

a = find(a), b = find(b);

if (a == b) return false;

if (sz[a] < sz[b]) swap(a, b);

sz[a] += sz[b];

par[b] = a;

return true;

}

* Triển khai thuật toán:

1. Nhập toạ độ N điểm, khởi tạo DSU: par[i] = i, sz[i] = 1
2. Tạo tập cạnh khả thi:

* Dùng vector để lưu cạnh.
* Với mỗi cặp điểm (i, j) tính trọng số dist(i, j) và đưa vào vector.

1. Sắp xếp các cạnh trong v theo trọng số tăng dần
2. Triển khai thuật toán Kruskal trên tập cạnh v đã sắp xếp

* Đối với bài này, N lên đến 105, duyệt 2 vòng for như ver 1 thì lên đến 1010 vòng lặp, chắc chắn TLE.
* Phần kiểm tra chu trình, sử dụng DSU đã là tối ưu nhất, không cần tối ưu hơn.

Ta cần xử lý chỗ tạo cạnh, sao cho kéo giảm ĐPT dưới O(N2).

* Đề cho biết tung độ (y) chỉ nằm trong đoạn [0, 10].

⇒ Ta có thể gom các điểm theo tung độ.

⇒ Với mỗi tung độ Y, tập hợp các điểm chung Y lại rồi sắp xếp chúng theo X tăng dần

* Với một điểm X:
* Không cần nối X với tất cả N - 1điểm khác.
* Ta chỉ cần xét với các tung độ từ 0 đến 10.
* Trong mỗi tung độ, chỉ cần nối X với **2 điểm gần nhất** theo hoành độ (một bên trái, một bên phải).
* Để tìm nhanh 2 điểm gần nhất này:
* Sắp xếp mỗi tập các điểm chung tung độ theo hoành độ.
* Dùng **tìm kiếm nhị phân** để chọn láng giềng 2 bên cho điểm X
* Độ phức tạp chỉ còn O(NlogN)

**BÀI F. CHIA HẾT CHO 8**

Một số chia hết cho 8 khi và chỉ khi 3 chữ số cuối của nó chia hết cho 8. Do đó, ta chia thành 3 trường hợp chính:

**TH1:** Số có 1 chữ số: Bằng 8 thì chia hết, ngược lại thì không

**TH2:** Số có 2 chữ số: 1 trong 2 số là chính nó và số đảo của nó chia hết cho 8 là được

**TH3:** Số có 3 chữ số đổ lên:

* Chúng ta chỉ quan tâm 3 chữ số cuối.
* Giả sử số đó có dạng \*\*\*\*\*\*\*abc:
* 0 ≤ a, b, c ≤ 9 và a, b, c nguyên
* Phần phía trước không ảnh hưởng đến tính chất chia hết cho 8, ta không cần xét
* Sinh 3 vòng for cho các chữ số a, b và c. Mỗi số đều duyệt từ 0 đến 9.

Lúc này, giá trị của đuôi chính là V = 100a + 10b + c. Nếu V chia hết cho 8, đây là một bộ ba khả thi

* Kiểm tra xem chuỗi gốc có đủ số chữ số để tạo ra bộ ba abc này không:
* Duyệt tất cả các chữ số trong xâu gốc, lưu vào mảng cnt[10]. Với cnt[i] là số lượng chữ số i
* Duyệt 3 chữ số a, b, c để đếm số lượng số cần cho bộ 3 này, lưu vào needed.
* So sánh needed[d] với cnt[d] cho mọi chữ số d.

Nếu cnt[d] >= needed[d] với tất cả d, thì bộ ba này có thể tạo ra từ chuỗi.

Ngược lại, bộ ba không khả thi.

* Tổng số trường hợp cần xét: 3 vòng lặp cho a, b và c: 103, mỗi bộ ba thêm 10 lần so sánh. ĐPT chung là 104