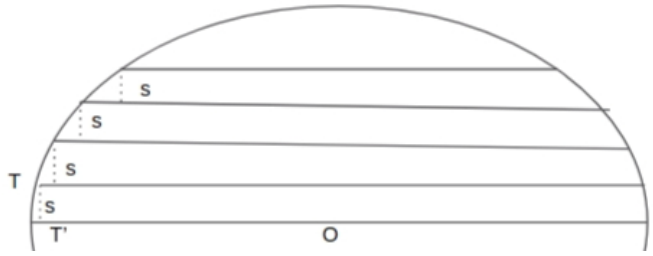
JCPC Kontes Mingguan 2

August 23, 2020

1 Gudang Kardus

Tag: Math
Perhatikan kita bisa membagi-bagi lingkaran menjadi potongan-potongan independen. Setiap potongan dapat diisi dengan kardus sebanyak mungkin dan tidak mempengaruhi potongan-potongan lain:



Anggap: O sebagai titik pusat lingkaran, T sebagai titik dimana garis berjarak S dan sejajar dengan diameter memotong lingkaran, T' sebagai proyeksi T pada diameter. Kita berusaha mencari panjang OT'. ini dapat dicari dengan rumus $\sqrt{(R^2 - S^2)}$ karena OT = R dan TT' = S. Setelah menemukan panjang OT', banyaknya kardus yang dapat dimasukkan pada lapisan ini adalah $\frac{2OT'}{S}$

Dengan logika yang sama, panjang daerah yang dapat diletakkan kardus pada lapisan ke-i adalah $\sqrt{R^2 - iS^2}$. Jadi, lakukan iterasi selama iS^2 masih lebih kecil dari R, lalu tambahkan semua jawabannya. Perhatikan untuk membulatkan kebawah ya.
Kompleksitas: O(R)

2 Tantangan Yayan

Tag: ad hoc

Jawabannya -1 jika dan hanya jika $\frac{N(N-1)}{2} < K$. Selain itu pasti ada jawaban.

Meminta permutasi lekiksografi terkecil adalah pentunjuk besar untuk meminimalkan elemen dari elemen paling pertama. Sebuah solusi dengan elemen terdepan i akan selalu lebih baik dari solusi yang elemen terdepannya i+1. Jadi optimalnya, elemen ke-i diisi dengan i. Elemen ke-i dapat diisi dengan i apabila jumlah inversi maksimal dari sisa array yang belum diisi maksimal, yakni $\frac{(n-i)(n-i-1}{2} \geq K$.

Bagaimana jika untuk suatu i, $\frac{(n-i)(n-i-1}{2} < K$? maka pasti ada j, j¿i, yang dimana jika kita meletakkan j pada posisi ke-i, jumlah inversi maksimalnya K. Buktinya cukup mudah jadi diserahkan sebagai latihan (hint: ingat algoritma sebelumnya). Setelah meletakkan j pada posisi i, elemen-elemen setelahnya pasti kebalikan dari elemen-elemen yang belum dipasang, karena jumlah maksimal inversi yang mungkin dicapai K sehingga optimalnya kita membentuk K tersebut disini.

Kompleksitas: O(N)

3 Barisan Menarik

Tag: Dynammic Programming
Misalkan DP[i][last] menyatakan banyaknya cara membariskan i elemen pertama yang valid dimana tinggi-tinggi anak maksimal i dan anak terakhir memiliki tinggi last. Saat transisi misalkan kita mau mencari nilai dp[i][x], ada 2 kasus bergantung pada syarat barisan:

Kasus 1: $A_{i-1} < A_i$**:**

1. Jika anak setinggi x tidak berada di belakang barisan: DP[i][x-1]
2. Jika anka setinggi x di depan barisan: tinggi anak sebelumnya harus kurang dari last. Kita bisa menambahkan semua elemen yang lebih tingginya tidak kurang dari x dibelakang barisan ini untuk menjaga agar syarat-syarat barisan tetap terjaga. Bukti singkat: ada untuk setiap pasang elemen bersebelahan ada 2 kemungkinan yaitu elemen terbesar yang naik, atau kedua elemen naik dan keduanya menyebabkan syarat tetap terjaga. Anak terakhir harus kurang dair last agar saat ditambahkan nilainya masih kurang dari last. DP[i-1][x-1]
3. sehingga DP[i][x] = DP[i][x-1] + DP[i-1][x-1]

Kasus 2: $A_{i-1} > A_i$

1. Jika anak setinggi x tidak berada di belakang barisan: DP[i][x+1]
2. Jika anak setinggi x di depan barisan: bisa menerapkan algoritma sama seperti pada kasus sebelumnya yaitu menaikkan semua elemen dibelakang permutasi yang nilainya tidak kurang dari x. Namun sekarangn nilai lastnya minimal x, karena jika dinaikkan akan menyebabkan terpenuhinya syarat. DP[i-1][x].
3. sehingga DP[i][x] = DP[i][x+1] + DP[i-1][x]

Didapat rumus DP:

1. DP[i][x] = DP[i][x-1] + DP[i-1][x-1] jika i-1 posisi menarik
2. DP[i][x] = DP[i][x+1] + DP[i-1][x] jika i-1 posisi tidak menarik

Basecase: DP[1][1] = 1. Jangan lupa untuk di modulo.
Kompleksitas: O(N^2

4 Hadiah Terindah

Tag: Set, Goldbach's conjecture

Ada 2 hal yang perlu dicari pada soal ini:

1. pembungkus minimal untuk membukus x buat permen 1 secara berurutan
2. cara menjaga agar sum tepat setiap update.

Untuk subtakss $1 \leq N \leq 2.000$, kamu bisa menghitung nilai ini dengan DP coin change karena banyaknya prima hanya 168. Untuk menghitung nilai ini pada constaint penuh kita dapat memanfaatkan Goldbach's conjecture dan Goldbach's weak conjecture:

1. Semua bilangan genap ¿ 2 dapat dinyatakan dengan penjumlahan dua buah bilangan prima
2. Semua bilangan ganjil ¿ 5 dapat dinyatakan dengan penjumlahan tiga buah bilangan prima

Tetapi dalam kasus ini, karena 1 dianggap prima, maka semua bilangan genap pasti dapat dinyatakan dengan penjumlahan dua bilangan, dan semua bilangan ganjil pasti bisa dinyatakan dalam penjumlahan tiga buah bilangan. Jadi, jumlah pembungkus minimalnya untuk x permen dapat dicari dengan membagi kasus:

1. 1, jika x prima
2. 2, jika x genap, atau x ganjil dan x-2 prima
3. 3, *otherwise*

Jadi dalam kasus ini dapat diselesaikan dengan precompute semua prima lalu menghitung pembungkus minimal dapat diselesaikan dalam O(1).

Berikutnya untuk mengupdate sum saat query, kamu bisa menyimpan sum dari query sebelumnya dan hanya mengupdate bagian yang sumnya berubah. Kamu bisa melakukan ini dengan menyimpan set berisi start dan end dari setiap segment 1 pada array. Jika sebuah operasi mengubah 0 menjadi 1, cari tahu posisi 1 terkiri dan terkanan yang bersebelahan lalu update kontribusi. Logika serupa jika mengubah 1 menjadi 0.
Kompleksitas: O(Q log N)

5 Kelereng

Tag: Combinatorics

Kita membutuhkan permutasi dan kombinasi untuk masalah ini. Anggap:

$$P(N, K) = \frac{N!}{(N-K)!}$$

.

$$C(N, K) = \frac{N!}{K!(N-K)!}$$

Ada 2 kasus. Kasus pertama jika $K \geq Q$, pada kasus ini jawabannya P(K, Q) Kasus kedua jika $K < Q$. Pada kasus ini pertama semua kelereng akan dimasukkan ke barisan, ada K! cara melakukan ini. Selanjutnya, kita perlu memilih (Q - K) kelereng pada kotak untuk diletakkan di barisan, ada C(N, Q - K) cara untuk melakukan ini. Untuk suatu pemilihan kelereng di tangan, kita dapat mengisinya sebagai berikut:

1. anggap kelereng dari kotak yang akan dimasukkan dinomori dari 1, 2, 3, ... , Q - K. Kelereng akan dimasukkan ke barisan dimulai dari kelereng ke-1 sampai Q-K.

2. Anggap saat isi isi barisannya ABCD, maka ada 5 tempat untuk memasukkannya, yaitu: _A_B_C_D_.

3. anggap kelereng memasukkan di slot kedua, Posisi menjadi AB1CD. Jumlah slot yang mungkin untuk dimasukkan naik 1 menjadi 6.

4. Lakukan ini untuk semua kelereng, sehingga banyaknya cara memasukkannya adalah S = (K+1)(K+2)(K+3)...(Q) = P(Q, Q-K).

Jadi, untuk kasus kedua ada K! C(N, Q - K) P(Q, Q-K) banyaknya cara.

Untuk mengihitung X! dan $\frac{1}{X!}$ secara cepat, kamu bisa precompute di awal dengan array. Anggap F[i] sebagai i! mod 100007 dan inv[i] sebagai 1/i! mod 100007. Rumusnya:

$$F[i] = iF[i-1] mod 100007$$

$$inv[i] = i^{10005} inv[i-1] mod 100007$$

Rumus inv didapat dari fermat's last theorem.
Kompleksitas: O(N log N) untuk precompute, O(1) untuk menjawab query