Blueprint for Seating

Problem author: Artem Vasilyev; problem developer: Niyaz Nigmatullin

Consider any group not located next to a window. If the group consists of t seats, then the inconvenience part of the group is $s(t_1) + s(t_2)$, where t_1 and t_2 are chosen optimally such that $t_1 + t_2 = t$, and $s(x) = \frac{x(x-1)}{2}$.

For a group of t seats next to a window, the inconvenience part is $s(t)$.

So the inconvenience is the sum of some $s(r_1) + s(r_2) + \ldots + s(r_{2k})$. Each r_i is either one of the two sizes of groups next to a window, or one of the $2 \cdot (k-1)$ subgroups created by division of middle groups.

Consider we've chosen r, $a = \max r_i$ and $b = \min r_i$ such that $a - b \geq 2$, then $s(a-1)+s(b+1) < s(a)+s(b)$, because

$$\begin{aligned} 2 \cdot (s(a-1) + s(b+1)) &= (a-1) \cdot (a-2) + (b+1) \cdot b \\ &= (a-1) \cdot a - 2 \cdot (a-1) + (b-1) \cdot b + 2b \\ &= 2 \cdot (s(a) + s(b)) + 2 \cdot (b - a + 1) \\ &< 2 \cdot (s(a) + s(b)) \end{aligned}$$

That's because $(b - a + 1)$ is negative.

So the only way to divide is almost equally. The multiset of r: $n \bmod (2k)$ times $\lceil \frac{n}{2k} \rceil$ and $2k - (n \bmod (2k))$ times $\lfloor \frac{n}{2k} \rfloor$.

We can iterate over four ways to choose the window groups, not considering, when the window group is of size of 0. Then we need to calculate how to arrange middle groups.

We have $(k-1)$ middle groups, and x subgroups of size $\lfloor \frac{n}{2k} \rfloor = t$ and y subgroups of size $t+1$.

Each group can be of size either $2t$, or $2t+1$, or $2t+2$. If we iterate with number of groups of size $2t+1$ say g, then $\frac{x-g}{2} = f$ groups will be of size $2t$ and $\frac{y-g}{2} = h$ groups of size $2t+2$. The number of arrangements with the fixed g is the product of some binomial coefficients, for example $\binom{f+g+h}{g} \cdot \binom{f+h}{f}$. We should only consider cases, when f and h are non-negative integers. And we should handle the case when $2t$ equals to zero, it means f needs to be zero as well.

The binomial coefficient can either be recalculated when iterating g, by several multiplications and divisions. Another way to do it is to pre-calculate all factorials and their inverse modulo the given prime number, and use them to calculate binomial coefficient in $\mathcal{O}(1)$. For a linear algorithm to pre-calculate all inverse values modulo prime p you can use the following recurrent formula:

$$x^{-1} \equiv -(p \bmod x)^{-1} \cdot \left\lfloor \frac{p}{x} \right\rfloor \pmod{p}$$

note that $(p \bmod x)^{-1}$ can be calculated before x^{-1}. The formula comes from relation $p \bmod x = p - \lfloor \frac{p}{x} \rfloor \cdot x$.

The total time complexity for a single test case could be $\mathcal{O}(k)$ or $\mathcal{O}(k \log k)$, depending on how you handle the binomial coefficients.