Problem A. Optimal Point

Setter: Fahim Tajwar Saikat(steinum)

We can apply **welzl's algorithm**(https://en.wikipedia.org/wiki/Smallest-circle_problem). For this, we need to solve some sub-problems, i.e. find the center of the smallest hypersphere for given k $(1 \leq k \leq 5)$ points on its boundary.

- Case 1: $k = 1$ (Single Point)

  If there is only one point p_1, the smallest hypersphere that encloses p_1 is centered at p_1 itself, with radius 0.

- Case 2: $k = 2$ (Two Points)

  If there are two points, p_1 and p_2, the center of the hypersphere is the midpoint of the line segment joining p_1 and p_2, and the radius is half the distance between the two points.

- Case 3: $k = 3$ (Three Points)

  If there are three points, p_1, p_2, and p_3, the center of the hypersphere is equivalent to finding the circumcenter of a triangle formed by these three points.

- Case 5: $k = 5$ (Five Points)

  When $k = 5$, the hypersphere in 4D space can be uniquely determined by the five points $p_1 = (x_1, y_1, z_1, w_1)$, $p_2 = (x_2, y_2, z_2, w_2)$, $p_3 = (x_3, y_3, z_3, w_3)$, $p_4 = (x_4, y_4, z_4, w_4)$, and $p_5 = (x_5, y_5, z_5, w_5)$.

  To find the center $c = (c_x, c_y, c_z, c_w)$ of the hypersphere, we solve a system of equations derived from the property that c is equidistant from all five points.

  The condition $\|c - p_i\| = \|c - p_j\|$ for any i, j translates to:

  $\|c - p_1\|^2 = \|c - p_2\|^2, \|c - p_1\|^2 = \|c - p_3\|^2, \ldots, \|c - p_1\|^2 = \|c - p_5\|^2$.

  Expanding and simplifying these equations yields:

  $2((x_i - x_1)c_x + (y_i - y_1)c_y + (z_i - z_1)c_z + (w_i - w_1)c_w) = x_i^2 + y_i^2 + z_i^2 + w_i^2 - x_1^2 - y_1^2 - z_1^2 - w_1^2, \forall i \in \{2, 3, 4, 5\}$.

  This gives a system of four linear equations in the four unknowns c_x, c_y, c_z, c_w. We can solve this using Gaussian Elimination, and the radius will be the distance from point c to point p_1

- Case 4: $k = 4$ (Four Points)

  When $k = 4$, the hypersphere in 4D space can be determined by the four points $p_1 = (x_1, y_1, z_1, w_1)$, $p_2 = (x_2, y_2, z_2, w_2)$, $p_3 = (x_3, y_3, z_3, w_3)$, and $p_4 = (x_4, y_4, z_4, w_4)$. We can approach this case like case 5, but we will a system of three linear equations in the four unknowns c_x, c_y, c_z, c_w.

  We can use Gaussian Elimination, but for $c_x, c_y, c_z, \text{and } c_w$ we will have generic formula e.g. $c_x = f_1 + g_1 \times x,\ c_y = f_2 + g_3 \times x,\ c_z = f_3 + g_3 \times x, \text{and } c_w = f_4 + g_4 \times x$

  Now, we need to minimize the value of the radius(or the squared value of the radius), $r^2 = ||c - p_1||^2$.

  By using differentiation, we can find the value of x for which r^2 is minimum. Thus, finding the value of x we can get the center(c) and radius r.

Problem B. The Fortune Dice

Setter: Fahim Tajwar Saikat(steinum)

To determine if the sum of two dice rolls, $a + b$, equals x, note that the possible sums range from 2 (when $a = 1, b = 1$) to 12 (when $a = 6, b = 6$). Simply check if $2 \leq x \leq 12$; if true, the answer is "Yes", otherwise "No".

Problem C. Expected Final Score

Setter: MD. All Shahoriar Tonmoy(AST_ TheCoder)

Let $f(n, p)$ represent the expected final value of p after all deletions when the set has n elements. We define $f(n, p)$ below:

$$f(n,p) = \begin{cases} 0 & \text{if n = 0 (no elements left)} \\ -n & \text{if p = 0 (no possible shift)} \\ p & \text{if } n \leq p \text{ (no elements to delete below } p) \\ \frac{n-p}{n} \cdot f(n-1, p-1) + \frac{p}{n} \cdot f(n-1, p) & \text{otherwise} \end{cases}$$

Here:

- $\frac{n-p}{n}$ represents the probability of deleting an element below p, and
- $\frac{p}{n}$ represents the probability of deleting an element above p.

Using DP techniques, you can find the value for $f(n, p)$.

Problem D. Maximum AND

Setter: Rudro Debnath(RD_ TheCoder)

For any $k = n$, you can't do any operation, answer $= (a_1 \, \& a_2 \, \& \, \ldots \, \& a_n)$.

Otherwise, do the following operations in order:

- Choose $i = 1$ and j in range $[k+1, n]$ i.e. $(k+1 \leq j \leq n)$. Hence, $a_1 = a_1 | (a_{k+1} | a_{k+2} | \ldots | a_n)$.
- Choose $i = n$ and j in range $[1, n-k]$ i.e. $(1 \leq j \leq n-k)$. Hence, $a_n = a_n | (a_1 | a_2 | \ldots | a_{n-k})$.
- Choose $i = 1, j = n$ and $i = n, j = 1$, Hence, $a_1 = (a_1 | a_n)$ and $a_n = (a_n | a_1)$

Thus we can make a group of OR using these indices: $[1, n-k] \cup [k+1, n]$.

- If $n - k \geq k + 1$, then answer $= (a_1 \, | a_2 \, | \, \ldots \, | a_n)$.
- Otherwise, answer $= (a_1 \, | a_2 \, | \, \ldots \, | a_{n-k}) \& (a_{n-k+1} \, \& a_{n-k+2} \, \& \, \ldots \, \& a_k) \& (a_{k+1} \, | a_{k+2} \, | \, \ldots \, | a_n)$

So all you need to calculate “Prefix Or”, “Suffix OR”, and some “Mid-Segment And” for each k.

Problem E. Cyclic Inversion

Setter: Saiful Islam Ramim(_r4m1m)

Consider the array(0-indexed) cyclic and i-th cycle of the array is $[a_i,\ a_{i+1},\ \ldots,\ a_{n-1},\ a_0,\ a_1,\ \ldots,\ a_{i-1}]$. For each cycle of the array, calculate a value I_i, i.e. the inversion count for the i-th cycle.

For each k $(1 \le k \le n-1)$, the answer will be $\min\limits_{i \ge 0} I_{\gcd(k,\,n)\cdot i}$

Problem F. Make Permutation

Setter: Jakir Hossen Shagor(purple_ghost)

Model this problem as a bipartite graph:

- Left side: indices $1, 2, \ldots, n$.
- Right side: values $1, 2, \ldots, n$.
- Add an edge from i(left) to j(right) if j can be formed from a_i by unsetting atmost a set-bit.

If the **maximum matching**(`https://cp-algorithms.com/graph/kuhn_maximum_bipartite_matching.html`) on this graph is n, then the answer is “Yes”, “No” otherwise.

Problem G. User Registration System

Setter: Fahim Tajwar Saikat(steinum)

For a given `username` let’s split it into two parts:

- `base username`
  - this will be a prefix of the `username`
- `digits`
  - A numeric suffix of the `username` that starts with a non-zero digit.
  - The length of this numeric string will be at most 5(it can also be 0).

Now, we can maintain a DS(set), where insert/remove/find MEX(minimum positive integer not contained in the set) for each `base username`.

We can handle both query in the following way:

- **Add**: For a given `username`, we can find MEX(let’s say x) on the corresponding DS. The User Registration System will respond “`username` + x” (Here, ‘+’ means concatenation). Now while inserting “`username` + x” in the system, for each `base username` of “`username` + x” we will insert the “`digits`” part in the corresponding DS of the `base username`.

- **Delete**: While deleting "`username`" in the system, for each `base username` of "`username`" we will delete the "`digits`" part in the corresponding DS of the `base username` if exists. If there is at least one such "`digits`" part that exists in the DS, then the User Registration System will respond `OK`, otherwise `INVALID`.

Problem H. Optimizing Weekend Days

Setter: Rakibul Ranak(RakibulRanak)

For each date in between `starting date` and `ending date`, if it is not a holiday, count it as a `working date`. For each day ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday') count the number of `working date`s are there on that day. You need to print two days, with the lowest of that count.