Kick Start 2021 - Round C

Analysis: Alien Generator

Test Set 1

We can check for every $i, (1 \leq i \leq \mathbf{G})$ whether there exists a k such that

$$\sum_{j=0}^{k}(i+j) = \sum_{j=0}^{k} i + \sum_{j=0}^{k} j = ((k+1) \times i) + \frac{k \times (k+1)}{2} = \mathbf{G}$$

Finding such a k (if one exists) can be done by binary searching the range $[0, \mathbf{G}]$, and hence takes $O(\log \mathbf{G})$ time. For a candidate k in that range, we check if $((k+1) \times i) + \frac{k \times (k+1)}{2} = \mathbf{G}$ and alter the range based on the equality. Time complexity here is $O(\mathbf{G} \log \mathbf{G})$.

Alternative solution

For this Test Set, we can implement a brute force solution. We iterate over every $i, (1 \leq i \leq \mathbf{G})$ and try to sum up numbers $[i, i+1, i+2, \ldots]$ until the sum exceeds or equals $\mathbf{G}$. If the sum equals $\mathbf{G}$, then we increment our result by one. Here, each iteration takes $O(\mathbf{G}/i)$ time.

$$\sum_{i=1}^{\mathbf{G}}(1/i) = O(log(\mathbf{G}))$$

Therefore, the overall time complexity of this solution is $O(\mathbf{G} \log \mathbf{G})$.

Test Set 2

Since the upper bound on $\mathbf{G}$ is 10^{12}, $O(\mathbf{G} \log \mathbf{G})$ solution times out. Let us define $H = \lceil \sqrt{2 \times \mathbf{G}} \rceil$. An observation can be made that $k \leq H$. Therefore, for each k in the range $[0, H]$, we can binary search for i in the range $[1, \mathbf{G}]$ thereby making the total runtime $O(\sqrt{\mathbf{G}} \times \log(\mathbf{G}))$.
This solution might not pass within the time limit for slow languages. Therefore, we will look at a better solution next.

We can rewrite the equation we saw in Test Set 1 as $i = \frac{2 \times \mathbf{G} - k^2 - k}{2 \times (k+1)}$. Next, for each k in the range $[0, H]$, we can check in $O(1)$ whether we can obtain a positive integer value for i that satisfies the above equation. The runtime here is $O(\sqrt{\mathbf{G}})$.

Alternative solution

We can dig deeper into the relationship between $\mathbf{G}$, K, and d, the number of days it takes for the machine to produce exactly $\mathbf{G}$ gold starting at K on day one. They form the equation

$$\frac{d(K + (K + d - 1))}{2} = \mathbf{G}$$

which is equivalent to $d(d + (2K - 1)) = 2\mathbf{G}$. Since one of d and $(d + (2K - 1))$ is even and the other is odd, any pair of positive integers x and y such that exactly one of them is even and

$x \times y = 2\mathbf{G}$ can be mapped to them with the smaller of the two being d and the larger one $(d + (2K - 1))$, which is always greater than d. Since each mapping produces a different d, each pair corresponds to a unique solution for d and K. Conversely, every pair of d and K that satisfies the equation corresponds to a different x, y pair.

To count the number of such pairs, let g be the largest odd factor of $2\mathbf{G}$. Note that any (ordered) pair x', y' such that $x' \times y' = g$ corresponds to a pair $x = \frac{2\mathbf{G}}{g}x'$ and $y = y'$. Finally, assume the prime factorization of g is

$$g = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n}$$

the number of such ordered pairs is $(\alpha_1 + 1)(\alpha_2 + 1) \cdots (\alpha_n + 1)$. We can thus prime factorize $\mathbf{G}$, ignore the 2, and multiply all other prime powers accordingly. Prime factorization can be trivially implemented in $O(\sqrt{\mathbf{G}})$ complexity and there are $o(\log(\mathbf{G}))$ prime factors. Therefore the total time complexity is $O(\sqrt{\mathbf{G}})$.