Cooked Fish (Hard Version) Editorial

Note this is for the original problem and not for the case of minimizing a in the answer. rip my checker....

This problem is spilt into three main cases. $k \geq 3$, $k = 2$, and $k = 1$.

$k \geq 3$

Let's try solving it for $k = 3$. We can see that once we take a value greater than 10^6 or $10^{18\frac{1}{3}}$, 10^{6^k} will exceed n. Then if we define $arr = \{1, 2^3, 3^3, \cdots, 10^{6^3}\}$, then the problem is reduced to finding a subarray with a sum of n. Then we can run an $O(n^{\frac{1}{3}})$ two pointers brute force to find the answer. For all $k \geq 3$, we can solve it in $O(n^{\frac{1}{k}})$ time.

$k = 1$

If n is not a power of 2 , there is a solution. Let's prove it. So for a valid solution (a, b), lets define $k = b - a$ (like the easy version). Again, k here does not refer to the k in the problemstatement but rather $b - a$. So we have

$$n = k \cdot a + \frac{k \cdot (k+1)}{2}$$

and

$$2n = 2 \cdot k \cdot a + k \cdot (k+1) = k \cdot (2a + k + 1)$$

So we know that $2n$ has an odd and an even factor since the parity of k and $(2a + k + 1)$ are different. So if n is a power of 2, then there are not two such factors. Also if we can find any two valid factors, they make a solution since we can use it solve for a and b. So now if have some n which is not a power of two, we can trivially find an even factor, lets call it y as the largest power of 2 that divides $2n$. Then we define $z = (2n)/y$. So we know that $|z - y|$ is $2 * a + 1$ and that $\min(y, z)$ is k. Then we can solve for a and k and output $(a, a + k)$ as our answer. Complexity $O(1$ or $O(\log n)$ depending on the implementation.

$k = 2$

Let's use our idea for the easy version here. So for a valid answer (a, b) we have $n = \sum_{i=a}^{b-1} i^2$. If we define the sum of integers from $[0, i]$ as $sum(i)$ and the sum of squares from $[0, i]$ as $sumsq(i)$, we can start simplifying

$$n = \sum_{i=a}^{b-1} i^2 = \sum_{i=0}^{b-a-1} (a+i)^2 = \sum_{i=0}^{b-a-1} a^2 + 2ai + i^2 = (b-a) \cdot a^2 + \sum_{i=0}^{b-a-1} 2ai + i^2 = (b-a) \cdot a^2 + 2 \cdot a \cdot sum(b-a-1) + sumsq(b-a-1$$

Which again is a sum that can make a only relative to n and $b - a$. For the $(b - a) \cdot a^2$ term, i just took it out of the sigma, for the $2 \cdot a \cdot sum(b - a - 1)$ i used distributive property on the sum of integers from $[0, b - a - 1]$, and i just wrote $sumsq(b - a - 1)$ since that's what the sum of all the i^2 was.

Now believe it or not, we can make a quadratic such that when give n and $b - a$, we can solve for a.

$$(b-a) \cdot a^2 + (2 \cdot sum(b-a-1)) \cdot a + (sumsq(b-a-1) - n) = 0$$

Now we can plug it into the quadratic formula and use $(b - a)$ and n to solve for a.

$$a = \frac{-(2 \cdot sum(b-a-1) \pm \sqrt{(2 \cdot sum(b-a-1)^2 - (4)(b-a)(sumsq(b-a-1) - n)))}}{2 \cdot (b-a)}$$

Now when $sumsq(b-a-1)-n>0$, the square root of the discriminator will simplify to be less than $2 \cdot sum(b-a-1)$ and the entire equation will be negative. Since $sumsq$ grows cubicly, we only have to check around $n^{\frac{1}{3}}$ values of $b-a$. But this is not all, evaluating $sqrt$ takes $O(\log n)$ time. So we have a complexity of $O(n^{\frac{1}{3}} \log n)$ which happens to be too slow.

I have no proof of the following heurstic, but it somehow speeds up the code by around 20 times. So now as we loop over $b-a$, evaluating $sqrt$ is the most time consuming part. But you can notice that if the value in the $sqrt$ is not divisible by i^2, there is no chance that it'll simply to a being an integer. So the optimization is to simply not eval $sqrt$ and continuing if the discrim for a certain $b-a$ is not divislbe by i^2. The end complexity is at most $O(n^{\frac{1}{3}} \log n)$, but I can't prove what it actually is.

apology

I really did not think that the grader was wrong for this :<. My model solution is ac but some code that only handles the case when $k=1$ is also ac. In fact, harry submitted code that

```
cout << "-1\n"
```

and its ac. sorry and i hope it was still a good problem :<.