The last foobar

Nedeljko Stefanović

Problem 1 *Write program for computing*

$$f(n) = \sum_{i=1}^{n} \lfloor i\sqrt{2} \rfloor.$$

for $n \leqslant 10^{1000}$.

Solution: We use the fact that $\sqrt{2}$ is irrational real, such that sometimes from non-strict inequality we can conclude strict inequality. Define we functions

$$g(n) = \lfloor \tfrac{n}{1+\sqrt{2}} \rfloor, \quad h(n) = \sum_{i=1}^{n} g(i).$$

Holds

$$f(n) = \sum_{i=1}^{n} \lfloor i + i(\sqrt{2}-1) \rfloor = \sum_{i=1}^{n} (i + g(i)) = \frac{n(n+1)}{2} + h(n).$$

For $m \in \mathbb{N}$ holds

$$\begin{aligned} g(i) = m \quad &\Leftrightarrow \quad m(1+\sqrt{2}) < i < (m+1)(1+\sqrt{2}) \\ &\Leftrightarrow \quad 2m + m(\sqrt{2}-1) < i < 2m + 2 + (\sqrt{2}-1)(m+1) \\ &\Leftrightarrow \quad 2m + m/(1+\sqrt{2}) < i < 2m + 2 + (m+1)/(1+\sqrt{2}) \\ &\Leftrightarrow \quad 2m + 1 + g(m) \leqslant i \leqslant 2m + 2 + g(m+1). \end{aligned}$$

There are $2 + g(m+1) - g(m)$ such numbers i. However, for $m = k$ we have to care that $i \leqslant n$, or

$$2k + 1 + g(k) \leqslant i \leqslant n,$$

and there are $n - 2k - g(k)$ such numbers. For

$$k = g(n)$$

holds

$$\begin{aligned} h(n) &= \sum_{m=1}^{k-1} m(2 + g(m+1) - g(m)) + k(n - 2k - g(k)) \\ &= (k-1)k + \sum_{m=1}^{k-1} ((m+1)g(m+1) - mg(m)) - \sum_{m=1}^{k-1} g(m+1) + k(n - 2k - g(k)) \\ &= (k-1)k + k\,g(k) - h(k) + k(n - 2k - g(k)) \\ &= k(n-k-1) - h(k). \end{aligned}$$

Because $k < n/(1+\sqrt{2})$ we have logarithmically many recursive calls in depth.

Of course,

$$g(1) = h(1) = 0.$$

The value $g(n)$ can be computed by computing value $\frac{n}{1+\sqrt{2}}$ in interval arithmetics using greater and greater precision until interval without integers is obtained as result, so that integral part of both interval endpoints is the same.

Another way is discretization of the Newton's tangent method. Holds

$$x = \frac{n}{1+\sqrt{2}} \Leftrightarrow n - x = \sqrt{2}x \Leftrightarrow x^2 + 2nx - n^2 = 0.$$

We can compute value $\frac{n}{1+\sqrt{2}}$ bu Newton's tangent method. Holds

$$x = \frac{n}{1+\sqrt{2}} \Leftrightarrow \sqrt{2}x = n - x \Leftrightarrow 2x^2 = n^2 - 2nx + x^2 \Leftrightarrow x^2 + 2nx - n^2 = 0.$$

Because function $x^2 + 2nx - n^2$ is increasing and convex, holds

$$\frac{n}{1+\sqrt{2}} < x \Rightarrow \frac{n}{1+\sqrt{2}} < \frac{x^2+n^2}{2(x+n)} < x.$$

By this and $n > \frac{n}{1+\sqrt{2}}$ the sequence

$$x_0 = n, \quad x_{k+1} = x_k - \frac{x_k^2 + 2nx_k - n^2}{2(x_k+n)} = \frac{x_k^2+n^2}{2(x_k+n)}$$

is decreasing and convergent to $\frac{n}{1+\sqrt{2}}$. For $x \in \mathbb{N}$ is

$$\left\lfloor \frac{x^2+n^2-1}{2(x+n)} \right\rfloor + 1 = \left\lfloor \frac{x^2+n^2+2(x+n)-1}{2(x+n)} \right\rfloor \geqslant \frac{x^2+n^2}{2(x+n)},$$

and holds

$$\frac{n}{1+\sqrt{2}} < x \Rightarrow \frac{n}{1+\sqrt{2}} < \left\lfloor \frac{x^2+n^2-1}{2(x+n)} \right\rfloor + 1.$$

Define we the sequence

$$x_0 = n, \quad x_{k+1} = \left\lfloor \frac{x_k^2+n^2-1}{2(x_k+n)} \right\rfloor + 1.$$

x_n is non-increasing sequence of integers that are greater than $\frac{n}{1+\sqrt{2}}$. Establish we by what condition is $x_{k+1} = x_k$.

$$\begin{aligned}
x_{k+1} \geqslant x_k \quad &\Leftrightarrow \quad \left\lfloor \frac{x_k^2+n^2-1}{2(x_k+n)} \right\rfloor \geqslant x_k - 1 \\
&\Leftrightarrow \quad \frac{x_k^2+n^2-1}{2(x_k+n)} \geqslant x_k - 1 \\
&\Leftrightarrow \quad (x_k+n-1)^2 \leqslant 2n^2 \\
&\Leftrightarrow \quad x_k - 1 \leqslant \frac{n}{1+\sqrt{2}}.
\end{aligned}$$

Because $x_k \in \mathbb{Z}$ $x_k - 1 \leqslant \frac{n}{1+\sqrt{2}} < x_k$, holds

$$\left\lfloor \frac{n}{1+\sqrt{2}} \right\rfloor = x_k - 1, \quad \text{for } k \text{ such that } x_{k+1} = x_k.$$

□

This can be coded in Python [1] as

```
def g(n):
        if n<3:
                return 0
        x = n
        n2 = n*n
        while True:
                xn = (x*x+n2-1)//(2*(x+n))+1
                if xn==x:
                        return x-1
                x = xn

def h(n):
        if n<3:
                return 0
        k = g(n)
        return k*(n-k-1)-h(k)

def f(n):
        return n*(n+1)//2+h(n)
```

It is easy to eliminate recursion from h.

```
def h(n):
        s = 0
        positive = True
        while n>=3:
                k = g(n)
                if positive:
                        s += k*(n-k-1)
                else:
                        s -= k*(n-k-1)
                positive = not positive
                n = k
        return s
```

Finally, functions f and h we can replace by single function.

```
#
def f(n):
        s = n*(n+1)//2
        positive = True
        while n>=3:
                k = g(n)
                if positive:
                        s += k*(n-k-1)
                else:
                        s -= k*(n-k-1)
```

[1]Python supports arbitrary integers.

```
            positive = not positive
            n = k
        return s
```

Finally, the complete code is

```
def g(n):
        if n<3:
                return 0
        x = n
        n2 = n*n
        while True:
                xn = (x*x+n2-1)//(2*(x+n))+1
                if xn==x:
                        return x-1
                x = xn

def f(n):
        s = n*(n+1)//2
        positive = True
        while n>=3:
                k = g(n)
                if positive:
                        s += k*(n-k-1)
                else:
                        s -= k*(n-k-1)
                positive = not positive
                n = k
        return s
```