Mini Hackathon Entry

Maiyun Zhang

November 16, 2023

Attempted Questions 1 to 5.

Question 1

- Zero skip: one possible path with nine jumps.
- One skip: eight possible paths with eight jumps.
- Two skips: seven jumps in total and two of them are skip jumps, so $\binom{7}{2} = 21$.
- Three skips: six jumps in total and three of them are skip jumps, so $\binom{6}{3} = 20$.

Total: $1 + 8 + 21 + 20 = 50$ possible paths.

Question 2

The question explicitly mentions that the rabbit is "her neighbor's rabbit" and is dead, but "the neighbor" says that the rabbit "not harmed" and "doing well." Therefore, I postulate that the neighbor's pet is and has always been a dead rabbit, presumably a specimen. The neighbor is probably a necromancer or a taxidermist.
Otherwise, if the narrative might be unreliable, it may be possible that the neighbor is polite.

Question 3

Projected distance from the top of the cylinder: $\sqrt{9^2 - 8^2} = \sqrt{17}$.
Angle difference between them: $2 \arcsin \frac{\sqrt{17}}{2\times5}$.
Projected arc length: $5 \times 2 \arcsin \frac{\sqrt{17}}{10}$.
Vertical distance: $8 + 2 = 10$.
Shortest path: $\sqrt{10^2 + (10 \arcsin \frac{\sqrt{17}}{10})^2} = 10\sqrt{1 + \arcsin^2 \frac{\sqrt{17}}{10}}$.

Question 4

A rather naïve approach in Python (sad face):

1. Start with the biggest square.
2. Keep track of the margin between the space and the squares.
3. If the next biggest square is smaller than the margin, then fit it there and update the margin accordingly.
4. If the next biggest square is bigger than the margin, then fit it in the corner and update the margin accordingly.
5. Repeat until all squares are fitted.

```
def fitsquares(sides: list[int]) -> int:
    sides = sorted(sides, reverse=True)
    resultnow = 0
    margin = 0
    for nextbiggest in sides:
        if margin < nextbiggest:
            # New margin: assume the next biggest is codiagonal with resultnow
            margin = 2 * resultnow
            resultnow += nextbiggest
        else:
            margin -= nextbiggest
    return resultnow
```

Some test cases:

```
>>> fitsquares([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
4
>>> fitsquares([3, 4, 1])
7
>>> fitsquares([3, 1, 1, 1])
4
```

Question 5

Numbers that are even and divisible by 5 are multiples of 10. I am using the closed form of the Fibonacci sequence. Wolfram Languange:

```
Total@Table[
  If[Mod[FullSimplify[
    (-(1/2 (1 - Sqrt[5]))^n + (1/2 (1 + Sqrt[5]))^n)/Sqrt[5]],
    10] == 0, 1, 0], {n, 1, 49}]
```

Result is 3.
C Code:

```
#include <math.h>
#include <stdio.h>
#include <inttypes.h>

static inline double ipow(double base, int exp) {
    double result = 1.0;
    while (1) {
        if (exp & 1) result *= base;
        exp >>= 1;
        if (!exp) break;
        base *= base;
    }
    return result;
}

uint32_t count(void) {
    uint32_t count = 0;
    // const double half1andsqrt5 = (1.0 + sqrt(5.0)) / 2.0;
    const double half1andsqrt5 = 1.6180340051651000976562 5;
    // const double half1minusqrt5 = (1.0 - sqrt(5.0)) / 2.0;
    const double half1minusqrt5 = -0.6180340051651000976562 5;
    const double sqrt5 = 2.2360680103302001953125;

    for (uint32_t i = 1; i < UINT32_MAX; ++i) {
        double fib = (ipow(half1andsqrt5, i) - ipow(half1minusqrt5, i)) / sqrt5;
        if (fib > UINT32_MAX) break;
        count += ((uint32_t)fib % 10 == 0);
    }
    return count;
}

int main(void) {
    printf("%" PRIu32 "\n", count());
    return 0;
}
```

Result is 3, it is really fast although we are pushing the limit of `double` precision. The last fibonnaci number that it gave was actually incorrect.

Question 6