Homework 1, CSE 232

Due September 3

Problem 1

During lab today, I gave you a Small problem from the Google Code Jam: Bullseye. Here's the text of the problem:

> Maria starts with t millilitres of black paint, which she will use to draw rings of thickness 1 cm (one centimetre). A ring of thickness 1 cm is the space between two concentric circles whose radii differ by 1 cm.
>
> Maria draws the first black ring around a white circle of radius r cm. Then she repeats the following process for as long as she has enough paint to do so:
>
> Maria imagines a white ring of thickness 1 cm around the last black ring. Then she draws a new black ring of thickness 1 cm around that white ring. Note that each "white ring" is simply the space between two black rings. The area of a disk with radius 1 cm is π cm^2. One millilitre of paint is required to cover area π cm^2. What is the maximum number of black rings that Maria can draw? Please note that:
>
> Maria only draws complete rings. If the remaining paint is not enough to draw a complete black ring, she stops painting immediately. There will always be enough paint to draw at least one black ring.

In the Small problem, both r and t are less than 1000.

> *a. How much paint will it take to paint the nth black ring in terms of n and r?*

A ring with an inner radius r has total area $\pi(r+1)^2 - \pi r^2$ - the area of the outer circle minus that of the inner circle. It takes 1 unit of paint to cover an area of π, so the total paint needed for the first ring (with inner radius r) is $(r+1)^2 - r^2$ or $2r+1$. The next ring has an inner radius that is 2 cm greater than the first ring - 1 cm for the first black ring and 1 cm for the white space which separates the two. Thus, the paint required for it is $2(r+2)+1$, or $2r+5$. Similarly, the third ring requires $2r+9$ units of paint and the formula is $P(r,n) = 2r + 4n - 3$.

> *b. Describe a brute force algorithm to solving the Small problem by iterating through each black ring using the formula derived in part a. How does this algorithm scale with maximum n? What is the largest possible number of black rings Maria can draw under using the Small input limits?*

We can solve the Small problem by starting n at 1 and adding up the paint required for each successive ring as we increment n. Once the total paint required is more than t, we've run out of paint on the current ring, so the last one was the most rings possible to paint. This algorithm is linear with maximum n - we need to iterate through each ring to do the calculation.

The most number of rings it would be possible to paint would be with large t and small r. So with the extreme values of these variables, $r = 1$ and $t = 1000$, we can simply solve the problem:

```
In [16]: r = 1
         t = 1000
         n = 0
         paint = 0
         while paint <= t:
             n += 1
             paint += 2 * r + 4 * n - 3
         print n - 1
```

In the Large problem, r is allowed to be as large as 10^{18}, while t is allowed to be as large as 2×10^{18}.

c. How does this limit the applicability of a brute force solution?

While we can try to use the same approach as above, unfortunately when we run it with the new maximum limits, our code takes far too long to run and we have to kill it. Clearly these limits are too large for brute force.

```
In [23]: r = 1
         t = 2e18
         n = 0
         paint = 0
         while paint <= t:
             n += 1
             paint += 2 * r + 4 * n - 3
         print n - 1
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)

<ipython-input-23-8edd99c33df8> in <module>()
  5 while paint <= t:
  6     n += 1
----> 7     paint += 2 * r + 4 * n - 3
  8 print n - 1


KeyboardInterrupt:
```

d. How much paint is needed to paint the first n black rings in terms of n and r? That is, $\sum_{i=1}^{n} P(i,r)$, where $P(n,r)$ is the paint needed for the nth ring as calculated in part b.

$P(n,r) = 2r + 4n - 3$, so we want to calculate

$T(n,r) = \sum_{i=1}^{n} 2r + 4i - 3.$

The first thing to note is that we can pull most of the terms out of the sum:

$T(n,r) = 2rn - 3n + 4\sum_{i=1}^{n} i.$

The remaining sum is a well known series and is the triangular number $n(n+1)/2$, giving us the final formula

$T(n,r) = n(2n + 2r - 1).$

e. Describe how you could potentially use this formula to solve this problem in $O(1)$ time. Implement this solution in your language of choice, attach your solution to your homework, and try to solve both the Small and Large problems with it using the Google Code Jam submission form. Does it work for both input sizes?

We can solve the quadratic equation $t = 2n^2 + 2rn - n$ for n and take the floor. If t paints exactly an integer number of rings, we'll get an integer solution - otherwise we discard the excess as we won't be able to complete that ring. The solution to this equation is

$n = \left\lfloor \sqrt{4r^2 - 4r + 8t - 1} - (2r - 1)/4 \right\rfloor.$

Unfortunately, while this works for the Small problem, it fails for the Large problem because of rounding errors!

f. Working with large integers and square root functions can often cause integer overflows and rounding errors, leading to wrong answers even when the algorithm is technically correct. Describe how to solve this problem in $O(\log n)$ time using the same formula by repeatedly dividing the search space. Implement this solution, attach it to your homework, and test it on both the Small and Large problems.

We can solve this problem by doing a binary search for n using the formula we calculated for $T(n, r)$. We first need lower and upper bounds for n. The lower bound is guaranteed by the problem statement to be 1, and we can find the upper bound by repeatedly doubling n until $T(n, r) > t$. We then use binary search to repeatedly divide this interval in half until we have an interval where $T(n, r) <= t$ and $T(n + 1, r) > t$, giving us n as the solution.