Code Jam 2010 - Round 1C

Analysis: Load Testing

Understanding the sample cases

In order to solve this problem, one first needs to get a feeling of what's asked, and looking at the sample cases is a good way to achieve that.

Take a look at the first sample case. The answer given is 2. How do we achieve the result with just 2 loadtests? Let's make a couple guesses. Suppose we do a loadtest that checks if the site supports 100 users. If we learn that the site can't support 100 users, then we're done: we know we can support 50 users but can't support 100 which is 50*2. However, if we learn that the site can actually support 100 users, we have a very difficult task at hand now: we have only one loadtest left, and we know that the site can support 100 users but can't support 700. Is it possible to solve it?

Suppose we now loadtest to check if the site supports 300 users. If we learn that the site can't support 300 users, then we've failed to solve the problem: we know we can support 100, but can't support 300 - but 300 is more than 100*2, so we don't have enough knowledge. Moreover, this actually helps us to prove that our loadtest must test for 200 users or less, otherwise we will hit the same issue.

Now we know that our second loadtest must use at most 200 users. But even when it's exactly 200, suppose we learn that our site can actually support all of them. Then we've failed to solve the problem again: we know we can support 200, but can't support 700 - but 700 is more than 200*2.

So there's no good choice for our second loadtest. It means that the choice of the first loadtest was wrong - 100 users is too small for it.

What have we learned?

However, we've learned an important lesson in our failed attempt to understand the example case: when we have only one loadtest left, and we know that the site can support **L** people but can't support **P** people, we must loadtest with such number **X** that **L***C>=**X**, and at the same time **X***C>=**P**. The first inequality will help us solve the problem when the loadtest fails, and the second one is helpful if the loadtest succeeds.

Since there's no such number **X** for **L**=100, **P**=700, **C**=2, our first attempt above has failed.

The question now is: how to check if such **X** exists? From the first equation, we get **X**<=**L***C**. From the second one, we get **X**>=**P**/**C**. Such **X** exists if and only if **L***C**>=**P**/**C**, which means $L*C^2>=P$. Forgetting the formulas, the upper bound of our range should be at most C^2 times more than the lower bound. In that case, we can just take **X**=**L***C** for our only loadtest.

The second attempt at understanding the first sample case

Equipped with this knowledge, we get back to the first sample case. 100 was wrong since $100*2^2=100*4<700$. Maybe we should loadtest for 300 people first? If the loadtest succeeds, then we will have one loadtest left, 300 people OK, 700 people not OK, and since 300*4>=700, we can solve the problem. However, what if the loadtest doesn't succeed? We know that our system can support 50 people but can't support 300 people and have only one loadtest left. Since 50*4<300, we can't do that. So the choice of 300 was also wrong.

What if we try 200 as the first loadtest? In case it succeeds, we get one loadtest, 200 OK, 700 not OK, 200*4>=700 - we can do that. In case it fails, we get 50 OK, 200 not OK, 50*4>=200 - we can do that as well. So we've finally figured out the algorithm to solve the first sample case using just 2 loadtests:

```
Loadtest for 200 people. If the site can support 200 people:
  Loadtest for 400 people.
If the site can't support 200 people:
  Loadtest for 100 people.
```

What have we learned?

So how do we figure out if two loadtests are enough? This is actually surprisingly similar to the study of the one loadtest case.

When we have two loadtests left, and we know that the site can support **L** people but can't support **P** people, we must loadtest with such number **X** that $\mathbf{L}*\mathbf{C}^2 >= \mathbf{X}$, and at the same time $\mathbf{X}*\mathbf{C}^2 >= \mathbf{P}$. The first inequality will help us solve the problem using one remaining loadtest when the loadtest fails, and the second one is helpful if the loadtest succeeds.

Using the same argument as above, one can see that such number **X** exists if and only if $\mathbf{L}*\mathbf{C}^4 >= \mathbf{P}$ (we got $\mathbf{C}^4$ as $\mathbf{C}^2*\mathbf{C}^2$).

More loadtests?

Now it's not so hard to figure out what happens with more than two loadtests. It's possible to solve the problem using three loadtests if and only if $\mathbf{L}*\mathbf{C}^8 >= \mathbf{P}$. For four loadtests, we get $\mathbf{L}*\mathbf{C}^{16} >= \mathbf{P}$. And so on. That pretty much describes the solution for this problem.

Understanding the sample cases, attempt 3

Now we can finally figure out the algorithm to solve the third sample case: **L**=1, **P**=1000, **C**=2. In order to do this in four loadtests, our first loadtest can be for $\mathbf{L}*\mathbf{C}^8=256$:

```
Loadtest for 256 people. If the site can support them:
  Loadtest for 512 people.
If we can't support 256 people:
  Loadtest for 16 people. If the site can support them:
    Loadtest for 64 people. If the site can support them:
      Loadtest for 128 people.
    If we can't support 64 people:
      Loadtest for 32 people.
  If we can't support 16 people:
    Loadtest for 4 people. If the site can support them:
      Loadtest for 8 people.
    If we can't support 4 people:
      Loadtest for 2 people.
```

This looks quite similar to the binary search algorithm, but performed on exponential scale.

Conclusion

We started solving this problem by trying to understand the answers for the sample cases, and by the time we actually understood them, we already have a complete solution. The only remaining thing is to implement the solution carefully avoiding the integer overflow issues.