

Hot and Cold Binary Search Game

Ask Question

Hot or cold.

I think you have to do some sort of binary search but I'm not sure how.

Your goal is the guess a secret integer between 1 and N. You repeatedly guess integers between 1 and N. After each guess you learn if it equals the secret integer (and the game stops); otherwise (starting with the second guess), you learn if the guess is hotter (closer to) or colder (farther from) the secret number than your previous guess. Design an algorithm that finds the secret number in $\lg N + O(1)$ guesses.


Hint: Design an algorithm that solves the problem in $\lg N + O(1)$ guesses assuming you are permitted to guess integers in the range $-N$ to $2N$.

I've been racking my brain and I can't seem to come up with a $\lg N + O(1)$.

I found this: http://www.ocf.berkeley.edu/~www/cgi-bin/yabb/YaBB.cgi?board=riddles_cs;action=display;num=1316188034 but could not understand the diagram and it did not describe other possible cases.


algorithm binary-search

edited Aug 28 '14 at 22:24

 [Oliver Charlesworth](#)

217k 25 438 576

asked Aug 28 '14 at 22:23

 [awesome_dude3933](#)

11 2

What is your algorithm? A simple binary search would achieve this... – [Memming](#) Aug 28 '14 at 22:30

2 See stackoverflow.com/q/15678789/427252 ?? – [Memming](#) Aug 28 '14 at 22:31

1 I'd think base 2. – [awesome_dude3933](#) Aug 28 '14 at 23:17

2 Because if the base isn't specified, any multiplicative constant can be absorbed into the base... – [Oliver Charlesworth](#) Aug 28 '14 at 23:18

2 A slightly more explicit version of the hint is that, if you know that the number is in the range $[a, b]$, and you make consecutive guesses less than a and greater than b in some order, then hot/cold tells you whether the number is over some threshold. – [David Eisenstat](#) Aug 28 '14 at 23:48

5 Answers

For numbers between $(1 \sim N)$:

1st guess: $(1/3)N$


2nd guess: $(2/3)N \rightarrow$ tells us the answer is either in $1 \sim (1/2)N$ or $(1/2)N + 1 \sim N$

3rd guess: in $1 \sim (1/2)N$: try $(1/6)N$; in $(1/2)N + 1 \sim N$: try $(5/6)N$

...

Starting from 2nd guess, each step cuts the range in $1/2$. So it's step $1 + \lg N$

answered Oct 7 '15 at 19:11

 [W. L.](#)

31 1

Suppose you know that your secret integer is in $[a, b]$, and that your last guess is c .

You want to divide your interval by two, and to know whether your secret integer lies in between $[a, m]$ or $[m, b]$, with $m = (a+b)/2$.

The trick is to guess d , such that $(c+d)/2 = (a+b)/2$.

Without loss of generality, we can suppose that d is bigger than c . Then, if d is hotter than c , your secret integer will be bigger than $(c+d)/2 = (a+b)/2 = m$, and so your secret integer will lie in $[m, b]$. If d is cooler than c , your secret integer will belong to $[a, m]$.

You need to be able to guess between $-N$ and $2N$ because you can't guarantee that c and d as defined above will always be $[a, b]$. Your two first guess can be 1 and N .

So, your are dividing your interval be two at each guess, so the complexity is $\log(N) + O(1)$.

A short example to illustrate this (results chosen randomly):

Guess	Result	Interval of the secret number
1	***	$[1, N]$ // $d = a + b - c$
N	cooler	$[1, N/2]$ // $N = 1 + N - 1$
$N/2$	cooler	$[N/4, N/2]$ // $-N/2 = 1 + N/2 - N$
$5N/4$	hotter	$[3N/8, N/2]$ // $5N/4 = N/4 + N/2 + N/2$
$-3N/8$	hotter	$[3N/8, 7N/16]$ // $-3N/8 = 3N/8 + N/2 - 5N/4$
.	.	.
.	.	.
.	.	.

Edit, suggested by @tmyklebu:

We still need to prove that our guess will always fall in between $[-N, 2N]$

By recurrence, suppose that c (our previous guess) is in $[a-(a+b), b+(a+b)] = [-b, a+2b]$

Then $d = a+b-c \leq a+b-(-b) \leq a+2b$ and $d = a+b-c \geq a+b-(a+2b) \geq -b$

Initial case: $a=1, b=N, c=1$, c is indeed in $[-b, a+2b]$

QED

edited Aug 29 '14 at 12:25

answered Aug 29 '14 at 12:05



R2B2
766 6 14

You glossed over how to handle the case where your desired guess falls outside $[1, N]$. This is the key to the problem. – tmyklebu Aug 29 '14 at 12:09

Your desired guess have to be in between $[-N, 2N]$. You just need a simple recurrence to prove it, but you are right, I'll edit my post. – R2B2 Aug 29 '14 at 12:17

That's a relaxation of the problem. The "end case" where you're trying to guess a number in $[1, K]$ allowed only guesses in $[1, N]$ also matters. – tmyklebu Aug 29 '14 at 12:29

@tmyklebu Not according to awesome_dude3933 specifications. However, I would be keen to see a solution in $\lg N$ with this additional constraint. – R2B2 Aug 29 '14 at 12:50

$\lg N$ is unattainable. I gave a solution that takes $\lg N + 6$ guesses in the worst case in my answer below. – tmyklebu Aug 29 '14 at 12:52

This was a task at IOI 2010, for which I sat on the Host Scientific Committee. (We asked for an optimal solution instead of simply $\lg N + O(1)$, and what follows is not quite optimal.)

Not swinging outside $-N \dots 2N$ and using $\lg N + 2$ guesses is straightforward; all you need to do is show that the obvious translation of binary search works.

Once you have something that doesn't swing outside $-N \dots 2N$ and takes $\lg N + 2$ guesses, do this:

Guess $N/2$, then $N/2+1$. This tells you which half of the array the answer is in. Then guess the end of that half-array. You're either in one of the two "middle" quarters or you're in one of the two "end" quarters. If you're in a middle quarter, do the thing before and you win in $\lg N + 4$ guesses. The ends are slightly trickier.

Suppose I need to guess a number in $1 \dots K$ without straying outside $1 \dots N$ and my last guess was 1 . If I guess $K/2$ and I'm colder, then I next guess 1 ; I spent two guesses to get a similar subproblem that's $1/4$ the size. If $K/2$ is hotter, I know the answer is in $K/4 \dots K$. Guess $K/2-1$ next. The two subcases are $K/4 \dots K/2-1$ and $K/2 \dots K$, both of which are nice. But it took me three guesses to (in the worst-case) halve the size of the problem; if I ever do this, I wind up doing $\lg N + 6$ guesses.

edited Mar 22 '17 at 17:34



Cyclotron3x3
1,255 10 24

answered Aug 29 '14 at 12:24



tmyklebu
11.7k 3 19 48

But it took me three guesses to (in the worst-case) halve the size of the problem In this case, your complexity is $3 \lg(N) + O(1)$, since you need to do this each time you divide your problem (which will be the case if you have to guess 2 or $N-1$ for example) – R2B2 Aug 29 '14 at 12:45

@user3923424: No, I need to do that at most once, since before I'm in the "end" case and after I'm in the "middle" case, where I never have to do that again. This does not change the coefficient of $\lg N$. – tmyklebu Aug 29 '14 at 12:49

What about guessing number 2? $K/2$ will always be colder than 1, except in the very last steps. I don't see at which point you will get in the "middle" case... (sorry to bother, I would like to understand) – R2B2 Aug 29 '14 at 12:55

@user3923424: The "middle-case" recursion is properly handled by your answer. The "end-case" recursion uses two guesses to either quarter the size of the end or get a "middle-case" $3/4$ the size of the end. $K/2$ will not always be colder than 1 since the answer could lie in $[K/4, K]$; it's only colder than 1 if we're staying in the end-case recursion. – tmyklebu Aug 29 '14 at 12:59

Ok... I see how it's working now: you are still in the "end" case, but guessing 1 allows you to reduce your interval by 2 as well. Therefore, except for the extra initial guess (1) you are dividing your interval by 2 each time. With your words: you either divide by 4 the interval, or get a "middle" case using 2 guesses. Cheers then. – R2B2 Aug 29 '14 at 13:02

The solution is close to binary search. At each step you have an interval that the number can be in. Start with the whole interval $[1, N]$. First guess both ends - that is the numbers 1 and N . One of them will be closer, thus you will know that now the number you are searching for is in $[1, N/2]$ or in $[N/2 + 1, N]$ (considering N even for simplicity). Now you go to the next step having a twice smaller interval. Continue using the same approach. Keep in mind that you've already probed one of the ends, however it may not be your last guess.

I am not sure what you mean by $\lg N + O(1)$, but the approach I suggest will perform $O(\log(N))$ operations and in the worst case it will do exactly $\log_4(N)$ probes.

answered Aug 29 '14 at 7:59

 **Ivaylo Strandjev**
54.1k 10 79 137

- $\lg N$ is the base-two logarithm of N . A function is $O(1)$ if it is bounded above by a constant. – tmyklebu Aug 29 '14 at 12:10

Hmm at least in the math I have studied $\lg(N)$ is base 10 logarithm, [not a base 2 logarithm](#) I know what is $O(1)$ very well, but as base 10 has nothing to do with the problem I have written: am not sure what you mean by $\lg N + O(1)$ – Ivaylo Strandjev Aug 29 '14 at 12:36

That's odd. $\lg N$ means base-two logarithm everywhere I've seen. – tmyklebu Aug 29 '14 at 12:48

@tmyklebu nope both ISO 31-11 and the superseding [ISO 80000-2](#) state otherwise. – Ivaylo Strandjev Aug 29 '14 at 12:51

1 @pjs please note that here the log is **not** in a big O. Only the remainder is in big O notation. I assume this means that the number of operations is exactly $\lg(N)$ plus a constant. – Ivaylo Strandjev Aug 29 '14 at 13:03

Here're my two cents for this problem, since I got obsessed with it for two days. I'm not going to say anything new to what others have already said, but I'm going to explain it in a way that might get some people to understand the solution easily(or at least that was the way I managed to understand it).

Drawing from the $\sim 2 \lg N$ solution, if I knew that solution existed in $[a, b]$ I'd want to know if it's in the left half $[a, (a + b) / 2]$ or the right half $[(a + b) / 2, b]$, with the point $(a + b) / 2$ separating the two halves. So what do I do? I guess a then b ; if I get colder with b I know I'm in the first(left) half, if I get hotter I know I'm in the second(right) one. So guessing a and b is the way to know the secret integer position with respect to the mid point $(a + b) / 2$. However a and b aren't the only points that I can guess at to know the secret position. $(a - 1, b + 1)$, $(a - 2, b + 2)$, ... etc are all valid pairs of points to guess at to know the secret position, as the mid point of all these pairs is $(a + b) / 2$, the mid point of the original interval $[a, b]$. In fact any two numbers c and d such that $(c + d) / 2 = (a + b) / 2$ can be used.

So considering $[a, b]$ as the interval we know the secret integer exists within, take c to be the last number we guessed. We want to determine the position of the secret with respect to the mid point $(a + b) / 2$, so we a new number d to guess at to know the secret relative position to $(a + b) / 2$. How do we know such number d ? By solving the equation $(c + d) / 2 = (a + b) / 2$, which yields $d = a + b - c$. Guessing at that d , we shrink the range $[a, b]$ appropriately based on the answer(colder or hotter) and then repeat the process taking d as our last guess and trying a new guess at number e for example with the same conditions.

To establish the initial conditions, we should start with $a = 1$, $b = N$, and $c = 1$. We guess at c to establish a reference(since the first guess can't tell you anything useful as there were no prior guesses). We then proceed with new guesses and adjusting the enclosing interval as appropriate with each guess. The table in [@R2B2's answer](#) explains it all.

You have to be vigilant however when trying to code this solution. When I tried to code it in [python](#), I first ran into the mistake of getting $[a, b]$ stuck when it was small enough(like $[a, a + 1]$) where neither a nor b would move inwards. I had to phase the cases where the interval size was 2 outside the loop and handle them separately(like I did with intervals with size 1 also).

edited May 23 '17 at 12:00

 Community ♦
1 1

answered Sep 27 '14 at 21:10

 **Mohammed Safwat**
71 1 3