LETSUPGRADE DATA STRUCTURES AND ALGORITHMS

DAY 4 ASSIGNMENT

Q1) In the Binary Search algorithm, it is suggested to calculate the mid as

beg + (end - beg) / 2 instead of (beg + end) / 2. Why is it so?

Because beg + end may overflow. Which then means you get a result that is less than beg. Or far into the negative if you are using signed integers.

So instead they take the distance between beg and end and add half of that to beg. This is only a single extra operation to make the algorithm more robust.

Q2) Write the algorithm/function for Ternary Search.

Let *f*(*x*) be a [unimodal](https://en.wikipedia.org/wiki/Unimodal) function on some interval [*l*; *r*]. Take any two points *m*1 and *m*2 in this segment: *l* < *m*1 < *m*2 < *r*. Then there are three possibilities:

* if *f*(*m*1) < *f*(*m*2), then the required maximum can not be located on the left side - [*l*; *m*1]. It means that the maximum further makes sense to look only in the interval [*m*1;*r*]
* if *f*(*m*1) > *f*(*m*2), that the situation is similar to the previous, up to symmetry. Now, the required maximum can not be in the right side - [*m*2; *r*], so go to the segment [*l*; *m*2]
* if *f*(*m*1) = f(*m*2), then the search should be conducted in [*m*1; *m*2], but this case can be attributed to any of the previous two (in order to simplify the code). Sooner or later the length of the segment will be a little less than a predetermined constant, and the process can be stopped.

choice points *m*1 and *m*2:

* *m*1 = *l* + (*r*-*l*)/3
* *m*2 = *r* - (*r*-*l*)/3

**Run time order**

{\displaystyle T(n)=T(2n/3)+1=\Theta (\log n)}

**Recursive algorithm**

**def** ternary\_search(f, left, right, absolute\_precision) -> float:

*"""Left and right are the current bounds;*

*the maximum is between them.*

*"""*

**if** abs(right - left) < absolute\_precision:

**return** (left + right) / 2

left\_third = (2\*left + right) / 3

right\_third = (left + 2\*right) / 3

**if** f(left\_third) < f(right\_third):

**return** ternary\_search(f, left\_third, right, absolute\_precision)

**else**:

**return** ternary\_search(f, left, right\_third, absolute\_precision)

**Iterative algorithm**

**def** ternary\_search(f, left, right, absolute\_precision) -> float:

*"""Find maximum of unimodal function f() within [left, right]*

*To find the minimum, reverse the if/else statement or reverse the comparison.*

*"""*

**while** abs(right - left) >= absolute\_precision:

left\_third = left + (right - left) / 3

right\_third = right - (right - left) / 3

**if** f(left\_third) < f(right\_third):

left = left\_third

**else**:

right = right\_third

*# Left and right are the current bounds; the maximum is between them*

**return** (left + right) / 2