DATA STRUCTURES AND ALOGRITHM.

SARAVANAKARUPPU. K

BATCH ONE

QUESTION ONE:

In general with binary search, the reason is to avoid overflow. Beg+end is subject to overflow with large values. Using end-beg avoids the overflow.

Imagine of beg was MAX\_INT-3 and end was MAX\_INT-1, then beg+end would be larger than MAX\_INT, but end-beg, would just be 2.

With iterators, this also works out because end-begin is a number, whereas begin+end is not valid. You can subtract two iterators to get the distance between them, but you can’t add two iterators.

Adding two iterators doesn’t make sense, and you can’t do that. You can call operator- on two iterators, and gives a reasonable result, i.e. the count of elements between two iterators. And you can add or subtract an integral number on iterator, means move it forward or backward.

Beg variable is only at the beginning of the vector the first time. After that, you’re moving either the near end or the far end to the old mid-point so as to reduce the search space. This method is called a binary search because it reduces the search space by ½ each iteration (it’s a O(logn) algorithm)

QUESTION 2:

Function:

Assume we are looking for a maximum of f(x) and that we know the maximum lies somewhere between A and B. For the algorithm to be applicable, there must be some value x such that

For all a,b with A ≤ a < b ≤ x, we have f(a) < f(b), and

For all a,b with x ≤ a < b ≤ B, we have f(a) > f(b).

Algorithm:

Let f(x) be a unimodal function on some interval [l; r]. Take any two points m1 and m2 in this segment: l < m1 < m2 < r. Then there are three possibilities:

If f(m1) < f(m2), then the required maximum can not be located on the left side – [l; m1]. It means that the maximum further makes sense to look only in the interval [m1;r]

If f(m1) > f(m2), that the situation is similar to the previous, up to symmetry. Now, the required maximum can not be in the right side – [m2; r], so go to the segment [l; m2]

If f(m1) = f(m2), then the search should be conducted in [m1; m2], 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 m1 and m2:

M1 = l + (r-l)/3

M2 = r – (r-l)/3

Run time order

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