Double Median

Another month has passed and it is time to reveal how the Codility challenge codenamed Nu can be solved. You can still give it a try, but no certificate will be awarded.

In the Nu certificate problem, two arrays, A and B, of integers sorted in ascending order, are given. Additionally, a sequence of queries is given in the following form: given two continuous segments, one in the first array and another in the second, find a median of numbers in both segments. The result that should be computed is a median of answers to the given queries.

The main question is: how quickly can a median of a sequence of numbers be computed? For an arbitrary sequence of N numbers, the median can trivially be computed in $O(N \log N)$ time. Simply, one can sort the sequence and pick the middle element. (Note that all the medians in this problem are taken from odd numbers of elements.)

More sophisticated algorithms, e.g. the algorithm of the five(s), can find the median in $O(N)$ time. (The name of the algorithm relates to two sources: one is that the given sequence is divided into groups of five elements; another is that the algorithm has five authors.) However, these algorithms don't make use of the fact that the input sequences are sorted. Using them, one can solve the problem in $O(K(N+M))$ time, where K is the number of queries and N and M are the lengths of the given sorted sequences.

We will show how to find the median of the union of two sorted sequences in a logarithmic time. It will allow us to answer all the queries in $O(K \log(N+M))$ time. Then, the result can be calculated even using sorting, which yields $O(K \log(K+N+M))$ total time complexity. Let us assume that `median(p, q, r, s)` is a function returning a median of $A[p..q]$ and $B[r..s]$, where A and B are the given sorted arrays. A solution coded in Python might be as follows:

```
1  def double_median(A,B,P,Q,R,S):
2      n = len(A)
3      m = len(B)
4      k = len(P)
5      medx = [0]*k
6      for i in xrange(k):
7          medx[i] = median(P[i], Q[i], R[i], S[i])
8      medx.sort()
9      return medx[k // 2]
```

We still have to implement the function `median`. Let us define $x = q-p+1$, $y = s-r+1$, $z = (|x-y|-1)/2$; note that $|x-y|$ is odd. First, we will simplify the problem and reduce it to a case where $x = y \pm 1$. We can either drop the top z elements and bottom z elements from the larger of the ranges, $p..q$ and $r..s$, or instantly point to the result. If `med(p, q,`

`r, s)` is a function returning a median of $A[p..q]$ and $B[r..s]$ for $q - p = r - s \pm 1$, then the implementation of function `median` can be as follows:

```
def median(p, q, r, s):
    x = q-p+1
    y = s-r+1
    z = abs(x-y) // 2
    if x > y:
        if A[p+z] >= B[s]:
            return A[p+z]
        elif A[q-z] <= B[r]:
            return A[q-z]
        else:
            return med(p+z, q-z, r, s)
    else:
        if B[r+z] >= A[q]:
            return B[r+z]
        elif B[s-z] <= A[p]:
            return B[s-z]
        else:
            return med(p, q, r+z, s-z)
```

Now, we have to implement function `med`. We will do it using bisection. The key observation is that if $A[p+d] \leqslant B[s-d]$, then we can drop the bottom d elements and top d elements by narrowing to the ranges $p+d..q$ and $r..s-d$. Similarly, if $A[q-d] \geqslant B[r+d]$, then we can narrow to the ranges $p..q-d$ and $r+d..s$. Moreover, if $p+d \leqslant q-d$ and $r+d \leqslant s-d$, then $A[p+d] \leqslant A[q-d]$ and $B[r+d] \leqslant B[s-d]$. Hence:

- either $A[p+d] \leqslant B[s-d]$ and we can narrow to the ranges $p+d..q$ and $r..s-d$;
- or $A[q-d] \geqslant A[p+d] > B[s-d] \geqslant B[r+d]$ and we can narrow to the ranges $p..q-d$ and $r+d..s$.

For $x + y > 5$ we can use bisection, taking $d = \lfloor (x+y)/4 \rfloor$. For smaller values of $x + y$, we can simply sort and pick the middle element.

```
def med(p, q, r, s):
    x = q-p+1
    y = s-r+1
    if x + y < 5:
        medi = A[p:q+1]+B[r:s+1]
        medi.sort()
        return medi[(x+y)//2]
    else:
        d = (x+y) // 4
        if A[p+d] < B[s-d]:
            return med(p+d, q, r, s-d)
        else:
            return med(p, q-d, r+d, s)
```
