(a): Xingan Liu

For every time, we first divide the set into two parts, and the no. of objects in each part is n_1, n_2, we should let $n_1 = n_2$ or $|n_1 - n_2| = 1$, and we test one part, if its positive, we repeat the process on this part, otherwise, we do it on another part, for example:

Step 1: First time, a set of n objects, we divide it into two parts, the first part contains $\lfloor \frac{n}{2} \rfloor$ objects, and the rest part contains $\lceil \frac{n}{2} \rceil$ objects

Step 2: Then, we test on the first part, if it's positive, we divide this part into two parts (similar to step 1), and repeat testing and dividing, until only 2 or 3 objects remains

if the first part is not positive, we know that special object is not in this part, it's definitely in other parts, so we turn to the other parts do splitting and testing until only 2 or 3 objects remains

Step3: At the end, when there are only two objects in group (which means each group has one object, we test one, if it's positive, then it's the special one, otherwise the other object is the special one.

When there is only one objects in group, we can assure that this one is positive (which means its previous group has 3 objects, when dividing, one group has two objects and the other only has one, in this way, if the group with two objects is not positive, we can conclude that the one object in other group is positive

and the number of test of n objects is

① when n is a power of 2

no. test $= \log_2 n$

② for other cases

for the worst case, no. test $= \lceil \log_2 n \rceil$, which means when grouping,

the positive objects hasn't be an one-object group. for other case, the no. of test $= \lfloor \log_2 n \rfloor$

so for worst case, the number of test is $\lceil \log_2 n \rceil$

(b)

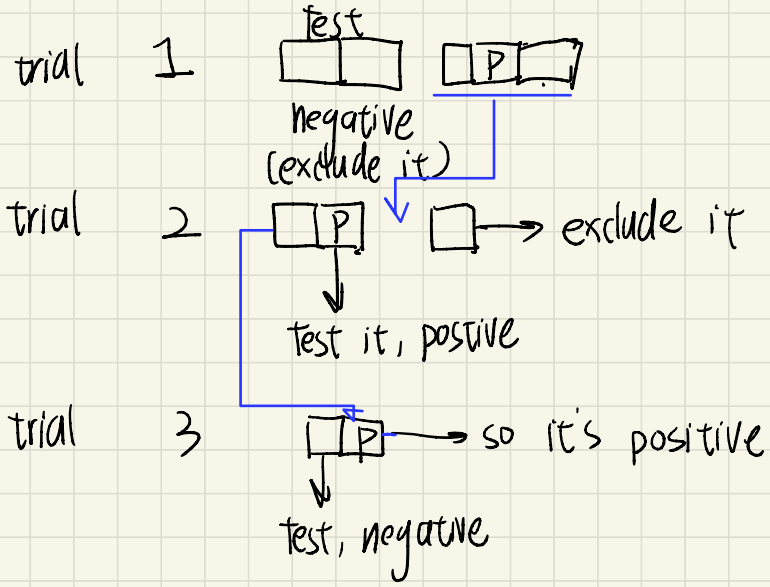
Yes

when we divide the n objects into 2 almostly equally subgroup, every step we can figure out $\lfloor \frac{n}{2} \rfloor$ to $\lceil \frac{n}{2} \rceil$ objects that are not positive, then we can exclude it,

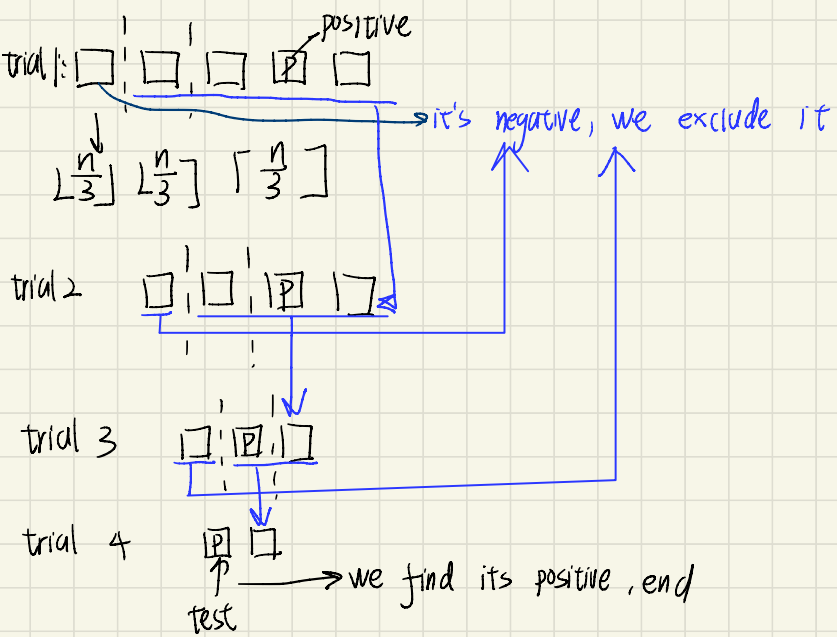
but if we divide it into 3 or other almost equally group, or even divide it randomly. Although in the best case we can find the positive one quicker the method proposed in (a), but for the worst case, the no. of testing $> \lceil \log_2 n \rceil$, because for the worse case, every step, the no. of objects provided to be non-positive $< \lfloor \frac{n}{2} \rfloor$

for example, we have 5 objects

if used method proposed in (a), and the second last object is positive



if used other method, such as divide it into 3 parts,
each part contains $\lfloor \frac{n}{3} \rfloor$, $\lfloor \frac{n}{3} \rfloor$
$n - 2 \times \lfloor \frac{n}{3} \rfloor$ objects



Compare with 2 methods, the method proposed in (a) has less test, because in the worse time, we can exclude at most $\lfloor \frac{n}{2} \rfloor$ negative objects, but for the second method, we can only exclude one negative objects, which mostly smaller than $\lfloor \frac{n}{2} \rfloor$, therefore, it needs more times to exclude the negative one and find remaining positive one.

Therefore, method proposed in (a) is optimal.

(c) it is true

as for n objects $n \rightarrow T_k(n)$

$T_k(n)$ is the time for finding the positive one among n objects

if we want to find the special one in n objects, after dividing into 2 parts, we will test whether first j objects are positive (and this will consume one test)

Then after testing, based on our results from first j objects, we may divide first j objects or the remaining objects into 2

parts and find the positive one among j or n-j objects, and the time is $T_k(n-j)$ or $T_k(j)$, for the worst case, time will be the larger one of $T_k(n-j)$ and $T_k(j)$

because j is variable, we want to find which j will make the time for worse cause quicker, so we need to iterative all j from 1 to k, figuring out the smallest time $T_k(n)$ and its corresponding j

so

$$T_k(n) = \min_{1 \le j \le k} \{1 + \max\{T_k(j), T_k(n-j)\}\}$$

$$= 1 + \min_{1 \le j \le k} \{\max[T_k(j), T_k(n-j)]\}$$

(d)

we can first assume, when $i \le j$, $T_k(i) \ge T_k(j)$.

if we add some item in i, which let $i' = j$, in this way $T_k(i)$ should be reduced to $T_k(j)$, which means $T_k(i') = T_k(j)$. But actually $i \le i'$, find the elements among i objects should not be slower than among i' subjects, otherwise $T_k(n)$ won't be the optimal solution.

Therefore, when $i \le j$, $T_k(i) \ge T_k(j)$ is false. In the contrary, T_k should be a montone function

(e)

if $j = k$ (proposed strategy)

$$T_k(n) = 1 + \max\{T_k(k), T_k(n-k)\}$$

From (c), we know the optimal number of test can be described as

$$T_k(n) = 1 + \min_{1 \le m \le k} \max\{T_k(m), T_k(n-m)\}$$

because $T_k(n)$ is monotone

$$T_k(k) \ge T_k(m)$$

$$T_k(n-k) \le T_k(n-m) \quad \text{as } k \ge m \rightarrow n-k \le n-m$$

if $n-k \ge k \Rightarrow n \ge 2k$, which means we will still use the method in (c)

as $n-k \ge k \Rightarrow T_k(n-k) \ge T_k(k)$

so $T_k(n) = 1 + T_k(n-k)$ (if $j=k$, our proposed strategy)

because K is the largest value among all m (as $1 \le m \le k$)

if $n \ge 2k$, $n \ge 2k \ge 2m$, so $n \ge 2m$

so $n-m \ge m \Rightarrow T_k(n-m) \ge T_k(m)$

so actually for optimal function in (c)

$$T_k(n) = 1 + \min_{1 \le m \le k} T_k(n-m)$$

because $T_k(n)$ is monotone, so when $m = k$, $T_k(n)$ reaches its minimal, where $T_k(n) = 1 + T_k(n-k)$

and this is just the same as the no. of test we used by our proposed methods, all use $1 + T_k(n-k)$ times

but if $n \le 2k$, we need to use divide and conquer

$$T_k(n) = 1 + \min_{1 \le m \le k} \max \{ T_k(m), T_k(n-m) \}$$

As we have proved in (a) and (b), $T_k(n)$ should be $\lceil \log_2 n \rceil$, and the optimal parameters m should be $\lfloor \frac{n}{2} \rfloor$ or $\lceil \frac{n}{2} \rceil$, but the formula is also correct for us to find the $T_k(n)$, as divide and conquer method can also be described by the formula in (c), which means we divide the whole set into two part, do the testing and then dividing, but for $n \le 2k$, the optimal parameter m is not k, but $\lfloor \frac{n}{2} \rfloor$ ($n \le 2k \rightarrow \lfloor \frac{n}{2} \rfloor \le k$, suit this formula.)

Therefore, the proposed strategy uses the same no. of test with the optimal algorithm in (c), which prove optimality of the proposed test algorithm.