14. **(12 points)** We showed that there is a worst-case linear time SELECT algorithm that selects the i-th order statistic from n numbers with $T(n) = \Theta(n)$ comparisons. But the constant hidden by the Θ-notation is rather large. When i is small relative to n, we can implement a different procedure that uses SELECT as a subroutine but makes fewer comparisons in the worst case.

(a) **(9 points)** Describe an algorithm that uses $U_i(n)$ comparisons to find the i-th smallest of n elements, where

$$U_i(n) = \begin{cases} T(n) & \text{if } i \geq \frac{n}{2}; \\ \lfloor \frac{n}{2} \rfloor + U_i(\lceil \frac{n}{2} \rceil) + T(2i) & \text{if } i < \frac{n}{2}. \end{cases}$$

Briefly explain why your algorithm is correct. (Hint: Begin with $\lfloor \frac{n}{2} \rfloor$ disjoint pairwise comparisons, and recurse on the set containing the smaller element from each pair.)

(b) **(3 points)** Show that, if $i < \frac{n}{2}$, then $U_i(n) = n + O\left(T(2i)\log\frac{n}{i}\right)$.

4

a) if $i < \frac{n}{2}$, we can find the median of n, as pivot with $\lfloor \frac{n}{2} \rfloor$ comparisons, and only look into that partition of elements that are smaller or equal to the pivot, and "i" will still be in that partition.

Recursively, if each time i is in the smaller half of the partition (i.e, $i < \frac{n}{2}$) we can go into the faster subroutine, with less comparisons.

So it finds the ~~对~~ i-th ~~ele~~ order element faster.

b) divide-and-conquer:

tree for $U_i(\lceil \frac{n}{2} \rceil)$ has ~~height~~ partition times $\log \frac{n}{i}$ $= \log n - \log i$

each combine step takes: $T(2i)$

n is to loop through the n elements

So $U_i(n) = n + O(\log \frac{n}{i}) \cdot T(2i) = n + O(T(2i)\log \frac{n}{i})$