ECE374 SP23 HW4

Contributors

Zhirong Chen (zhirong4)

Ziyuan Chen (ziyuanc3)

Problem 3

Below is a divide-and-conquer sorting algorithm called FifthSort.

$$
\begin{array}{l}
\text{FifthSort}(A[1:n]) \\
\quad \text{if } n < 100 \\
\quad\quad \text{Sort } A \text{ by brute force} \\
\quad \text{else} \\
\quad\quad k = \lceil \frac{n}{5} \rceil \\
\quad\quad \text{FifthSort}(A[1:3k]) \qquad // \text{ First Sort} \\
\quad\quad \text{FifthSort}(A[2k+1:n]) \quad // \text{ Second Sort} \\
\quad\quad \text{FifthSort}(A[1:3k]) \qquad // \text{ Third Sort} \\
\quad\quad \text{FifthSort}(A[k+1:4k]) \quad // \text{ Fourth Sort}
\end{array}
$$

(a) Prove that FifthSort correctly sorts its input. (Hint: Where can the smallest k elements be?)

(b) Would FifthSort still sort correctly if we replace "$\text{if } n < 100$" with "$\text{if } n < 10$"?

(c) Would FifthSort still sort correctly if we replace "$\text{if } n < 100$" with "$\text{if } n < 13$"?

(d) Would FifthSort still sort correctly if we replace "$k = \lceil \frac{n}{5} \rceil$" with "$k = \lfloor \frac{n}{5} \rfloor$"?

(e) What is the running time of FifthSort? Set up a running-time recurrence and then solve it, ignoring the floors and ceilings.

Solution

(a)

Intuition. FifthSort divides the array into five segments each of length k, placed in five positions.

In the following example, the segments are assumed to initially appear in descending order (or "the furthest away" from their sorted positions).

Step	Pos. A	Pos. B	Pos. C	Pos. D	Pos. E
Initial	***5***	4	3	2	***1***
First Sort	3	4	***5***	2	***1***
Second Sort	3	4	***1***	2	***5***
Third Sort	***1***	3	4	2	***5***
Fourth Sort	***1***	2	3	4	***5***

Proof. If $n < 100$, we can always trust the brute-force algorithm.

For $n \geq 100$, we use ***mathematical induction***. Assuming $\mathrm{FifthSort}$ can sort $A[1:k], \forall k < m$, we prove that $\mathrm{FifthSort}$ can sort $A[1:m], m > 100$.

- First Sort on $A[1:3k]$ sorts the first $3k$ elements
  - Position C holds the largest segment among Segments A, B, C
- Second Sort on $A[2k+1:m]$ sorts the last $m-2k$ elements
  - Position E holds the largest segment among Position C, Segments D, E
    - That is, the largest among Segments A, B, C, D, E
  - Position C holds the smallest segment among Segments C, D, E
- Third Sort on $A[1:3k]$ sorts the first $3k$ elements
  - Position A holds the smallest segment among Segments A, B, Position C
    - That is, the smallest among Segments A, B, C, D, E
- Fourth Sort on $A[k+1:4k]$ sorts the middle $3k$ elements
  - The whole array is sorted
  - (P.s. Sorting $A[k+1:m-k]$ suffices if m is not dividable by 5)

(b) No.

If $n < 10$, we can always trust the brute-force algorithm.

But for some $n \geq 10$, say $n = 11$, $k = \lceil \frac{11}{5} \rceil = 3$. The **Fourth Sort** has bounds $A[k+1:4k] = A[4:12]$. Out of range!

P.s. This is in fact the ***only*** *case where the algorithm fails.*

(c) Yes.

If $n < 13$, we can always trust the brute-force algorithm; same as (b).

For $n \geq 13$, the mathematical induction process in (a) still applies.

Note that k not dividable by 5 is not an issue since the indices can handle the edge cases.
Adding dummy elements also works

(d) No.

Setting the brute-force threshold to 10, we present a counterexample ($n = 11$).

$$
\begin{aligned}
A[1:11] &= [5,6,7,8,9,10,0,1,2,3,4] \\
k &= \lfloor \frac{11}{5} \rfloor = 2 \\
\text{FifthSort}(A[1:6]) &= [\{\ 5,6,7,8,9,10\ \}\ \ ,0,1,2,3,4] \\
\text{FifthSort}(A[5:11]) &= [5,6,7,8,\ \ \{\ 0,1,2,3,4,9,10\ \}] \\
\text{FifthSort}(A[1:6]) &= [\{\ 0,1,5,6,7,8\ \}\ \ ,2,3,4,9,10] \\
\text{FifthSort}(A[3:8]) &= [0,1,\ \{\ 2,3,5,6,7,8\ \}\ ,4,9,10]
\end{aligned}
$$

The out-of-order 4 cannot be included in any of the five segments!

(e) $O(n^{2.714})$

```
              T(n)                    ··· C
     /       |        |        \
T(3n/5) T(3n/5) T(3n/5) T(3n/5) ··· C
   .       .        .          .
   .       .        .          .
   .       .        .          .
```

$$
\begin{aligned}
T(n) &= 4T(\frac{3n}{5}) + C \\
&\approx T(m) * \prod_{i=1}^{\log_{\frac{5}{3}} n} 4 \quad // \, T(m) \text{ is constant (brute-force)} \\
&= O(4^{\log_{\frac{5}{3}} n}) \\
&= O(n^{\log_{\frac{5}{3}} 4}) \\
&\approx O(n^{2.714})
\end{aligned}
$$

Intuition is that each of the $\log_{\frac{5}{3}} n$ levels of recursion operates in 4^{Depth} time.