CSE515 Advanced Algorithms
Lecture 23
Quicksort

Antoine Vigneron

antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

May 18, 2021

1. Introduction
2. Algorithm
3. Analysis
4. Analysis using phases
5. Proof by Induction
6. Conclusion

Course Organization

- Assignment 4 due this Friday.
- In this lecture, we present QUICKSORT, a randomized algorithm for sorting, and show how to analyze it.
- References:
  - Section 13.5 of Algorithm Design by Kleinberg and Tardos.
  - Section 7 of Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein. (Available online from the UNIST library website.)

Problem Statement

Problem

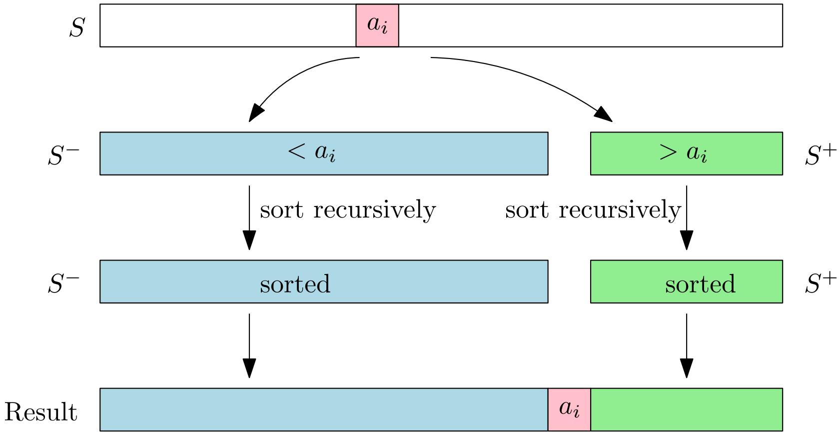
Given a set S of n numbers, the sorting problem is to compute a list of the elements of S in nondecreasing order.

Example

Given $S = \{8, 4, 5, 6, 12, 9, 7, 1\}$, then the answer is $A = [1, 4, 5, 6, 7, 8, 9, 12]$.

- To simplify the presentation, we will assume that the elements of S are distinct, i.e. $a_i \neq a_j$ whenever $i \neq j$.

Quicksort



S
a_i
S^-
$< a_i$
$> a_i$
S^+
sort recursively
sort recursively
S^-
sorted
sorted
S^+
Result
a_i

Quicksort

- First pick a pivot $a_i \in S$ at random.
- Then construct S^- and S^+, containing the elements of S that are smaller and larger than a_i, respectively.
- Sort S^- and S^+ recursively.
- Merge the results: S in sorted order is sorted S^-, followed by a_i, and by sorted S^+.

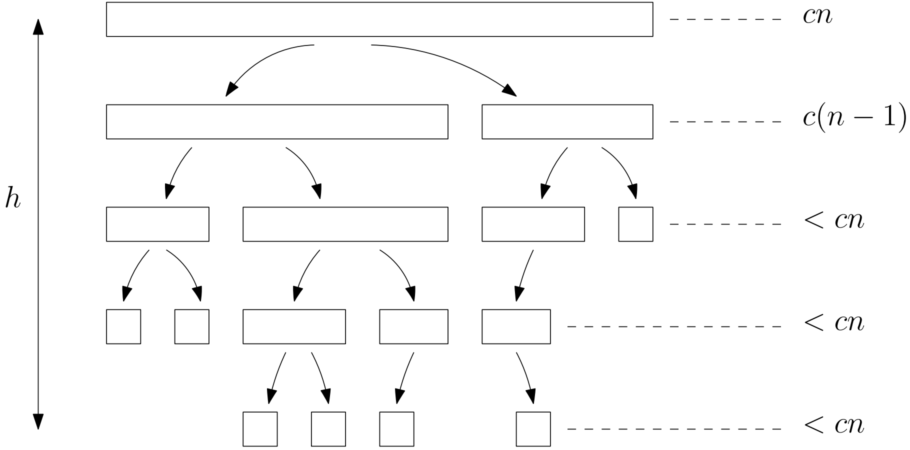
Quicksort

Pseudocode

```
procedure QUICKSORT(S)
    if |S| ⩽ 1 then return S
    pick a_i ∈ S at random
    for each a_j ∈ S do
        if a_j < a_i then insert a_j into S⁻
        if a_j > a_i then insert a_j into S⁺
    return QUICKSORT(S⁻) · a_i · QUICKSORT(S⁺)
```

- Here, the dot "·" means "concatenate".

Analysis: Recursion Tree

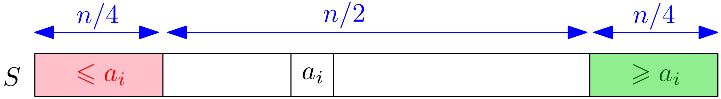


Analysis: Recursion Tree

- Ignoring recursive calls, QUICKSORT runs in $O(n)$ time.
- At each level of the recursion tree, the total size of the arrays is $\leqslant n$.
- So if h is the height of the tree, then the running time is $O(hn)$.
- How to bound h?

Analysis: Intuition

- As we did in previous lecture, we say that the pivot a_i is *central* if at least one quarter of the elements are $\leqslant a_i$ and at least one quarter are $\geqslant a_i$.

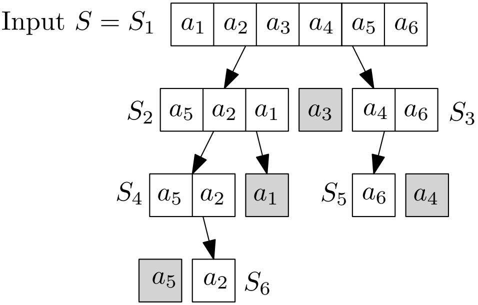


- Then the pivot is central with probability $1/2$.
- And if the pivot is central, the sizes of S^- and S^+ are at most $3n/4$.

Analysis: Intuition

- What can we say about the height of the recursion tree if all pivots are central?
- Then $(4/3)^h \leqslant n$, so $h \leqslant \log_{4/3} n = \frac{\log n}{\log 4/3} = O(\log n)$.
- Then QUICKSORT runs in time $O(n \log n)$.
- Half of the pivots are central.
- So that the expected worst-case running time should still be $O(2n \log n) = O(n \log n)$.
- This argument is not really a proof. In the following, we give two proofs that the expected running time is $O(n \log n)$.

Analysis using Phases



$|S_1| = 6$

$|S_2| = 3$ $\quad |S_3| = 2$

$|S_4| = 2$ $\quad |S_5| = 1$

$|S_6| = 1$

$$a_5 \leqslant a_2 \leqslant a_1 \leqslant a_3 \leqslant a_6 \leqslant a_4$$

Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
2	4	1	2	3	3

$$|Y_1| + \cdots + |Y_6| = |S_1| + \cdots + |S_6|$$
$$= 15$$

Analysis using Phases

- $S_1, \dots, S_k$: subsets on which we call QUICKSORT
- Y_i is the number of subsets S_k containing a_i:

$$Y_i = |\{1 \leqslant q \leqslant k \mid a_i \in S_q\}|.$$

- Then we have

$$\sum_{q=1}^{k} |S_q| = \sum_{i=1}^{n} Y_i$$

  This is a *double counting* argument.
- The running time of quicksort is proportional to $\sum_{q=1}^{k} |S_q|$.
- So it is proportional to $Y = \sum_{i=1}^{n} Y_i$.

Analysis using Phases

- By linearity of expectation

$$E[Y] = \sum_{i=1}^{n} E[Y_i]$$

  so we need to find a bound on $E[Y_i]$.

- We say that a_i is in *phase* j if it is contained in a set S_q such that

$$n\left(\frac{3}{4}\right)^{j+1} < |S_q| \leqslant n\left(\frac{3}{4}\right)^{j}.$$

Analysis using Phases

- If a_i is in phase j and the pivot is central, then a_i moves to phase $> j$.
- A pivot is central with probability $1/2$, so by the waiting time bound, so the expected number of times a_i remains in phase j is $\leqslant 2$.
- As there are $O(\log n)$ phases, it means that the expected number of sets S_q containing a_i is $O(2 \log n) = O(\log n)$.
- In other words, $E[Y_i] = O(\log n)$.
- Therefore, the expected running time of QUICKSORT is

$$E[Y] = \sum_{i=1}^{n} E[Y_i] = O(n \log n).$$

Proof by Induction

Lemma

Let f be the function defined by $f(x) = x \log x$ for all $x > 0$, and $f(0) = 0$. Then we have

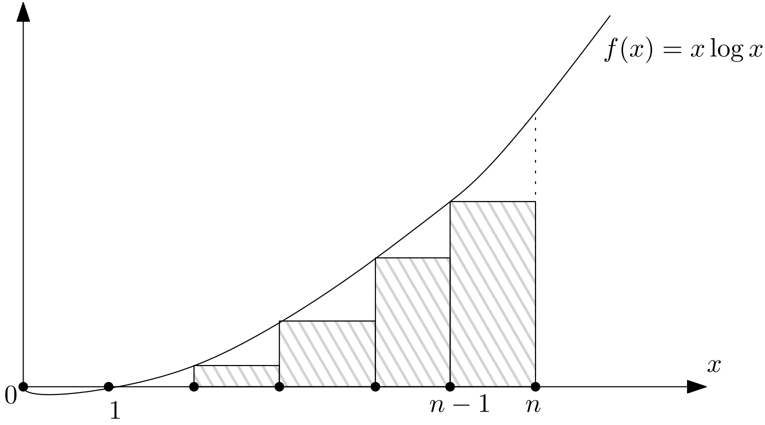
$$\sum_{i=0}^{n-1} f(i) \leqslant \frac{n^2}{2} \log n - \frac{n^2}{4} + \frac{1}{4}.$$

Proof.

$$\sum_{i=0}^{n-1} f(i) \leqslant \int_{x=1}^{n} x \log x dx \qquad \text{(See next slide.)}$$

A primitive function of $x \log x$ is $(x^2/2)\log(x) - (x^2/4)$, which yields the desired result. $\square$

Proof by Induction



Proof by Induction

- We will prove that the expected running time $T(n)$ satisfies

$$T(n) \leqslant af(n) + b$$

  for some constants a and b.

- Ignoring recursive calls, QUICKSORT runs in linear time, i.e. time cn for some constant c.
- When $|S^-| = i$, it recurses on sets of sizes i and $n - i - 1$.
- So we have

$$\begin{aligned} T(n) &\leqslant cn + \frac{1}{n} \sum_{i=0}^{n-1} T(i) + T(n-i-1) \\ &= cn + \frac{2}{n} \sum_{i=0}^{n-1} T(i). \end{aligned}$$

Proof by Induction

- Induction hypothesis (IH):

$$T(n) \leqslant af(n) + b \quad \text{for all} \quad m < n$$

- It implies

$$\begin{aligned} T(n) &\leqslant cn + \frac{2}{n} \sum_{i=0}^{n-1} af(i) + b \\ &= cn + 2b + \frac{2a}{n} \sum_{i=0}^{n-1} f(i). \end{aligned}$$

Proof by Induction

- By the lemma above, it implies

$$\begin{aligned} T(n) &\leqslant cn + 2b + af(n) - \frac{an}{2} + \frac{a}{4} \\ &\leqslant af(n) + \frac{2c-a}{2}n + \frac{8b+a}{4}. \end{aligned}$$

- To complete the proof, we need to obtain $T(n) \leqslant af(n)$, so it suffices that

$$\frac{2c-a}{2}n + \frac{8b+a}{4} \leqslant 0.$$

- It should be possible because for $n \geqslant 1$, this quantity goes to $-\infty$ when $a \to \infty$.

Proof by Induction

- Supose that $a \geqslant 8c$ and $n \geqslant 1$. Then we have

$$\begin{aligned}\frac{2c-a}{2}n+\frac{8b+a}{4} &\leqslant \frac{a/4-a}{2}n+\frac{8b+a}{4}\\ &= -\frac{3a}{8}n+\frac{8b+a}{4}\\ &\leqslant -\frac{3a}{8}+\frac{8b+a}{4}\\ &= 2b-\frac{a}{8}\end{aligned}$$

- In order for the inductive step to work, it suffices that $a \geqslant 16b$ and $a \geqslant 8c$. In addition, we need to have $T(0) \leqslant af(0)+b$, in other words $T(0) \leqslant b$.
- So we choose $b = T(0)$ and $a = \max(8c, 16b)$, and it follows that $T(n) \leqslant af(n)+b$.

Concluding Remarks

- There are better ways of implementing QUICKSORT than the pseudocode given in these slides. In particular, it can be done without creating auxiliary arrays. (See the MIT textbook.)
- Several algorithms are known that sort in time $O(n \log n)$: MERGESORT, HEAPSORT.
- QUICKSORT is a simpler algorithm and faster in practice, that runs in *expected* time $O(n \log n)$.
- It can be proved that there is no sorting algorithm faster than $O(n \log n)$ in the worst case. (At least no comparison-based algorithm. See CSE331.)

Concluding Remarks

- QUICKSORT as well as the randomized selection algorithm are simple and fast algorithms.
- Deterministic counterparts are more complicated and slower in practice.
- On the other hand analyzing randomized algorithms is not so easy.
- For many problems, randomized algorithms are simpler.
- They are more easily implemented and faster in practice.
- So it is interesting from a programmer standpoint.