COMP 460 Lecture Notes

R. I. Greenberg

1 Quicksort

1.1 Quicksort (Deterministic Version)

Worst-case time $\Theta(n^2)$ but average-case $\Theta(n \lg n)$ with small constant hidden in the Θ.

Quicksort(A, p, r) (Initial call is Quicksort$(A, 1, \text{length}[A])$.)
1 **if** $p < r$ **then**
2 $x \leftarrow A[r]$ (x is the "pivot" elt.)
3 Rearrange $A[p..r]$ so that $A[q] = x$ (for some q with $p \leq q \leq r$),
and $A[i] \leq x \leq A[j]$ whenever $i \leq q \leq j$.
4 Quicksort$(A, p, q-1)$
5 Quicksort$(A, q+1, r)$
6 **endif**

No merge is required due to the way we've partitioned.

Step 3 needs some elaboration: Easy to do in $\Theta(n)$ time (when $n = r - p$) using an auxiliary array. Book shows can even do it in place.

1.2 Quicksort Performance

Depends on sizes of subarrays into which we partition.

1.2.1 Worst-Case

We'll see later it's when we divide an n elt. array into arrays of size $n-1$ and 0 (plus the pivot). If we do this at each stage,

$$\begin{aligned} T(n) &= T(n-1) + \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$

(Occurs when input already sorted!)

1.2.2 Best-Case

When always divide array in half,

$$\begin{aligned} T(n) &= 2T(n/2) + \Theta(n) \\ &= \Theta(n \lg n) \end{aligned}$$

1.2.3 Average-Case

It's likely that many splits will be pretty balanced, so we'll see later avg. case is $\Theta(n \lg n)$ with a little larger constant.

2 Randomized Quicksort

- deterministic alg.: For avg. case, we need to assume, e.g., all inputs equally likely. We get in trouble if a bad input is presented often .
- randomized alg.: Design so good performance likely regardless of input, e.g., permute input randomly before applying original quicksort alg. We still end up with $\Theta(n^2)$ worst case, but now it depends on bad luck with the random number generator and not on the input.
- An easier to analyze randomized quicksort: Just like original version, except pick pivot elt. at random from $A[p..r]$.

2.1 Quicksort Analysis (Randomized)

2.1.1 Worst-Case

$$T(n) = \max_{0 \le k \le n-1} [T(k) + T(n-1-k)] + \Theta(n) \ .$$

Substitution method: Try $T(n) \le cn^2$:

$$T(n) \le \max_{0 \le k \le n-1} [ck^2 + c(n-1-k)^2] + \Theta(n) \ .$$

The max. is achieved at $k = 0$ or $k = n-1$ (by a calculus argument).

So

$$\begin{aligned} T(n) &\le c(n-1)^2 + \Theta(n) \\ &= cn^2 - 2cn + c + \Theta(n) \\ &\le cn^2 \text{ for large enough constant } c \end{aligned}$$

2.1.2 Average-Case

(Analysis here similar to CLRS 3rd ed. exercise 7–3.)

Like last recurrence, but instead of choosing k to maximize running time, each value of k is roughly equally likely. (Can make partition work this way even if not all elements distinct.)

Then, avg. time (expectation of running time) is

$$\begin{aligned} T(n) &= \Theta(n) + \frac{1}{n}\sum_{k=0}^{n-1}[T(k) + T(n-1-k)] \\ &= \Theta(n) + \frac{2}{n}\sum_{k=0}^{n-1} T(k) \end{aligned}$$

Solve by substitution method: Try $T(n) \le anlgn$:

$$\begin{aligned} T(n) &= \Theta(n) + \frac{2}{n}\sum_{k=1}^{n-1} T(k) \\ &\le \Theta(n) + \frac{2}{n}\sum_{k=1}^{n-1} aklgk \\ &= \Theta(n) + \frac{2a}{n}\sum_{k=1}^{n-1} klgk \\ &\le \Theta(n) + \frac{2a}{n}(\frac{1}{2}n^2 \lg n - \frac{1}{8}n^2) \text{ by CLRS 3rd ed. Exercise 7–3d} \\ &= \Theta(n) + an\lg n - \frac{a}{4}n \\ &\le an\lg n \text{ for large enough constant } a \end{aligned}$$

To handle the base case, you can change the guess to $an\lg n + b$ or prove only for $n \ge n_0$ with $n_0 > 1$.

Here is the proof for CLRS 3rd ed. Exercise 7–3d, using the techniques of “splitting the sum” and “bounding the terms” (which comes up again in the lower bound for comparison-based sorting):

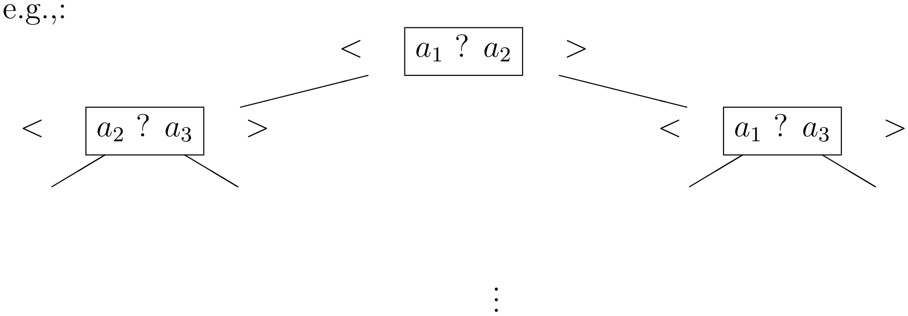
$$\begin{aligned} \sum_{k=1}^{n-1} k\lg k &= \sum_{k=1}^{\lceil n/2\rceil -1} k\lg k \quad + \quad \sum_{k=\lceil n/2\rceil}^{n-1} k\lg k \\ &\le (\lg n - 1)\Big(\sum_{k=1}^{\lceil n/2\rceil -1} k\Big) \quad + \quad (\lg n)\Big(\sum_{k=\lceil n/2\rceil}^{n-1} k\Big) \\ &= (\lg n)\sum_{k=1}^{n-1} k \quad - \quad \sum_{k=1}^{\lceil n/2\rceil -1} k \end{aligned}$$

$$
\begin{aligned}
&\le (\lg n)\frac{n}{2}(n-1) \quad - \quad \frac{n}{4}(\frac{n}{2}-1) \\
&\le \frac{1}{2}n^2 \lg n \quad - \quad \frac{1}{8}n^2
\end{aligned}
$$

3 Lower Bound on Comparison Sorts

Worst-case lower bound for algs. that don't use values of elts. except to compare them.

WLOG, assume all elts. distinct; a comparison asks which of two elts. is larger, e.g., $a_i?a_j$. The program may branch according to the answer ($<$ or $>$); described by a "decision tree", e.g.,:



Each execution of the program follows a path from the root to a leaf based on the results of the comparisons. When we reach a leaf, we must have a decision on the sorted order. Thus we need a distinct leaf for each of the $n!$ possible answers.

So $n! \le 2^h$ (where h is the tree height), which implies

$$h \ge \lg(n!) = \Theta(n \lg n)$$

(by CLRS 3rd ed. Exercise 3.2–3)

So the worst-case time is $\Omega(n \lg n)$.