Data Structures and Algorithms: Lecture 7

Barbara Morawska

September 4, 2018

Quicksort

- Worst case: $O(n^2)$.
- Average case: $O(n \lg n)$ and the hidden constants are small.
- Sorting in place.

Often the practical choice for sorting.

Quicksort - how does it work?

Divide-and-conquer like MERGE-SORT.
Input: subarray $A[p..r]$

Divide:

Arrange $A[p..r]$ into two subarrays:

- take $A[r]$
- $A[p..q-1]$ – contains all elements smaller than $A[r]$
- $A[q+1..r]$ – contains all elements bigger than $A[r]$
- return the index q (the right index for $A[r]$)

Conquer:

Recursively call QUICKSORT on the subarrays.

Combine:

The array $A[p..r]$ is sorted. Nothing left to do.

Quicksort – pseudocode

Procedure QUICKSORT(A, p, r)

```
if p < r then
    q = Partition(A, q, r)
    Quicksort(A, p, q − 1)
    Quicksort(A, q + 1, r)
```

Initially QUICKSORT is called on $(A, 1, A.length)$.

Quicksort – pseudocode

Procedure PARTITION(A, p, r)

```
x = A[r]
i = p - 1
for j = p to r do
    if A[j] ≤ x then
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

$x = A[r]$ is called a pivot element of the partition.
Running time: $\Theta(n)$

Run $\textsc{Partition}$ on $A =< 2, 8, 7, 1, 3, 5, 6, 4 >, 1, A.length$ and observe:

1. For each $A[k]$ such that $1 \leq k \leq i$, $A[k] \leq x$.
2. For each $A[k]$ such that $i + 1 \leq k \leq j - 1$, $A[k] \geq x$.

Worst case partitioning

The worst case is when the partition is unbalanced:
$\textsc{Partition}(A, p, r)$ produces one array with $n-1$ elements and the other with 0 elements.

Recurrence in the worst case:

$$\begin{aligned} T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n) \end{aligned}$$

The solution is: $T(n) = \Theta(n^2)$

(The same as Insertion-sort.)

When the worst case occurs?
What is the running time of Insertion-sort in this case?
Is this really the worst case partitioning?

Best case partitioning

The partition is balanced:
the subproblems have size $\leq \frac{n}{2}$

Recurrence in the best case:

$$T(n) = 2T(\frac{n}{2}) + \Theta(n)$$

The solution (case 2 of Master Theorem): $T(n) = \Theta(n \lg n)$

How much unbalanced the input can be?

If the partition yields 9-to-1 split, is this still balanced?

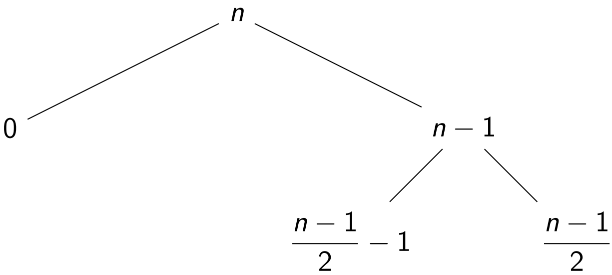
Recurrence for 9-to-1 split

$$T(n) = T(\frac{9n}{10}) + T(\frac{n}{10}) + cn$$

- The recursion terminates at the depth of the recursion tree: $\log_{\frac{10}{9}} n = \Theta(\lg n)$.
- Hence the solution is $O(n \lg n)$.
- If the problem is divided with constant proportionality, the running time will be $O(n \lg n)$.

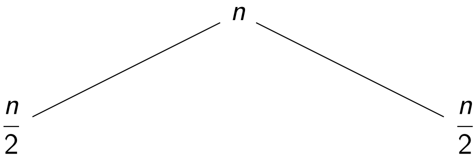
Average case...

Suppose the best case and the worst case partitions alternate



Cost of partitioning n is $\Theta(n) + \Theta(n-1) = \Theta(n)$ and notice the size of the subproblems.

Compare with the best case partitioning with cost $\Theta(n)$



Randomized version of QUICKSORT

Randomize QUICKSORT to ensure that all the permutations of input are equally likely.

Random sampling

- Instead of using $A[r]$ as the pivot, randomly choose an element from $A[p..r]$.
- Randomly choose a pivot and exchange it with $A[r]$
- Any of the elements in the array is equally likely to be a pivot.
- Expect a balanced split.

Randomized version of QUICKSORT

Procedure RANDOMIZED-PARTITION(A, p, r)

$i = RANDOM(p, r)$
exchange $A[r]$ with $A[i]$
return PARTITION(A, p, r)

Procedure RANDOMIZED-QUICKSORT(A, p, r)

if $p < r$ **then**
    $q =$ RANDOMIZED-PARTITION(A, q, r)
    RANDOMIZED-QUICKSORT($A, p, q - 1$)
    RANDOMIZED-QUICKSORT($A, q + 1, r$)

What is really the worst case partitioning?

Proof that the bad split is the worst

The general form of the recurrence for QUICKSORT:

$$T(n) = T(q) + T(n - q - 1) + \Theta(n)$$

- Worst case: $T(n) = \max_{0 \leq q \leq n-1} (T(q) + T(n - q - 1)) + \Theta(n)$ (counting from 0)
- Guess $T(n) \leq cn^2$ and use substitution method.
-

$$\begin{aligned} T(n) &\leq \max_{0 \leq q \leq n-1} (cq^2 + c(n - q - 1)^2) + \Theta(n) \\ &= c \cdot \max_{0 \leq q \leq n-1} (q^2 + (n - q - 1)^2) + \Theta(n) \end{aligned}$$

- $q^2 + (n - q - 1)^2$ achieves maximum at either $q = 0$ or $q = n - 1$. In either case we have bad split!

Expected running time of QUICKSORT

Assume that all the values of an input array are distinct.

Running time and comparisons

- There can be at most n calls to PARTITION
- Each call takes $O(1)$ plus the time for comparisons.
- How many comparisons must be made in line 4 of PARTITION?

Lemma 7.1

If X is the number of comparisons made during run of QUICKSORT on an n-element array, then the running time of QUICKSORT is $O(n + X)$

Proof.

n calls to PARTITION and X comparisons counted separately. $\square$

Expected running time of QUICKSORT

Observations, definitions...

- Each pair of elements in an array is compared only once (because we compare an element with a pivot, and then pivot is never compared again).
- Notation: let the array $A = z_1, z_2, \ldots, z_n$
- Notation: let Z_{ij} be a subarray from z_i to z_j, $i < j$.
- Define an indicator random variable:

$$X_{ij} = I\{z_i \text{ is compared to } z_j\}$$

  z_i is chosen to be a pivot or z_j is chosen.

- Total number of comparisons: $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$
- What is the expectation of X?

Expectation of the number of comparisons

$$
\begin{aligned}
E[X] &= E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \qquad \text{by linearity of expectation} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr\{z_i \text{ is compared to } z_j\}
\end{aligned}
$$

What is $Pr\{z_i \text{ is compared to } z_j\}$?

$= Pr\{z_i \text{ or } z_j \text{ is chosen to be a pivot in } Z_{ij}\}$

$= Pr\{z_i \text{ is chosen to be a pivot for } Z_{ij}\} +$

$\qquad Pr\{z_j \text{ is chosen to be a pivot for } Z_{ij}\}$

$$= \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}$$

Expectation of the number of comparisons

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} && \text{for } k = j - i \\
&< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} && \text{harmonic series} \\
&= \sum_{i=1}^{n-1} O(\lg n) \\
&= O(n \lg n)
\end{aligned}
$$

Hence the expected running time of RANDOMIZED QUICKSORT is $O(n \lg n)$