Data Structures and Algorithms
Randomized Quicksort
February 01, 2018

Quicksort

In quicksort, we first decide on the pivot. This could be the element at any location in the input array. The function `Partition` is then invoked. `Partition` accomplishes the following: it places the pivot in the location that it should be in the output and places all elements that are at most the pivot to the left of the pivot and all elements greater than the pivot to its right. Then we recurse on both parts. The pseudocode for Quicksort is as follows.

```
QSort(A[lo..hi])
  if hi <= lo then
    return
  else
    pivotIndex = floor((lo+hi)/2)  (this could have been any location)
    loc = Partition(A, lo, hi, pIndex)
    QSort(A[lo..loc-1])
    QSort(A[loc+1..hi])
```

One possible implementation of the function `Partition` is as follows.

```
Partition(A, lo, hi, pIndex)
  pivot = A[pIndex]
  swap(A, pivotIndex, hi)
  left = lo
  right = hi-1
  while left <= right do
     if (A[left] <= pivot) then
        left = left + 1
     else
        swap(A, left, right)
        right = right - 1
  swap(A, left, hi)
  return left
```

The worst case running time of the algorithm is given by

$$T(n) = \begin{cases} 1, & n = 1 \\ T(n-1) + cn, & n \geq 2 \end{cases}$$

Hence the worst case running time of `QSort` is $\Theta(n^2)$.

An instance where the quicksort algorithm in which the pivot is always the first element in the input array performs poorly is an array in descending order of its elements.

Randomized Quicksort

In randomized version of quicksort, we pick a pivot uniformly at random from all possibilities. We will now show that the expected number of comparisons made in randomized quicksort equals $2n \ln n + O(n)$.

Theorem. For any input, the expected number of comparisons made by randomized quicksort is $2n \ln n + O(n)$.

Proof. Let $y_1, y_2, \ldots, y_n$ be the input elements in sorted order. Let X be a random variable denoting the total number of pair-wise comparisons made between elements of the input array A. Let X_{ij} be a random variable denoting the total number of times elements y_i and y_j are compared during the algorithm. Observe the following.

- comparisons between elements in the input array are done only in the function `Partition`.
- there are $n-1$ distinct pivots chosen in the algorithm and hence $n-1$ calls to `Partition`.
- two elements are compared only if one of them is a pivot.

Let X_{ij}^k be an indicator random variable that is 1, iff elements y_i and y_j are compared in the kth call to `Partition`. Note that

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \qquad \text{and} \qquad X_{ij} = \sum_{k=1}^{n-1} X_{ij}^k$$

We will now calculate $\mathbf{E}[X_{ij}]$. By the linearity of expectation, we have

$$\mathbf{E}[X_{ij}] = \sum_{k=1}^{n-1} \mathbf{E}[X_{ij}^k] = \sum_{k=1}^{n-1} \Pr[X_{ij}^k = 1]$$

Let z be the first call to `Partition` during which one of the elements from $y_i, y_{i+1}, \ldots, y_j$ is used as the pivot. From our observations above, note that for all h and for all ℓ such that $1 \leq h < z$ and $z < \ell < n$, $X_{ij}^h = 0$ and $X_{ij}^\ell = 0$. If one of y_i or y_j is chosen as the z^{th} pivot then clearly, $X_{ij}^z = 1$, otherwise, $X_{ij}^z = 0$, and y_i and y_j will be separated into different sublists and hence will never be compared again. Thus we have

$$\mathbf{E}[X_{ij}] = \Pr[X_{ij}^z = 1] = \frac{2}{j - i + 1} \tag{1}$$

Applying the linearity of expectation and using (1), we get

$$\begin{aligned}
\mathbf{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbf{E}[X_{ij}] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{k=2}^{n} \frac{2(n-k+1)}{k} \\
&= (n+1)\sum_{k=2}^{n} \frac{2}{k} - 2(n-1) \\
&= 2(n+1)\sum_{k=1}^{n} \frac{1}{k} - 2(n-1) - 2(n+1) \\
&= 2(n+1)\ln n + c - 4n \qquad (c \text{ is a constant less than } 1) \\
&= 2n\ln n + O(n)
\end{aligned}$$