QuickSort and QuickSelect notes — Oct. 9, 2014

Here are some notes on QuickSort and QuickSelect, which will hopefully clear up some confusion from the last lecture. The analysis of QuickSelect is essentially the same as that in the textbook, except I have filled in a few details (hopefully, without obscuring things too much).

Recall that QuickSelect takes as input a list $L$ and an index $k$, such that $1 \leq k \leq |L|$, and returns the $k$th smallest item in $L$. It makes use of a partition procedure that chooses a random pivot $x$ from $L$, and splits $L$ into three sublists: $L_<$, $L_=$, and $L_>$, where $L_<$ (resp., $L_=$, $L_>$) consists of all those items in $L$ that are less than (resp., equal to, greater than) $x$. QuickSelect works as follows:

QuickSelect($L$, $k$):
  choose a random pivot and partition $L$ into sublists $L_<$, $L_=$, $L_>$
  if $k \leq |L_<|$ then
    return QuickSelect($L_<$, $k$)
  else if $k \leq |L_<| + |L_=|$
    return an arbitrary item from $L_=$
  else
    return QuickSelect($L_>$, $k - (|L_<| + |L_=|)$

Let $T_{L,k}$ be a random variable representing the running time of QuickSelect in finding the $k$th smallest item in a list $L$ of distinct items. Let $\tilde{T}(n)$ be the maximum value of $E[T_{L,k}]$ over all $L, k$ with $1 \leq k \leq |L| \leq n$.

We want to show $\tilde{T}(n) = O(n)$.

So fix $n$ and consider the behavior of QuickSelect in finding the $k$th smallest item in a list $L$ of at most $n$ distinct items. Let us run the algorithm, performing successive partitions until the recursion stops, or we get a subproblem of size at most $(3/4)n$, whichever comes first. We then pause the execution of the algorithm and take a look at what has happened and what remains to be done. Say that at this point, we have performed a total of $P$ partitions, and the subproblem we end up with (if any) takes additional time $S$ to solve. Here, $P$ and $S$ are random variables. We have $T_{L,k} \leq cP + S$, where $c$ is an implementation-defined constant. By linearity of expectation, we have $E[T_{L,k}] \leq cnE[P] + E[S]$.

So it remains to bound $E[P]$ and $E[S]$. Now, $E[P]$ is bounded by a constant $d \approx 2$, because the probability of getting a “good” split — one that reduces the input size by a factor of $3/4$ — is roughly $1/2$. We discussed this point in class.

Since the remaining subproblem is always of size at most $(3/4)n$, it follows that $E[S] \leq \tilde{T}((3/4)n)]$. This is perhaps “intuitively obvious”, but one can verify it a bit more carefully as follows. Let $Q$ be a random variable representing the remaining subproblem. The time $S$ depends on both the subproblem $Q$ and the random choices made by the algorithm in solving $Q$. For every possible value $q$ taken by $Q$, consider the conditional distribution of $S$ given that $Q = q$: one (unlikely) possibility is that the recursion has stopped, and $q$ is the empty problem, so that $E[S \mid Q = q] = 0$; the other (more likely) possibility is that $q$ represents a subproblem $L', k'$ with $1 \leq k' \leq |L'| \leq (3/4)n$, so that $E[S \mid Q = q] = E[T_{L', k'}] \leq \tilde{T}((3/4)n)]$. In either case, we have $E[S \mid Q = q] \leq \tilde{T}((3/4)n)]$. Therefore, by total expectation, we have

$$E[S] = \sum_{q} E[S \mid Q = q] P(Q = q) \leq \sum_{q} \tilde{T}((3/4)n)] P(Q = q)$$

$$= \tilde{T}((3/4)n)] \sum_{q} P(Q = q) = \tilde{T}((3/4)n)].$$

So we have

$$E[T_{L,k}] \leq cnE[P] + E[S] \leq cdn + \tilde{T}((3/4)n)].$$

Since this holds for all $L, k$ with $1 \leq k \leq |L| \leq n$, we have

$$\tilde{T}(n) \leq cdn + \tilde{T}((3/4)n)],$$

which, together with

$$\tilde{T}(0) = 0,$$

1
gives us a recurrence relation for $\tilde{T}$. One calculates:

$$\tilde{T}(n) \leq cdn(1 + (3/4) + (3/4)^2 + \cdots) = O(n).$$

Let $D_{L,k}$ be a random variable representing the recursion depth of QuickSelect in finding the $k$th smallest item in a list $L$ of distinct items. Let $\tilde{D}(n)$ be the maximum value of $E[D_{L,k}]$ over all $L, k$ with $1 \leq k \leq |L| \leq n$. Then the above argument shows that

$$\tilde{D}(n) \leq d + \tilde{D}((3/4)n),$$

which implies that $\tilde{D}(n) = O(\log n)$.

Now consider QuickSort. It takes as input a nonempty list $L$ of items to be sorted, and runs as follows:

```
QuickSort(L):
  choose a random pivot and partition $L$ into sublists $L_\prec, L_\equiv, L_\succ$
  if $|L_\prec| > 0$ then $L_\prec \leftarrow$ QuickSort($L_\prec$)
  if $|L_\succ| > 0$ then $L_\prec \leftarrow$ QuickSort($L_\succ$)
  return $L_\prec || L_\equiv || L_\succ$
```

Assume we are sorting a list $L$ of $n$ distinct items. For $k = 1, \ldots, n$, define $D'_{L,k}$ to be the depth of the node in the recursion tree where the $k$th smallest item occurs as a pivot. We observed in class that this item appears in subproblems in the recursion tree only at levels $0, \ldots, D'_{L,k}$. It follows that if $T'$ is the total running time of QuickSort, then

$$T' \leq c' \sum_{k=1}^{n} (D'_{L,k} + 1) = c'n + c' \sum_{k=1}^{n} D'_{L,k},$$

for some constant $c'$, and so by linearity of expectation,

$$E[T'] \leq c'n + c' \sum_{k=1}^{n} E[D'_{L,k}].$$

We also observed in class that $D'_{L,k}$ has precisely the same distribution as $D_{L,k}$, the recursion depth of QuickSelect in finding the $k$th smallest item in $L$, and since the expected value of the latter is $O(\log n)$, the bound $T' = O(n \log n)$ is now immediate.