CS-E3190 Principles of Algorithmic Techniques

06. Randomized Algorithms – Tutorial Exercise

1. **Revisiting Quicksort.** In the lectures we proved that paranoid Quicksort has an expected runtime of $O(n \log n)$. Here we prove that the runtime is $O(n \log^2 n)$ *with high probability*. Let us recap what paranoid Quicksort does. Recall that A is an array of n *unique* integer elements and l, r are indices of A. We call an index p of A a pivot. Given a pair A, p we can *partition* A on p such that all elements of A smaller than $A[p]$ are put to the left of p, and all elements of A larger than $A[p]$ are to the right of p. A pivot p (and the corresponding partition) is called *good* if at least $|A|/10$ elements are larger and at least $|A|/10$ are smaller than $A[p]$, or $|A| < 10$. The algorithms are shown below.

Algorithm 1: QUICKSORT(A, l, r)

```
if l ≥ r then
    return A;
repeat
    Randomly select p in {l, ..., r};
    PARTITION(A[l, r], p)
until PARTITION(A[l, r], p) is good;
QUICKSORT(A, l, p − 1)
QUICKSORT(A, p + 1, r)
```

Algorithm 2: PARTITION(A, p)

```
SWAP(A[0], A[p])
i ← 1
for j = i, i + 1, ... do
    if A[j] < A[0] then
        SWAP(A[i], A[j]);
        i ← i + 1;
end
SWAP(A[i − 1], A[0])
```

Let $T(k)$ be the randomized runtime of QUICKSORT$(A, 0, k-1)$ on array A of length k. Define a new random variable $N(k)$ as the number of partition-calls during one call of Quicksort on array A of size k; $N(k)$ does not include the time of recursive calls. Because the runtime of one Partition call on array A of size k takes $O(k)$ time, we have

$$T(k) \leq T(k-i) + T(i) + O(k) \cdot N(k)$$

for any i. In the lectures we studied the expectation of $T(n)$ and used $\mathrm{E}[N(n)] \leq 2$ to prove that $\mathrm{E}[T(n)] = O(n \log n)$. To prove the high probability result we want to bound $N(n)$ with high probability, as opposed to bounding its expectation.

(a) Consider the nodes of the recursion tree of the algorithm. Each node is associated with one Quicksort call. Take any node i in the tree. Let k_i be the size of the subarray of A at i. Say the node i is *bad* if $N(k_i) > 3 \log n$. Recall that $N(k_i)$ is the number of partition-calls during one call of Quicksort on an array of size k_i. Prove that the probability of node i being bad satisfies the following bound.

$$\mathrm{P}[\text{node } i \text{ is bad}] \leq (2/5)^{3 \log n}.$$

Solution. First, note that if the number of elements in the subarray A at node i is $k_i < 10$ the bound is trivially true; when $k_i < 10$ any pivot is good, so only one Partition call is made and the probability of i being bad is 0. Suppose now that $k_i \geq 10$. The probability of picking a good pivot is bounded by

$$\mathrm{P}[p \text{ is good}] \geq 6/10 = 3/5.$$

To see this, consider $k_i = 11$. There are only $7/11$ entries that give good pivots[1]. The probability is never lower than this; we use a conservative bound of $3/5$.

[1] This is a technical detail. The reason we do not use $4/5$ is that $|A|$ is not generally divisible by 10. To ensure that at least $|A|/10$ elements are larger than $A[p]$ we may need to round the value $|A|/10$ up. We can lose up to two elements to rounding.

Thus the probability of choosing a bad pivot is at most $1 - 3/5 = 2/5$. Each successive choice of pivot is independent. To observe t or more attempts one must pick a bad pivot t times in a row. This occurs with probability

$$\begin{aligned} \mathrm{P}[N(k) > t] &= \mathrm{P}[(1^{st}\text{pivot bad}) \cap (2^{nd}\text{pivot bad}) \cap \cdots \cap (t^{th}\text{pivot bad})] \\ &= \prod_{j=1}^{t} \mathrm{P}[(j^{th}\text{pivot bad})] \leq (2/5)^t \end{aligned}$$

The second equality follows from independence, and the inequality from our bound. You can also argue the random variable $N(k)$ follows a geometric distribution. Finally, plugging in $3 \log n$ for t gives the desired bound.

(b) Observe that there are at most $n \log n$ nodes in the recursion tree (note that this is a very loose bound). Prove that

$$\mathrm{P}[\text{at least one node is bad}] \leq n \log n \cdot (2/5)^{3 \log n}.$$

Solution. We can use the union bound and part (a). Let M be the number of nodes.

$$\begin{aligned} \mathrm{P}[\text{at least one node is bad}] &= \mathrm{P}\left[\bigcup_{i=1}^{M} (i^{th} \text{ node is bad})\right] \\ &\leq \sum_{i=1}^{M} \mathrm{P}[(i^{th} \text{ node is bad})] \\ &\leq M(2/5)^{3 \log n} = n \log n \cdot (2/5)^{3 \log n}. \end{aligned}$$

Note that the union bound here is particularly useful because we do not know whether events $(i^{th}$ node is bad$)$ and $(j^{th}$ node is bad$)$ are independent.

(c) Prove that the probability of having no bad nodes is at least $1 - n^{-1}$.

Solution. We can use what we derived in part (b).

$$\begin{aligned} \mathrm{P}[\text{no nodes are bad }] &= 1 - \mathrm{P}[\text{at least one node is bad}] \\ &= 1 - n \log n \cdot (2/5)^{3 \log n} \\ &= 1 - n \log n \cdot n^{-3} \\ &\geq 1 - n^{-1}, \end{aligned}$$

where the last inequality follows from $n \geq \log n$, where we assume $n > 1$.

(d) Explain why we have a runtime of $O(n \log^2 n)$ with high probability.

Solution. We have established in part (c) that with high probability, no nodes are bad and hence $N(k_i) = 3 \log k_i = O(\log n)$ for an array of size k_i. So the cost per recursion level is $O(n) \cdot O(\log n) = O(n \log n)$. Since the recursion tree depth is $O(\log n)$, the total cost is $O(\log n) \cdot O(n \log n) = O(n \log^2 n)$ with high probability.

2. **Partitioning, Union Bound.** Consider partitioning the vertex set of a tree $T = (V, E)$ into two sets V_1 and V_2 (i.e., $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$) such that each vertex belongs to either V_1 or V_2 with a probability of $1/2$, independently of the other vertices. Let $T[V_1]$ and $T[V_2]$ denote the subgraphs induced by node sets V_1 and V_2 respectively. Show that the partitioning leads to each connected component of $T[V_1]$ and $T[V_2]$ having a diameter of $O(\log n)$, with high probability.

Solution. The argument is rather simple. Each path of length $\omega(\log n)$ is present in $T[V_1]$ or $T[V_2]$ with probability $1/2^{\omega(\log n)}$, which is less than $1/\text{poly}(n)$. As there are at most $O(n^2)$ distinct paths in the tree, we can union bound over all of them and conclude that no path of length $\omega(\log n)$ will be present in $T[V_1]$ or $T[V_2]$, with probability $1 - O(n^2)/\text{poly}(n) = 1 - 1/\text{poly}(n)$. This is equivalent to each connected component of $T[V_1]$ and $T[V_2]$ having a diameter of $O(\log n)$, with high probability.