Algorithms & Data Structures 2024/25

Practical Week 11

I realise that some model answers may be easily available. Please however try to come up with your own solutions. If you do get stuck then ask the demonstrators for help.

1. Finish anything that may be left over from last week.

2. **Worst-Case Inputs for QuickSort.** Consider an implemention of QuickSort where the Partition() function does not re-arrange the elements that are smaller than the pivot and does not rearrange the elements that are larger than the pivot. In other words, after execution of the Partition() function the elements that are smaller than the pivot appear in the same order as they appeared originally, and the same holds for the elements that are larger than the pivot. For each of the following choices of the pivot element, explain how to construct a worst-case input with n elements that causes QuickSort to make a quadratic number of comparisons (that is, $\Omega(n^2)$ comparisons) in general, and give an example of such a worst-case input with $n = 10$. It suffices to consider inputs with n elements that are permutations of the n integers $\{1, 2, \ldots, n\}$.

   (a) The rightmost element of the array is chosen as pivot element.

   (b) The leftmost element of the array is chosen as pivot element.

   (c) The element in the middle of the array is chosen as pivot element. In other words, when Partition(A,left,right) is called and right>left, then the element $A[m]$ with $m = \left\lceil \frac{\text{left+right}}{2} \right\rceil$ is chosen as pivot element.

3. **Balanced QuickSort.** Assume (as we did on Slide 80 of the lecture slides) that QuickSort happens to choose pivots that always split the inpout array into subproblems of size $\frac{2}{15}n$ and $\frac{13}{15}n$, giving the recurrence

$$T(n) = T\left(\frac{2}{15}n\right) + T\left(\frac{13}{15}n\right) + cn$$

   where $c \geq 1$ is some constant (and where we ignore the rounding to integers of the sizes of the recursive subproblems).

   Assuming (as inductive hypothesis) that $T(n') \leq \alpha n' \log_2 n'$ for a suitable choice of the constant α holds for all $n' < n$, make the inductive step, that is, prove (using the recurrence relation above) that $T(n) \leq \alpha n \log_2 n$ also holds.

4. **Balls into Bins, and "with high probability" analysis.**

   Solving this question requires familiarity with basic probability theory. Please feel free to skip this questions if you feel that you are not sufficiently familiar with probability theory.

   A common way to strengthen statements to do with performance of randomised algorithms (such as the expected running time of $O(n \log n)$ for randomised quicksort) is to reanalyse them in terms of what is called "high probability", that is, to establish a bound on the probability for a given random variable (expressing the performance of such an algorithm) to deviate from its expectation by "much", for some sensible value of "much".

   One definition of "high probability" is, given a random variable X,

$$P(\text{good outcome}) = P(X \leq \text{some threshold}) = 1 - o(1),$$

   and a stronger one is

$$P(\text{good outcome}) = P(X \leq \text{some threshold}) \geq 1 - 1/n^k$$

for some constant $k \geq 1$, with n, as usual, reflecting the size of the input in a meaningful fashion (why is the second one stronger than the first one?). These can also be read as

$$P(\text{bad outcome}) = P(X > \text{some threshold}) = o(1)$$
$$P(\text{bad outcome}) = P(X > \text{some threshold}) < 1/n^k$$

in the sense that e.g. the probability for the running time of a randomised algorithm (expressed as X) to exceed a certain value (the "threshold" is small.

A commonly used tool in such an analysis is Chernoff's inequality, of which we state a simplified variant here:

Theorem 1 (Variant of Chernoff's inequality)**.** *Let X be a random variable defined as $X = X_1 + X_2 + \cdots + X_n$ where each X_i is a $0/1$-valued (Bernoulli) random variable, and all X_i are independent. Let $p_i = P(X_i = 1)$. Then $E[X] = \sum_{i=1}^{n} P(X_i = 1) = \sum_{i=1}^{n} p_i$ (see below for more on linearity of expectation to see this). For any $\delta \geq e^2 - 1$,*

$$P(X \geq (1+\delta)E[X]) \leq e^{-(\delta+2)E[X]}.$$

Another one is the Union bound:

Theorem 2 (Union bound)**.** *For any countable collection of events $\{A_i\}$,*

$$\Pr(\bigcup_i A_i) \leq \sum_i \Pr(A_i)$$

Typical use: to show that if an algorithm can fail only if various improbable events occur, then the probability of failure is no greater than the sum of the probabilities of these events. It reduces the problem of showing that an algorithm works with probability $1 - \epsilon$ to constructing an error budget that divides the ϵ probability of failure among all the bad outcomes.

Finally, another frequently used tool is called "linearity of expectation". It says that for any two random variables X and Y, $E[X + Y] = E[X] + E[Y]$. This can be generalised to any finite number of random variables.

Consider the following probabilistic process. You have a collection of n many "bins", and also n many "balls". The balls are thrown into the bins independently and uniformly at random, that is, for each ball, every bin has the same probability of being chosen for the ball. This process if frequently used to model and analyse load balancing mechanisms.

(a) Show that for a $0/1$ (Bernoulli) random variable X, $E[X] = P(X = 1)$.

(b) What is the expected number of balls in the first bin? In the 17-th bin?
Hint: Use linearity of expectation.

(c) As formally as possible, using both theorems from above, prove that with high probability (of the form $1 - 1/n^k$ for some constant $k \geq 1$ of your choosing) the maximum number of balls in any bin is $O(\log n)$.
Hint: First fix your favourite bin and show, using Chernoff's inequality, that *this* bin is unlikely to receive too many balls, and then use the Union bound to show that it's unlikely that there exists a bin with too many balls.