Lecture 12

Average Case Quicksort

Sample space $S_n = \{$ all permutations of $[1, 2, \ldots, n]\}$ with uniform probability distribution (not necessarily a good assumption in general).
Note implicit assumption in our choice of S_n: no repeated element. Why is this reasonable?

- running time can only decrease with repeated values (they get put into E and never examined again), so assumption does not underestimate complexity for more general sample spaces
- more general samples spaces tricky to define clearly and precisely

Input S one permutation from S_n.
$t_n(s) = \#$ of comparisons on input S in S_n.
Let $i =$ first element of S. This will be the pivot.
$t_0(S) = 0, t_1(S) = 0, t_n(s) = n - 1 + t_{i-1}(L) + t_{n-1}(G)$
Because of our choice of probability distribution over the sample space, all permutations are equally likely
$\Rightarrow$ each element of $\{1, 2, \ldots, n\}$ equally likely to be first in S
$\Rightarrow$ pivot equal 1 with probability $1/n$, …, pivot equal n with probability $1/n$
Also, after parition, L, G equally likely to be in any order.

This leads to recurrence for $T'(n) = E[t_n]$:

$$\begin{aligned}
T'(0) &= 0 \\
T'(1) &= 0 \\
T'(n) &= E[t_n] \\
&= \sum_{i=1}^{n} \frac{1}{n}(n - 1 + T(i-1) + T(n-i)) \\
&= n - 1 + \frac{2}{n}\sum_{j=1}^{n-1} T'(j)
\end{aligned}$$

First, write down expressions for $T'(n)$ and $T'(n-1)$:

$$T'(n) = n - 1 + \frac{2}{n}\sum_{j=1}^{n-1} T'(j)$$
$$T'(n-1) = n - 2 + \frac{2}{n-1}\sum_{i=1}^{n-2} T'(i)$$

We want to subtract $T'(n)$ and $T'(n-1)$, but n and $n-1$ disagree, so we can't.

$$nT'(n) = n(n-1) + 2\sum_{i=1}^{n-1} T'(i)$$
$$(n-1)T'(n-1) = (n-1)(n-2) + 2\sum_{i=1}^{n-2} T'(i)$$

$$\begin{aligned}
nT'(n) - (n-1)T'(n-1) &= n(n-1) + 2\sum_{i=j}^{n-2} T'(j) - (n-2)(n-1) - 2\sum_{i=j}^{n-2} T'(j) \\
&= 2T'(n-1) + 2(n-1) \\
nT'(n) &= (n+1)T'(n-1) + 2(n-1) \\
\frac{nT'(n)}{n(n+1)} &= \frac{2(n-1)}{n(n+1)} + \frac{(n+1)T'(n-1)}{n(n+1)} \\
\frac{T'(n)}{n+1} &= \frac{2(n-1)}{n(n+1)} + \frac{T'(n-1)}{n}
\end{aligned}$$

Let $A(n) = \frac{T'(n)}{n+1}, A(0) = \frac{T'(0)}{1} = 0, A(1) = 0$

$$\begin{aligned}
A(n) &= A(n-1) + \frac{2(n-1)}{n(n+1)} \\
&= \sum_{i=1}^{n} \frac{2(i-1)}{i(i+1)} \\
&= 2\sum_{i=1}^{n} \frac{i}{i(i+1)} - 2\sum_{i=1}^{n} \frac{1}{i(i+1)} \\
&= 2 \underbrace{\sum_{i=1}^{n} \frac{1}{(i+1)}}_{\text{harmonic series } \in \ln n} - 2\sum_{i=1}^{n} \frac{1}{i(i+1)}
\end{aligned}$$

$$\sum_{i=1}^{n} \frac{1}{i} \text{ is } \ln n + O(1)$$

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \sum_{i=1}^{n} \frac{1}{i} - \sum_{i=1}^{n} \frac{1}{i+1} = 1 - \frac{1}{n+1} \rightarrow \text{ telescoping series } = \frac{1}{1} - \frac{1}{n}$$

Thus $A(n)$ is $\ln n$, and $T'(n) = A(n)(n-1) \in \Theta(n \ln n)$.

Randomized QuickSort

- Average-case analyis works only if each permutation equally likely. In practice, this may not be true.
- Instead of relying on unknown distribution of inputs, randomize by picking random element as pivot. This way, random behaviour of algorithm on any fixed input is equivalent to fixed behaviour of algorithm on a uniformly random input. I.e., expected worst-case time of randomized algorithm *on every single input* is $\Theta(n \log n)$.
- In general, randomized algorithms are good when there are many good choices but it is difficult to find one choice that is gaurenteed to be good.

Equality Testing: Monte Carlo and Las Vegas

- Randomized quicksort is a "Las Vegas" algorithm: solution is guaranteed to be correct, but runtime is random (depends on random choices)
- Here is a "Monte Carlo" algorithm: runtime is deterministic, but answer is random (usually one answer is certain and the other is correct with high probability)

- Problem: A has bit-string x and B has bit-string y, and they need to determine whether $x = y$. Measure of interest: how many bits need to be exchange between A and B.
- Isn't this trivial? Just have A send x to B and B do comparison (or the other way around). Communication complexitiy $\in \Theta(n)$. Now suppose x, y represent contents of two 10TB hard drives...
- Idea: Prime p is "witness" of $x \mathrel{!=} y$ if $x \bmod p \mathrel{!=} y \bmod p$ (treating x, y as numbers in binary)
- Algorithm:
  (a) A chooses prime $p \in \{2, \ldots, n^2\}$ uniformly at random, where $n = |x| = |y|$ (so $|p| \leq \log(n^2) = 2\log n$)
  (b) A sends $x \bmod p$ to B (at most $4\log n$ bits)
  (c) B computes $y \bmod p$ and comparies it with $x \bmod p$: output "same" if equal, "different" if not equal

  Note: communication down to $4\log n$ bits, e.g. if $n = 10\text{TB} \leq 2^{50}$ bits, then $4\log_2 n = 200$ bits!
- If B outputs "different", then $x \mathrel{!=} y$ (with certainty); if B outputs "same", then x may or may not be equal to y.
- If $x \mathrel{!=} y$, what is the probability that B answers "same"? $\rightarrow \leq \frac{2\ln n}{n}$. Proof:
- **Prime number theorem**: number of primes in range $[1, m] \approx \frac{m}{\ln m}$. So number of primes in $\{2, \ldots, n^2\} \approx \frac{n^2}{\ln n^2} = \frac{n^2}{2}\ln n$.
- How many primes satisfy $x \bmod p = y$ even though $x \mathrel{!=} y$? At most $n - 1$. How do we know this? $x \bmod p = y \bmod p \Leftrightarrow |x - y|$ is a multiple of p and since $|x - y| < 2^n$ it has no more than $n - 1$ prime divisors. Prove this by assuming it has at least n prime divisors and the smallest one is q.

$$|x - y| > q^n \text{ but } q > 2 \text{ so } |x - y| > 2^n \rightarrow \text{ contradiciton!}$$

- So probability that p is "bad" is at most

$$\frac{n-1}{n^2/(2\ln n)} \leq \frac{2\ln n}{n}$$

  E.g., for $n = 2^{50}$, probability of error $= 6.156\ldots \cdot 10^{-14} = 0.00000000000006156\ldots$
- If this is "too high", rerun the algorithm K times independently. Communication goes up by a factor of K but probability of error goes down to $((2\ln n)/n)^K$ - very small, very quickly! In fact, probability of error quickly becomes smaller than the probability that the computer used to run the algorithm will crash...