CS5330: Assignment for Week 3

Due: Tuesday, 11th Feb 2020.

Please email your solutions to `arnabb@nus.edu.sg` by 11th February, 6:29 pm. I strongly encourage you to write your solutions using LaTeX.

You may discuss the problems with your classmates, though you should write up your solutions on your own. Please note the names of your collaborators in your submission.

1. (Thanks to Seth Gilbert for the question)

   QuickSort is perhaps the most surprising example in the last ten years of experimental evidence leading to significant new theoretical insights and significant improvement. QuickSort was invented in 1959 by Tony Hoare and has served as the standard sorting algorithm for most of computer science ever since. There have been a variety of optimizations since it was first invented (e.g., on how to do the partitioning and how to do the recursion), but the basic algorithm has stayed the same: choose a random pivot, partition the graph, and recurse on the two halves.

   In 2009, a revolution occurred: Vladimir Yaroslavskiy proposed using *two pivots* instead of one. The theorists observed that this change yielded no real improvement: the running time remains $O(n \log n)$ with high probability (and in expectation). Why bother? Yaroslavskiy responded with experiments: two pivots are indeed faster than one!

   This (experimental) discovery led to a much closer analysis of QuickSort, carefully counting the number of comparisons and thinking harder about caching effects. There has been a significant amount of work over the last several years analyzing why two pivots may be better than one, and attempting to understand what is really going on. And it has also had a significant practical impact: multi-pivot QuickSort has now become the standard implementation of QuickSort across a variety of platforms!

   Implement Quicksort and 2-pivot Quicksort as efficient as you can and take some measurements. Think about the right way to implement partitioning, which may have a significant impact on performance.

   The question is very much open-ended. You should investigate (conceptually as well as experimentally) why more pivots help and when they start to hurt. You should also try

experimenting on a variety of datasets, both synthetic as well as real (be creative!), with different types of key elements.

I actually haven't done the implementations. So, I look forward to reading about your experiments.☺.

2. Consider m balls being thrown randomly into n bins, however, the distribution is not uniformly at random. In particular, each ball lands in bin 1 with probability p_1, bin 2 with probability p_2, and so on, where $\mathbf{p} := (p_1, \ldots, p_n)$ is a probability vector with $\sum_{i=1}^{n} p_i = 1$ and $p_i \geq 0$ for all i. What is the smallest m for which you can say that there would be at least one bin with at least 2 balls with probability $> 1/2$?

   Solution sketch: Let X_{ij} be the indicator random variable for the event that balls i and j collide for $1 \leq i < j \leq m$. Then $X_{ij} \sim \text{Ber}(\|p\|_2^2)$. Let us define $X = \sum_{1 \leq i < j \leq m} X_{ij}$. Then we have the following.

$$\mathbb{E}[X] = \binom{m}{2} \|p\|_2^2$$

$$Var(X) \leq O(m^2) \cdot \|p\|_2^2 + O(m^3) \cdot \|p\|_3^3$$

   The bound on the variance uses that $Var(X) \leq \sum_{i<j} \mathbb{E}[X_{ij}^2] + \sum_{i,j,k,\ell : i<j, k<\ell, (i,j) \neq (k,\ell)} Cov(X_{ij}, X_{k\ell})$. Each of the last covariance terms is nonzero only when $|\{i,j,k,\ell\}| = 3$. In that case, we bound $Cov(X_{ij}, X_{i\ell}) \leq \mathbb{E}[X_{ij} X_{i\ell}] = \|p\|_3^3$.

   Now we have the following from Chebyshev, for some constant C.

$$Pr[X > 0] \geq Pr[|X - E[X]| < E[X]] \geq 1 - \frac{1}{Cm^2\|p\|_2^2} - \frac{m^3\|p\|_3^3}{Cm^4\|p\|_2^4}$$

   From the inequality above, there exists a constant $c > 0$ such that for $m > \frac{c}{\|p\|_2}$, we have the probability that at least one bin with at least 2 balls is at least $\frac{1}{2}$.

3. Compute the variance of the number of comparisons made by the QuickSort algorithm. We are looking for the correct order of magnitude and not the exact constants.

   Solution sketch: Let X be the random variable counting the number of comparisons made by the algorithm. Let X_{ij} be the indicator random variable for the event that i^{th} and j^{th} element (in the sorted order) are compared by the algorithm for $1 \leq i < j \leq n$. Then we know that $X_{ij} \sim \text{Ber}(\frac{1}{j-i+1})$. From definitions we have the following.

$$
\begin{aligned}
X &= \sum_{1\le i<j\le n} X_{ij} \\
Var(X) &= \sum_{1\le i<j\le n} Var(X_{ij}) + \sum_{1\le i<j\le n, 1\le k<\ell\le n} Cov(X_{ij}, X_{k\ell}) \\
Var(X) &\le \sum_{1\le i<j\le n} E(X_{ij}) + \sum_{1\le i<j\le n, 1\le k<\ell\le n} Cov(X_{ij}, X_{k\ell}) \\
Var(X) &\le n\log n + \sum_{1\le i<j\le n, 1\le k<\ell\le n} Cov(X_{ij}, X_{k\ell})
\end{aligned}
$$

Now we compute $\mathrm{Cov}(X_{ij}, X_{k\ell})$ for $1 \le i < j \le n, 1 \le k < \ell \le n$ by the following case analysis. Let the sorted array be $A[1 \ldots n]$.

- Case 1: $1 \le i < j < k < \ell \le n$
  $\mathrm{Cov}(X_{ij}, X_{k\ell}) = 0$ since X_{ij} and $X_{k\ell}$ are independent in this case.
- Case 2: $1 \le k < i < \ell < j \le n$

$$
\begin{aligned}
Cov(X_{ij}, X_{k\ell}) &= E[X_{ij}X_{k\ell}] - E[X_{ij}]E[X_{k\ell}] \\
&= Pr[k \text{ is chosen first in } A[k \ldots j]] \cdot Pr[i \text{ or } j \text{ is chosen first in } A[i \ldots j]] \\
&\quad + Pr[k \text{ is chosen first in } A[k \ldots j]] \cdot Pr[k \text{ or } \ell \text{ is chosen first in } A[k \ldots \ell]] \\
&\quad - \frac{4}{(j-i+1)(\ell-k+1)} \\
&= \frac{2(\ell-j+k-i)}{(j-k+1)(j-i+1)(\ell-k+1)} \\
&\le 0
\end{aligned}
$$

- Case 3: $1 \le i < k < j < \ell \le n$
  Following argument along the line same as case 2, we have: $Cov(X_{ij}, X_{k\ell}) \le 0$.
- Case 4: $1 \le k < i < j < \ell \le n$
  Following argument along the line same as case 2, we have: $Cov(X_{ij}, X_{k\ell}) = 0$.
- Case 5: $1 \le i = k < \ell < j$

$$
\begin{aligned}
Cov(X_{ij}, X_{k\ell}) &= E[X_{ij}X_{k\ell}] - E[X_{ij}]E[X_{k\ell}] \\
&= Pr[j \text{ is chosen first in } A[k \ldots j]] \cdot Pr[k \text{ or } \ell \text{ is chosen first in } A[k \ldots \ell]] \\
&\quad + Pr[i \text{ is chosen first in } A[i \ldots j]] \\
&\quad - \frac{4}{(j-i+1)(\ell-k+1)} \\
&= \frac{1}{j-i+1} - \frac{3}{(j-i+1)(\ell-i+1)} \\
&\le \frac{1}{j-i+1}
\end{aligned}
$$

Hence, we have the following.

$$\begin{aligned}\sum_{1\le i<j\le n,1\le k<\ell\le n} Cov(X_{ij},X_{k\ell}) &= \sum_{1\le i<j\le n}\sum_{1\le k<\ell\le n} Cov(X_{ij},X_{k\ell}) \\ &\le \sum_{1\le i<j\le n} 1 \\ &\le n^2\end{aligned}$$

Hence, $Var(X) = O(n^2)$.