CS 388R: Randomized Algorithms, Fall 2021 November 30th 2021

Lecture 26: Review

Prof. Eric Price *Scribe: Anirudh Srinivasan, Sai Surya Duvvuri*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In the last lecture, we covered Randomized Rounding, a technique via which NP-hard problems with an ILP (Integer Linear Programming) solution can be solved approximately. In today' class, we will review the list of topics covered this semester.

2 Concentration Inequalities

We looked at different concentraion inequalities that can be applied to a random variable X which is a sum of n independent random variables $0 \leq X_i \leq 1$.

- Chernoff Bound - $X \leq E[X] + \sigma\sqrt{2\log\frac{T}{\delta}}$ with probability $1-\delta$, $\sigma = \sqrt{n}$
- Bernstein Bound - $X \leq E[X] + \sigma\sqrt{2\log\frac{1}{\delta}} + O(\log(1/\delta))$ with probability $1-\delta$, $\sigma = Var(n)$

The second term in Bernstein bound linearly varies with $\max_i |X_i|$, which in this case is 1.

3 Quick Sort and Quick Select

Quicksort with a random pivot gives us an algorithm that takes $O(n \log n)$ is expected time. This can be proved by looking at the probability that elements i and j are compared ($P[i$ and j compared$] = \frac{2}{j-i+1}$, and summing this up over all i and j. To prove that this happens with high probability, we look at the number of "good" pivot selections that happen. These are when the splits of the array are $> 3/4$ of the original array.

Quickselect works in a method similar to quicksort, where a random pivot is selected, but the recursion happens on only one of the 2 division of the array. The expected runtime for quickselect is $O(n)$. But, Quickselect doesn't have a good bound on error probability, i.e there is good probability that pivot falls in the first n/k elements of the array.

To resolve the above issue, a new algorithm is proposed, where $n^{3/4}$ elements are randomly sampled with replacement from the array and the following property is used: an element in S with rank $r.|S|$ has rank in the original array around $[rn - n\sqrt{\log n/|S|}, rn + n\sqrt{\log n/|S|}]$. This property is used to reduce the search space for finding the median.

4 Balls and Bins and Hashing

We covered different problems in hashing using balls and bins as an example.

- Coupon Collector - Expected and with high probability $O(n \log n)$ to get one of each n variants of the coupon
- Birthday Paradox - Expected $O(\sqrt{n})$ and with high probability $\sqrt{n \log n}$ people to pick before 2 people have the same birthday
- Balls in Bins - with high probability max load across all bins is $O(\frac{\log n}{\log \log n})$
- Balls in Bins with 2 choices - with high probability max load of $O(\log \log n)$
- Cukoo Hashing - with high probability max load of O(1), worst case $O(\log n)$ for insertion.
- Bloom Filters - $n \log(1/\delta)$ bits are used to store n words where δ is false positive rate.
- Limited Independence of Hash Functions - this concept is introduced to have weaker assumptions for analysis when hash functions are used in any algorithm. Specifically, limited independence assumes properties such as k-wise independence which results in preservation of upto k order moments.
- Perfect Hashing - Hash all elements to a primary hash table of size n, and use a secondary hash tables of size $O(n^2)$ for any bin that exceeds a certain size

5 Graph Sparsification and Markov Chains

Given a graph G, we can represent it by a sparse graph H where we sample edges with p_e proportional to the effective resistance such that the spectral norm of H is an ϵ multiplicative factor to the spectral norm of G. This is proved using the matrix Bernstein inequality and the RV lemma. We looked into Markov Chains and properties that make a markov chain ergodic. We also looked into commute time and the hit time between 2 nodes, and how to compute this using R_{eff}, the effective resistance between the 2 nodes.

6 Computational Geometry

We covered randomized algorithms for

- Convex Hull
- Closest Pair
- Planar Point Location

In addition to these, we also covered locality sensitive hashing. This gives us a hashing algorithm where the collision probability increases as the points are closer $(P[h(x) = h(y)] \geq P_1, |x - y| \leq r)$, and decreases when the points are further apart $(P[h(x) = h(y)] \leq P_2, |x - y| \geq Cr)$. Using such functions the query time complexity is n^ρ and space complexity is $n^{1+\rho}$

7 Numerical Linear Algebra

When we need to solve a linear regression problem of the form $AX = b, A \in R^{n \times d}$ with a very large n, we can sample a matrix S to reduce the dimensionality of A down to $\frac{d}{\epsilon^2}$ and then solve $SAX = Sb$ in a much faster way. S can be a simple sparse embedding, sampled from a Gaussian or be obtained using the Fast-Johnson-Lindenstrauss-Transform. We also looked at a technique where we can use SVD to compute a preconditioner R, which reduces the condition number of AR after which we can solve $ARy = b$ using conjugate gradient.

8 Randomized Rounding

When we have a NP hard problem with a solution in the form of an ILP (Integer Linear Program), we can relax the ILP to a fractional LP which is solvable. The solution to the fractional LP can then be rounded up/down, giving a solution to the ILP. This may not be a correct solution though $[Pr(failure) \leq 1/e]$, so the above technique can be repeated multiple times to yield valid solutions, after which the minimum one can be picked.

References

[MR95] Motwani, R. & Raghavan, P. Randomized algorithms. (Cambridge university press,1995)