CS 591, Lecture 20

Jared Saia
University of New Mexico

Class Outline

- Presentations - Good Job!!
- Smoothed Analysis Intro
- Smoothed Analysis of Shortest Path Problem

Smoothed Analysis Intro

Motivation:

- Worst Case Analysis is too hard - typical instances of real-world problems are not at all similar to the worst case
- Average Case Analysis is too easy - a "random" instance of a problem is usually not a typical instance
- Smoothed Analysis is just right - combines the difficulty of worst case analysis with the easiness of average case analysis

Smoothed Analysis

- The smoothed complexity of an algorithm is

$$max_x E_{y \in N_\epsilon(x)} C(y)$$

- x ranges over all inputs, y is a random instance in a neighborhood of x (whose size depends on the smoothing parameter ϵ), $C(y)$ is the cost of running the algorithm on y and E denotes expectation
- As ϵ gets small, the smoothed complexity approaches worst case complexity; as ϵ gets large, smoothed complexity approaches best case complexity

Most Famous Result (so far)

- The smoothed complexity of the simplex algorithm for linear programming is polynomial (even though the worst case run time of the simplex algorithm is exponential!)
- Linear programming is a continuous problem where the input is continuous numbers
- Smoothing operation adds Gaussian noise with parameter ϵ to each number in the input
- Running time is polynomial with the degree of the polynomial depending on $1/\epsilon$

Problems

- Most known results on smoothed analysis are on continuous problems
- Hard to know how to add "noise" to discrete problem
- We will discuss two ideas

Partial Bit Randomization

- Imagine we have a problem which involves the use of integers
- We parameterize the smoothness by an integer k
- For each integer, the last k bits are randomly modified
- We will use this analysis for the single source shortest paths problem

Single Source Shortest Paths

- Given a graph with n vertices and m edges
- Edge weights are in $[0, 2^K - 1]$ (K bit integers)
- Assume that addition, etc of integers can be done in constant time
- Average complexity is known to be $O(n + m)$ (assumes integers are all random)
- Worst case is known to be $O(m + nK)$

Theorem: Smoothed Analysis

Theorem

- Let G be an arbitrary graph, $c : E \longrightarrow [0, \ldots, 2^K - 1]$ be an arbitrary cost function and let k be such that $0 \leq k \leq K$
- Let $\bar{c}$ be obtained from c by making the last k bits of each edge cost random
- Then the single source shortest path problem can be solved in expected time $O(m + n(K - k))$

Proof

- For a node v, let $MinInCost(v)$ be the minimum cost of any incoming edge
- Goldberg has shown that the running time for his algorithm is:

$$O(n + m + \sum_v (K - \log MinInCost(v) + 1)$$

Proof

- Note that $MinInCost(v)$ is the minimum of $d_{in}(v)$ number of which the last k bits are random
- For an edge e, let $r(e)$ be the number of leading zeros in the random part of e.
- Note that $E(r(e)) = 2$. Why?

Proof

Thus we have:

$$
\begin{aligned}
K - \log MinInCost(v) &\leq K - k + \max_{e \in inedges(v)} r(e) && (1)\\
&\leq K - k + \sum_{e \in inedges(v)} r(e) && (2)
\end{aligned}
$$

Thus:

$$E(K - \log MinInCost(v)) \leq K - k + O(d_{in}(v))$$

and if we sum over all n vertices, the time bound follows.

Partial Permutations

- Smoothed Analysis model we'll use for Quicksort is Partial Permutations
- Parameterized by real number $p : 0 \leq p \leq 1$
- Select each element independently with probability p and let m be the number of selected elements
- Take one of the $m!$ permutations of m elements (uniformly at random) and let it act on the selected elements.

QuickSort

- Theorem: Expected number of comparisons of quicksort is $4/p(1 + o(1))n \log_{4/3} n$
- To show this will involve several steps
- 1) We will calculate p_i, the probability that the i-th position is selected and yet unchanged by the permutation of selected elements
- 2) For a fixed element, we will say a call of quicksort is "good" if the subproblem containing the elem has less than 3/4 its original size
- 3) We'll show that the probability a call is good is relatively large (using the p_i's calculated in step 1), so that the total number of expected calls is small

Some Notes

Review of quicksort:

- Choose a pivot element in the list, call it p
- Split the list into $l1$ and $l2$ where $l1$ is all elements less than or equal to p and $l2$ is all elems greater than p
- Recursively sort $l1$ and $l2$
- Return the sorted list $l1, p, l2$

Note: Our version of quicksort just chooses the first element in the list as the pivot element

p_i

- Assume we have a list of x elements.
- Let p_i be the probability that in the "smoothing", the first element is selected and filled with the i-th element
- Note that p_1 is greater than p_i for all $i > 2$
- But all the remaining p_i $(i > 1)$, are equal by symmetry

p_1

$$
\begin{aligned}
p_1 &= p \sum_{0 \le j \le x-1} \binom{x-1}{j} p^j (1-p)^{x-1-j} \frac{1}{j+1} && (3)\\
&= \frac{1}{n} \sum_{0 \le j \le x-1} \binom{x}{j+1} p^{j+1} (1-p)^{x-1-j} && (4)\\
&= \frac{1}{n} \sum_{1 \le j \le x} \binom{x}{j} p^j (1-p)^{x-j} && (5)\\
&= \frac{1}{n}(1-(1-p)^x) && (6)
\end{aligned}
$$

p_i

- By symmetry, $p_2 = p_3 = \cdots = p_x$
- Hence we have

$$
\begin{aligned}
p_i &= (p - p_1)/(x - 1) && (7) \\
&\geq \frac{p - 1/x}{x - 1} && (8)
\end{aligned}
$$

Good Calls

We can now bound the runtime

- Consider a fixed element and say a call is "good" if the subproblem containing the element has less than 3/4 its original size
- How many calls are needed until the elem is in some subproblem of constant size d?
- Number of good calls is bounded by $\log_{4/3} n$!

Good Calls

- Q: What is the probability that a call is good?
- A: A call is good if the pivot (the first element) is among the elements with rank $x/4$ to $3x/4$ in the input list
- There are $x/2$ such elems which would make good pivots
- Each of these elems is the pivot with probability at least p_2
- Hence the call is good with probability at least $(x/2)p_2$ (events that i-th elem chosen as pivot are mutually exclusive so can sum probabilities)

Good Calls

$$
\begin{aligned}
\frac{x}{2}p_2 &\geq \frac{x}{2} \cdot \frac{p - 1/x}{x - 1} && (9) \\
&\geq \frac{p - p/d}{2} && (10) \\
& && (11)
\end{aligned}
$$

- The last equation follows provided that we choose d and bound x such that $p/d \geq 1/x$ is always true
- Let $\pi = \frac{p-p/d}{2}$ be a lower bound on the probability a call is good
- We need $\log_{4/3} n$ good calls

The End Game

- Expected number of calls to get $\log_{4/3} n$ good calls is $\pi^{-1} \log_{4/3} n$
- Hence running time is no more than $n\pi^{-1} \log_{4/3} n + dn$ where dn is the total cost of the small calls
- Choosing the optimum value of d gives a running time of $4/p(1 + o(1))n \log_{4/3} n$

Conclusion

- Smoothed Analysis is a brand new and possibly very useful way to analyze algorithms
- There are still many, many problems it hasen't been tried on yet
- Frequently, The challenge for discrete problems is figuring out how to add in the "noise"
- Keep it in mind the next time you are trying to analyze a "real world" problem

Good luck on Exams and Have a Great Winter Break!!! Enjoy the good ski conditions!!!