Randomized Algorithms II

Advanced Algorithms (CSE 794)
Instructor: Tamal K. Dey
Scribe: Praneeth Uppalapati [*]

May 28, 2009

1 Introduction

A randomized algorithm or probabilistic algorithm is an algorithm which employs a degree of randomness as part of its logic. The algorithm typically uses uniformly random bits as an auxiliary input to guide its behavior, in the hope of achieving good performance in the "average case" over all possible choices of random bits. Formally, the algorithm's performance will be a random variable determined by the random bits; thus either the running time, or the output (or both) are random variables.

The basic approach is:
1. Randomize the algorithm internally.
2. Produce the correct output with a high probability.

2 Preliminaries

A **Random variable** can be thought of as an unknown value that may change every time it is inspected. Thus, a random variable can be thought of as a function mapping the sample space of a random process to the real numbers. More formally, Let (Ω, $\mathcal{F}$, P) be a probability space and (Y, Σ) be a measurable space, then a random variable X is defined as a measurable function X: $\Omega \rightarrow$ Y.

In probability theory and statistics, the **Expected value** (or expectation value, or mathematical expectation, or mean, or first moment) of a random variable is the integral of the random variable with respect to its probability measure. For discrete random variables this is equivalent to the probability-weighted sum of the possible values, and for continuous random variables with a density function it is the probability density -weighted integral of the possible values.

If X is a discrete random variable with probability mass function p(x), then the expected value becomes

$$E(X) = \sum_i x_i \, p(x_i)$$

If the probability distribution of X admits a probability density function f(x), then the expected value can be computed as

[*]Department of Computer and Information Sciences, The Ohio State University, Columbus, Ohio, USA.

$$\mathrm{E(X)} \;=\; \int_{-\infty}^{\infty} x\, \mathrm{f}(x)\ \mathrm{d}\,x\ .$$

The expected value operator (or expectation operator) E is linear in the sense that

$$\begin{aligned} \mathrm{E(X + c)} &= \mathrm{E(X) + c} \\ \mathrm{E(X + Y)} &= \mathrm{E(X) + E(Y)} \\ \mathrm{X\ E(aX)} &= \mathrm{a\ E(X)} \end{aligned}$$

Similarly, if X $= \sum_{i=1}^{n} x_i$ then E(X) $= \sum_{i=1}^{n} \mathrm{E}(x_i)$

Waiting for first success: Suppose we conduct experiments such that the probability of success of the experiment is p. Then, ”Waiting for the first success” is the expected number of trails before we meet success.

If X = # of trails needed to meet success, then

$$p(\mathrm{X} = \mathrm{j}) \;=\; (1 - p)^{j-1}\, p$$

$$\begin{aligned} \mathrm{E(X)} &= \sum_{j=0}^{\infty} j\,(1 - p)^{j-1}\, p \\ &= \frac{p}{1-p} \sum_{j=1}^{\infty} j\,(1 - p)^{j} \;=\; \frac{p}{1-p} \cdot \frac{1-p}{p^2} \;=\; \frac{1}{p} \end{aligned}$$

note :

$$\begin{aligned} \mathrm{x} &= 1(1\text{-}p) + 2(1\text{-}p)^2 + 3(1\text{-}p)^3 + \ldots\ldots\ldots \\ \mathrm{x}(1\text{-}p) &= \qquad\qquad 1(1\text{-}p)^2 + 2(1\text{-}p)^3 + 3(1\text{-}p)^4 + \ldots\ldots\ldots \\ \mathrm{x} - \mathrm{x}(1\text{-}p) &= (1\text{-}p) + (1\text{-}p)^2 + (1\text{-}p)3 + \ldots\ldots\ldots = \frac{1-p}{p} \\ \mathrm{x} &= \frac{1-p}{p^2} \end{aligned}$$

3 The Max 3-SAT Problem

Satisfiability is the problem of determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE.Boolean satisfiability problem (SAT) is a decision problem, whose instance is a Boolean expression written using only AND, OR, NOT, variables, and parentheses.

The 3-SAT problem is defined over a set of boolean variables X = $\{\ x_1, x_2, \ldots., x_n\ \}$ and a set of clauses $\mathcal{C} = c_1, c_2, \ldots., c_k$

where each clause c_i = ($t_{i_1} \ \vee\ t_{i_1} \ \vee\ t_{i_1}$) i.e. 3 variables per clause giving the name 3-SAT and t$_i \in$ (x_i, $\bar{x_i}$)

Problem: Set an assignment to X such that c_i is true $\forall\ i$

The 3-SAT problem is polynomial time reducible to NP-complete , 3-SAT $\leq$ (NP)

So, the Max 3-SAT problem is defined as : Find a truth assignment such that maximum # of clauses become satisfied.

The Randomized approach for this problem is to randomly assign values. This method has proved to give good results most of the time. Let Z denote the random variable equal to the # of satisfied clauses.

$$\begin{aligned} z_i &= 1 \text{ if } c_i \text{ satisfies} \\ &= 0 \text{ otherwise} \end{aligned}$$

$$Z = z_1 + z_2 + \ldots\ldots + z_k$$

$$E(Z) = E(Z_1) + E(Z_2) + \ldots\ldots + E(Z_k)$$

$$E(Z_i) = 1 - \tfrac{1}{2}.\tfrac{1}{2}.\tfrac{1}{2} = \tfrac{7}{8}$$

$$E(Z) = \tfrac{7}{8}\text{k} = \tfrac{7}{8}\text{ optimal} \quad ;\text{ where k} = \text{\# of clauses}$$

Theorem 1. *The Expected number of clauses satisfied by a random assignment is within an approximation factor of $\frac{7}{8}^{th}$ the optimal.*

Corollary *For every instance of 3-SAT, there is a truth assignment that satisfies at least $\frac{7}{8}^{th}$ fraction of the clauses.*

$\Rightarrow$ if k $\leq$ 7, then the 3-SAT is always solvable. So conduct the trails until at least $\frac{7}{8}$k clauses are satisfied.

For j = 0,1,.....,k, let p_j be the probability that a random assignment satisfies exactly j clauses. Then

$$E(Z) = \sum_{j=0}^{k} j\, p_j = \tfrac{7}{8}\text{ k}$$

We are interested in the probability $p = \sum_{j \geq \frac{7}{8}k}^{k} p_j$

$$\tfrac{7}{8}\text{ k} = \sum_{j=0}^{\infty} j\, p_j = \sum_{j<\frac{7}{8}k}^{k} j\, p_j + \sum_{j \geq \frac{7}{8}k}^{k} j\, p_j$$

Let k$'$ be the largest natural number less than $\frac{7}{8}$ k .

$$\tfrac{7}{8}\text{ k} \leq \sum_{j<\frac{7}{8}k}^{k} \text{k}'\, p_j + \sum_{j \geq \frac{7}{8}k}^{k} \text{k}\, p_j = \text{k}'(1-p) + \text{k}p$$

hence, kp $\geq$ $\frac{7}{8}$ k - k$'$.

But $\frac{7}{8}$ k - k$'$ $\geq$ $\frac{1}{8}$ $\Rightarrow p$ $\geq$ $\frac{7}{8}$ k - k$'$ / k = $\frac{1}{8k}$ $\Rightarrow \frac{1}{p}$ $\leq$ 8k

$\therefore$ # of iterations is at most 8k.

$\therefore$ This is a polynomial time algorithm.

Theorem 2. *There is a randomized algorithm with polynomial expected time that is guaranteed to produce a truth assignment satisfying at least $\frac{7}{8}$ of the clauses.*

4 Median Finding Problem

Given a set of "n" unordered numbers we want to find the median of the numbers. Sometimes is also stated as the problem of finding the kth smallest number where k is between 1 and n. A deterministic algorithm in the worst case takes O(n^2) time. So, we define a randomized algorithm where the pivot is choosen randomly. It is proven that it is better than the deterministic algorithm.

Let us divide the process of finding the solution into phases, where each phase j is the time during which the size s of the set under consideration is between $(\frac{3}{4})^{j+1}n \leq s \leq (\frac{3}{4})^{j}n$. Note: there can be more than one iterations in each phase.

Let the Expected time spent in phase j is X$_j$

Suppose at phase j, we prove that expected time is k,   then the total time taken = $\sum_{j=0}^{n}$ k.   If k = $(\frac{3}{4})^{j}n$, then the total time is linear in n.

We define ***central element*** as the splitter that throws away at least $\frac{1}{4}^{th}$ of the elements of the array.
Note: If at any iteration of phase j, the splitter is central , then the phase j ends.

At least half of the elements in a list are central. So, the probability that the splitter is central is $\frac{1}{2}$. Then by the waiting time principle, the Expected number of iterations before a central element is found is 2.

$$E(X_j) = O((\tfrac{3}{4}^{j})\, n) \quad \text{or} \quad 2.C.(\tfrac{3}{4}^{j})\, n$$

$$E(X) = \sum E(X_j) = 2C \sum (\tfrac{3}{4}^{j})\, n = \Theta(n) \quad \text{i.e. linear time in } n$$

5 Concluding remarks

When the model of computation is restricted to Turing machines, it is currently an open question whether the ability to make random choices allows some problems to be solved in polynomial time that cannot be solved in polynomial time without this ability. However, in other contexts, there are specific examples of problems where randomization yields strict improvements.

References

[1] Tamal K Dey, *Advanced Algorithms class, spring 09*, OSU.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Chapter 5: Probabilistic Analysis and Randomized Algorithms, pp.91122.

[3] $http://en.wikipedia.org/wiki/Random\_variable$

[4] $http://en.wikipedia.org/wiki/Boolean\_satisfiability\_problem$