Verification of α-Approximately-Sorted Lists

Randomized Algorithms Final
Spring 2016

Chris Celi
Mark Westerhoff

Contents

1 Introduction

In this paper we present an algorithm that verifies a (potentially large) list of integers is α-approximately-sorted (αAS). As no clear definition to what precisely 'sorted' means, we present one as well.

A list $L = \{a_1, a_2, ..., a_n\}$ is perfectly sorted if for all $i \in [2, n]$, $a_{i-1} \leq a_i$. Consequently a list of size 1 is always sorted so the case where $i = 1$ is ignored. Intuitively this boils down to if each element must be increased to reach the next element (including an increase of zero) then the list is perfectly sorted.

A more interesting case however is when a list is αAS. A list $L = \{b_1, b_2, ..., b_n\}$ is αAS sorted for some α such that $0 \leq \alpha \leq 1$, if for all $i \in [2, n]$, $\Pr[b_{i-1} \leq b_i] \geq \alpha$. Thus a perfectly sorted list is also αAS when $\alpha = 1$. Providing some more intuition, the probability that each element is greater than or equal to element prior is lower bounded by α. We operate with this assumption as a definition for the design of the algorithm.

1.1 Problem Analysis

Simply verifying if a list is αAS is infeasible to do linearly (in accordance with the definition provided) when the list is on the scale of several billion elements. Instead we take a randomized approach to the problem in order to break down the task at hand.

Suppose we are provided a list $L = \{a_1, a_2, ..., a_n\}$ where $n \geq 10^9$, and some α where $0 \leq \alpha \leq 1$. Then we define a collection of $n-1$ variables $X_1, ..., X_{n-1}$ to be a metric for if each sequential pair of terms in the list is sorted. Thus we define X_i as

$$X_i = \begin{cases} 1 & a_{i-1} \leq a_i \\ 0 & \text{otherwise} \end{cases}$$

and as a result if we let $X = \sum X_i$, then X is a metric for the sorted nature of the entire list. This presents an easy way to verify if a list is αAS by having the algorithm return true when $E[X] \geq \lceil \alpha n \rceil$ and false when $E[X] < \lceil \alpha n \rceil$.

Instead of checking each term linearly, we can approximate $E[X]$ using randomized means.

2 Algorithm

We can approximate $E[X]$ using a Monte Carlo method. We sample from X various values X_i which are all easily computed in sub-linear time based on L. This is done by pulling out only the two necessary numbers for each X_i selected and seeing which case 0 or 1 applies. Naturally we are able to sample from X uniformly and at random and do not need to generate the full list to determine any X_i. Using a Monte Carlo method, and a sample of size k, we know that $E[X] = n \sum_{i=1}^{k} \frac{X_i}{k}$ from Motwani and Raghavan [1]. We devise an algorithm to approximate $E[X]$ as α-estimate and compare that value to the provided α.

2.1 Assumptions

A few of assumptions are required concerning the input in order to be allowed some of the theoretical assurances discussed in this paper. They are as follows:

1. The input must be provided as a file where each number is padded (with zeros perhaps) to ensure that each number is the same length. This allows us to use offsets when seeking the file to jump quickly and accurately to specific values of a_i. In other words, access to a_i is desired to be constant time (i.e. a quick offset calculation and jump to location in a file). This is a very strict requirement that directly affects the runtime of the algorithm.

2. The input file must be large. The Monte Carlo method works best on larger input sizes, and so we require that $n \geq 10^9$. In practice, we generate a k sample size which is independent from n. Thus n must be sufficiently large for the algorithm to make sense at all.

3. The desired value for α is known and is relatively close to 1. Later we will derive a sample size based on α which scales best with larger sizes of α. For this algorithm and testing we say that we are looking for an $\alpha \approx .9$. We have not tested the algorithm when α was significantly lower. It is important that the lists tested be relatively close to α sorted as the theory depends on it.

2.2 Specification

The algorithm is as follows. It is important to note that it is an (ϵ, δ)-fully polynomial-time randomized approximation scheme ((ϵ, δ)-FPRAS) like other Monte Carlo algorithms.

Data: A file of n integers to be verified, α, ϵ, and δ.
Result: True if the list is αAS, false otherwise.
load file;
$k = \frac{4}{\epsilon^2 \alpha} \ln \frac{2}{\delta}$;
$sum = 0$;
for *i from 1 to k* **do**
  x = random index of a value pair with replacement;
  v_1, v_2 = get sequential value pair for x;
  if $v_1 \leq v_2$ **then**
    $sum++$;
  end
end
α-estimate = sum $\div$ k;
return (α-estimate $\geq \alpha$);

Algorithm 1: A Monte Carlo-based verification algorithm for αAS lists.

2.3 Runtime Analysis

As we mentioned, with the given assumptions, loading the file and retrieving each pair from the file is considered $O(1)$. Thus the algorithm is trivially $O(k)$ and scales with the size of the sample set used.

3 Theoretical Analysis

Remember that the random variables X_i have the Bernoulli distribution with parameter approximately α. We will say that the true parameter is β. That is, the lowest value of α that would return true for the given list would be β.

Using the Estimator Theorem from Motwani and Raghavan [1], with β being the ratio of sorted elements to total elements, the Monte Carlo method yields an ϵ-approximation to the number of sorted elements with probability at least $1 - \delta$ given a sample size of size

$$k \geq \frac{4}{\epsilon^2 \beta} \ln \frac{2}{\delta}.$$

The result of this theorem is a bound on $E[X]$ as

$$\Pr\left[[(1-\epsilon)\beta n] \leq E[X] \leq [(1+\epsilon)\beta n]\right] \leq 1 - \delta.$$

where βn refers to the true number of sorted pairs.

We can simplify this further by removing the factor of n from each term to yield

$$\Pr\left[[(1-\epsilon)\beta] \leq \alpha \leq [(1+\epsilon)\beta]\right] \leq 1 - \delta.$$

Thus we have a method of estimating the true approximately sorted value of the given list. However, our sample size depends on an unknown value β. From our third assumption we approximate β with α which is known from the specification of the algorithm. A lower α value will lead to a drastically increased value for k increasing the corresponding runtime.

This leaves us with

$$k \geq \frac{4}{\epsilon^2 \alpha} \ln \frac{2}{\delta}$$

where we commonly use $\alpha = .9$ for testing the algorithm. It is rather important to note that k is not dependent on n in any way. Thus this algorithm scales well, especially on larger inputs.

The algorithm produces two-sided error however. Going back to our probabilistic bounds, the probability that the algorithm yields an α-estimate that is within ϵ of β is $1 - \delta$. We can then compare the produced α-estimate to the verification metric (α) provided from the onset.

The algorithm though, needs to verify whether or not a list is αAS. This can be done by requiring a small ϵ to tighten the window for our α-estimate. The α-estimate can also be tightened by boosting the algorithm with multiple iterations. By running the algorithm i times, we see that the probability that α-estimate is within ϵ of β is $1 - \delta^i$. Depending on the relaxation of runtime, the result can be boosted to the desired constant. Even with large δ values, (such as $\delta = .25$) the rate at which $1 - \delta^i$ converges to 1 is very fast. Thus not many iterations are required. This leads to a very confident result rather quickly.

4 Experimental Analysis

sdfsd

5 Conclusion

This algorithm performs surprisingly well for how simple it is in design. By pulling k samples from the n sized list and predicting the β value for the list we end up with a very accurate verification algorithm based on α. It is important to note that the size of k does not scale with n which is a very pleasing result. This allows the size of the list to potentially be unlimited in size.

References

[1] Rajeev Motwani, and Prabhakar Raghavan. *Randomized Algorithms*. 1995. 9th ed. Cambridge University Press, 2007. 311-313. Print.