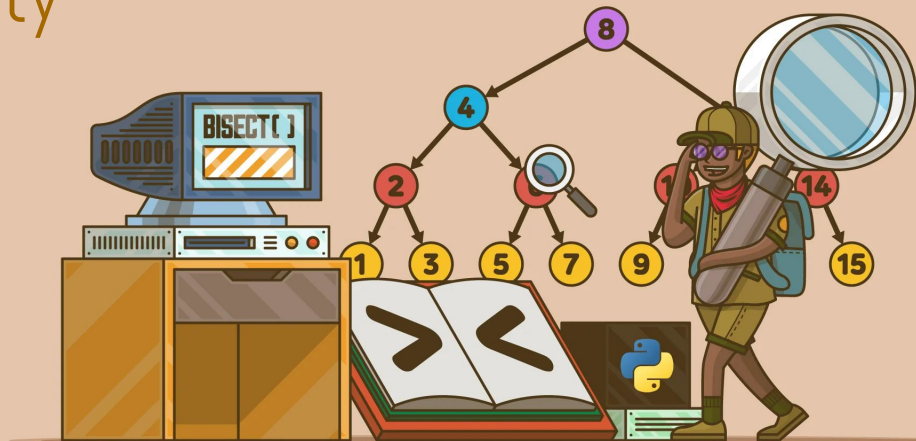


Algorithm Complexity II

Logarithmic Time Complexity

ivanovitch.silva@ufrn.br
@ivanovitchm



Received 17 November 2020; revised 13 April 2021; accepted 17 April 2021. Date of publication 27 April 2021;
date of current version 11 June 2021. The review of this article was arranged by
Associate Editor Eric L. Miller.

Digital Object Identifier 10.1109/OJSP.2021.3075913

A Compressed Sensing Approach to Pooled RT-PCR Testing for COVID-19 Detection

SABYASACHI GHOSH¹, RISHI AGARWAL¹, MOHAMMAD ALI REHAN¹, SHREYA PATHAK¹,
PRATYUSH AGARWAL¹, YASH GUPTA¹, SARTHAK CONSUL², NIMAY GUPTA¹, RITIKA¹, RITESH GOENKA¹,
AJIT RAJWADE¹, AND MANOJ GOPALKRISHNAN²

¹Department of Computer Science and Engineering, IIT Bombay, Mumbai 400076, India

²Department of Electrical Engineering, IIT Bombay, Mumbai 400076, India

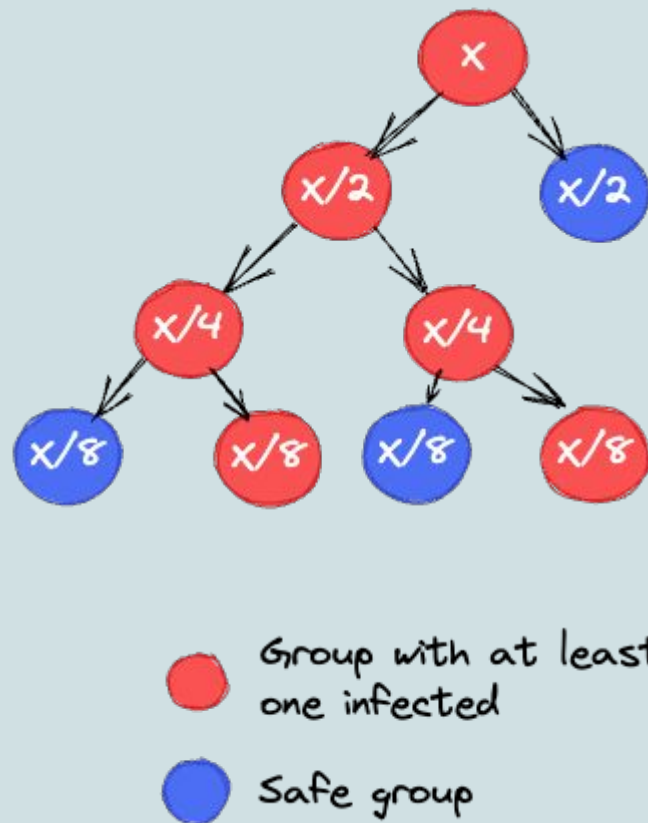
CORRESPONDING AUTHOR: AJIT RAJWADE (e-mail: ajitr@ce.iitb.ac.in).

The work of Ajit Rajwade was supported in part by SERB Matrics under Grant MTR/2019/000691, The work of Ajit Rajwade and Manoj Gopalkrishnan was supported in part by IITB WRCB under Grant #10013976 and in part by the DST-Rakshak under Grant #10013980.

This article has supplementary downloadable material available at <https://doi.org/10.1109/OJSP.2021.3075913>, provided by the authors.

ABSTRACT We propose ‘Tapestry’, a single-round pooled testing method with application to COVID-19 testing using quantitative Reverse Transcription Polymerase Chain Reaction (RT-PCR) that can result in shorter testing time and conservation of reagents and testing kits, at clinically acceptable false positive or false negative rates. Tapestry combines ideas from compressed sensing and combinatorial group testing to create a new kind of algorithm that is very effective in deconvoluting pooled tests. Unlike Boolean group testing algorithms, the input is a quantitative readout from each test and the output is a list of viral loads for each sample relative to the pool with the highest viral load. For guaranteed recovery of k infected samples out of $n \gg k$ being tested, Tapestry needs only $O(k \log n)$ tests with high probability, using random binary pooling matrices. However, we propose deterministic binary pooling matrices based on combinatorial design ideas of Kirkman Triple Systems, which balance between good reconstruction properties and matrix sparsity for ease of pooling while requiring fewer tests in practice. This enables large savings using Tapestry at low prevalence rates while maintaining viability at prevalence rates as high as 9.5%. Empirically we find that single-round Tapestry pooling improves over two-round Dorfman pooling by almost a factor of 2 in the number of tests required. We evaluate Tapestry in simulations with synthetic data obtained using a novel noise model for RT-PCR, and validate it in wet lab experiments with oligomers in quantitative RT-PCR assays. Lastly, we describe use-case scenarios for deployment.

INDEX TERMS Compressed sensing, coronavirus, COVID-19, group testing, Kirkman/Steiner triples, mutual coherence, pooled testing, sensing matrix design.





ORIGINAL RESEARCH

Anti-collision algorithm based on slotted random regressive-style binary search tree in RFID technology

Yibo Ai^{1,2} | Tianrui Bai¹ | Yue Xu¹ | Weidong Zhang^{1,2}

¹ National Center for Materials Service Safety,
University of Science and Technology Beijing, No.
12 Kunlun Road, Beijing, Beijing 100083, China

² Southern Marine Science and Engineering
Guangdong Laboratory (Zhuhai), Zhuhai 519080,
China

Correspondence

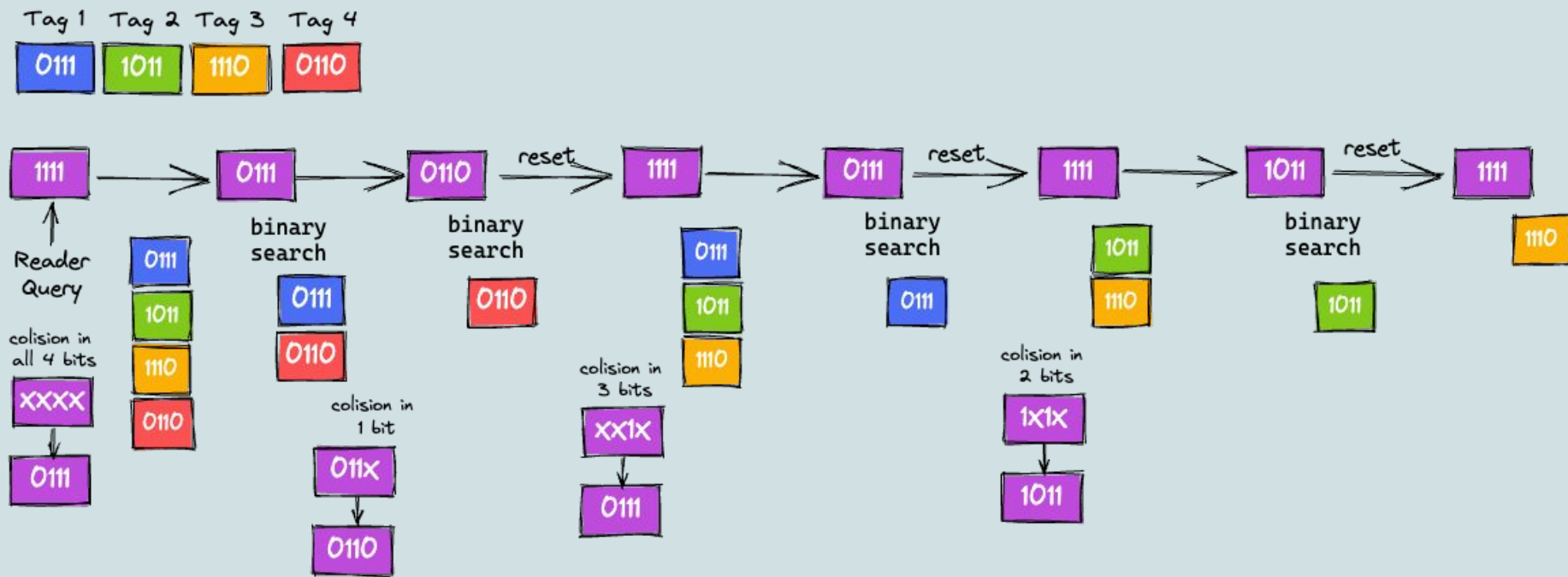
Weidong Zhang, National Center for Materials
Service Safety, University of Science and Technology
Beijing, Beijing 100083, China.
Email: zwd@ustb.edu.cn

Abstract

In recent years, the rapid development of the Internet of Things (IoT) technology has provided a strong technical support for the technological transformation of the logistics industry. The informatization development of logistics industry increasingly relies on the Internet of Things technology represented by Radio Frequency Identification (RFID) technology. These technologies lead the whole business process to optimize the business process in the direction of accurate, efficient and real-time. In order to solve the problem that the reader cannot identify the label information correctly due to the phenomenon of data collision in the application of RFID technology, this paper proposes an anti-collision algorithm based on Slotted Random Regressive-style Binary Search Tree (SR-RBST). Based on Slotted ALOHA (SA) the method proposed in this paper uses the Regressive-style Binary Search Tree (RBST) to process the RFID labels in the collision time slot. With the same size of tags, the SR-RBST algorithm needs less total time slot and has higher efficiency and shorter identification time, while with the increase of the number of tags, the SR-RBST anti-collision algorithm has more obvious advantages. The SR-RBST algorithm effectively improves the time slot utilization efficiency of the system.

How a RFID tag
communicate
with a reader?

Medium Access Control for RFID





The new, enhanced version of the search results page is available. Give the new page a try and share any feedback before it is finalized.

[Try the new version](#)

63 document results

TITLE ("binary search") AND (LIMIT-TO (SRCTYPE , "j")) AND (LIMIT-TO (PUBYEAR , 2022) OR LIMIT-TO (PUBYEAR , 2021) OR LIMIT-TO (PUBYEAR , 2020))

Edit Save Set alert

Search within results...



Refine results

Limit to

Exclude

Open Access



Year



☐ 2022

(20) >

☐ 2021

(15) >

☐ 2020

(20) >

Documents

Secondary documents

Patents

[View Mendeley Data \(45\)](#)

Analyze search results

[Show all abstracts](#) Sort on: [Date \(newest\)](#)



All

CSV export



Download

[View citation overview](#)

[View cited by](#)

[Save to list](#)



Document title

Authors

Year

Source

Cited by



1

A Fast Parallelizable Algorithm for Constructing
Balanced Binary Search Trees
Open Access

Ruzankin, P.S.

2022

SN Computer Science
3(5),367

0

[View abstract](#)



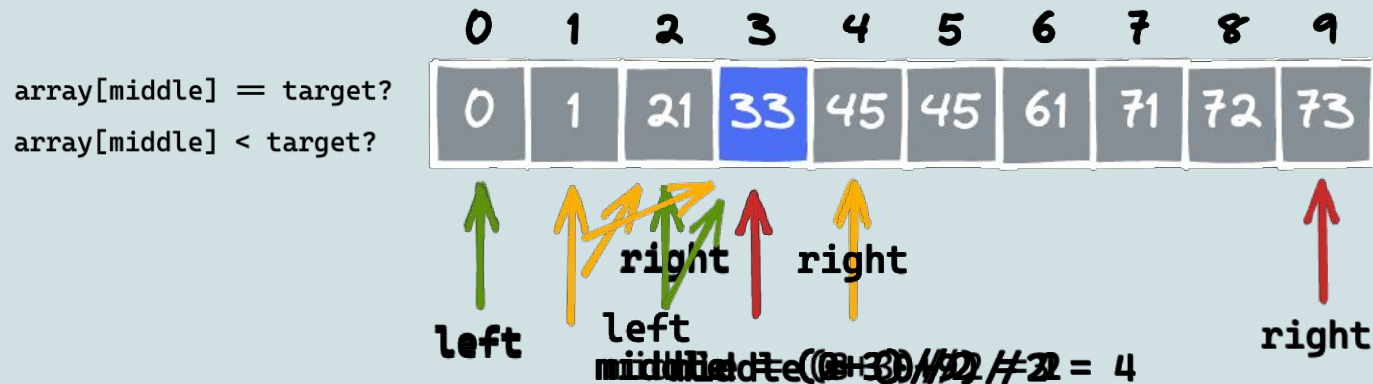
[View at Publisher](#)

[Related documents](#)

Binary Search Algorithm

0	1	2	3	4	5	6	7	8	9
0	1	21	33	45	45	61	71	72	73

Target = 33



Binary Search Algorithm

0	1	2	3	4	5	6	7	8	9
0	1	21	33	45	45	61	71	72	73

Target = 74




```
1: def binarySearch(array, target, left, right):
2:     if (left > right) or (right < left):
3:         return -1
4:     middle = (left + right)//2
5:     potentialMatch = array[middle]
6:     if target == potentialMatch:
7:         return middle
8:     elif target < potentialMatch:
9:         return binarySearch(array, target, left, middle-1)
10:    else:
11:        return binarySearch(array, target, middle+1, right)
```

```
1: def binarySearch(array, target, left, right):
2:     while left <= right:
3:         middle = (left + right)//2
4:         potentialMatch = array[middle]
5:         if target == potentialMatch:
6:             return middle
7:         elif target < potentialMatch:
8:             right = middle - 1
9:         else:
10:            left = middle + 1
11:    return -1
```

How to calculate the time complexity?

$$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{2^2} \rightarrow \frac{N}{2^3} \rightarrow \dots \rightarrow \frac{N}{2^k}$$

In the worst case scenario:

$$2^k = N$$

$$\log 2^k = \log N$$

$$k = \log N$$

$$O(\log N)$$

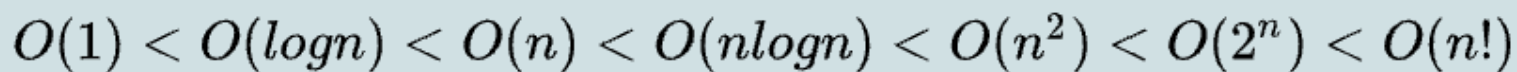
$$\frac{N}{2^k} = 1$$

Binary Search vs Linear Search

$$O(\log N)^*$$

$$O(N)$$

$$O(N \log N) + O(\log N)$$





How to benchmark and compare algorithms?

Are there some python package available?

Manual evaluation using different scenarios? How to guarantee the same input?

How many experiments is necessary to investigate in order to obtain a target confidence interval?