Fast Splitting Algorithms for Noisy and Sparsity-Constrained Group Testing

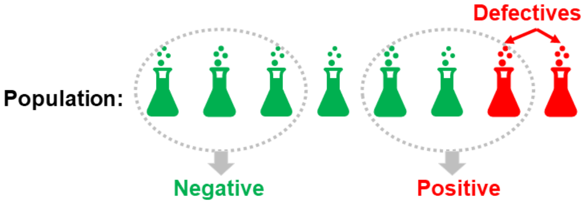
Final Year Project (CP4101)

Nelvin Tan

National University of Singapore (NUS)

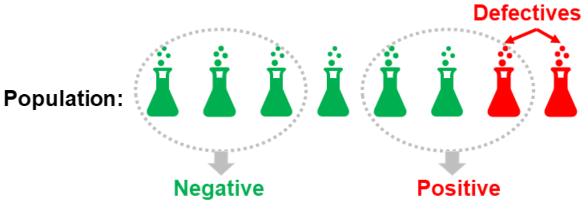
April 2021

Introduction



- **Goal:** Identify a subset of defective items within a larger set of items based on pooled tests.
- Can help to reduce the #tests, which is ideal when tests are costly.
- **Some applications:**
  - Medical testing (e.g., COVID-19)
  - Data science
  - Communication protocols

Introduction



- **Goal:** Identify a subset of defective items within a larger set of items based on pooled tests.
- Can help to reduce the #tests, which is ideal when tests are costly.
- **Some applications:**
  - Medical testing (e.g., COVID-19)
  - Data science
  - Communication protocols

Setup

- Population of n items labelled $\{1, \dots, n\}$.
- Defective set $\mathcal{S} \subset \{1, \dots, n\}$, where $k = |\mathcal{S}| = o(n)$.
- We consider the following settings:
  - **Non-adaptive:** Test pools are designed in advance (makes parallel implementation of the tests more viable).
  - **Noiseless:** Get a +ve test outcome if there is least one defective item, and a -ve outcome if there is no defective item.
  - **For-each recovery:** The algorithm is allowed vanishing error probability, i.e.,

$$\mathbb{P}[\widehat{\mathcal{S}} \neq \mathcal{S}] \to 0 \text{ as } n \to \infty.$$

Setup

- Population of n items labelled $\{1, \ldots, n\}$.
- Defective set $\mathcal{S} \subset \{1, \ldots, n\}$, where $k = |\mathcal{S}| = o(n)$.
- We consider the following settings:
  - **Non-adaptive:** Test pools are designed in advance (makes parallel implementation of the tests more viable).
  - **Noiseless:** Get a $+$ve test outcome if there is least one defective item, and a -ve outcome if there is no defective item.
  - **For-each recovery:** The algorithm is allowed vanishing error probability, i.e.,

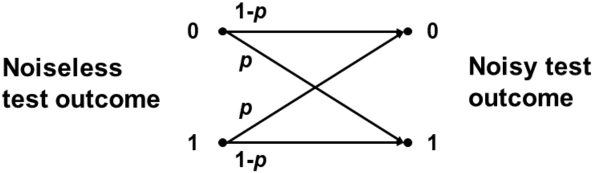
$$\mathbb{P}\big[\widehat{\mathcal{S}} \neq \mathcal{S}\big] \to 0 \text{ as } n \to \infty.$$

Noise Model

Previously, we considered the following constraints under the noiseless setting:

- bounded tests-per-item;
- bounded items-per-test.

In this talk, we study a symmetric noise model (with no sparsity constraints):



+ Our result holds under any asymmetric noise model where $0 \to 1$ and $1 \to 0$ flips both have probability at most constant p (e.g., Z-channel model).

Previous Result

Under the for-each recovery criteria and our noise model, we have:

Reference	Number of tests	Decoding time	Construction
Lower Bound	$\Omega(k \log n)$	–	–
Inan et al.	$O(k \log n)$	$\Omega(n)$	Explicit
Inan et al.	$O(k \log n)$	$O\big(k^3 \cdot \log k + k \log n\big)$	Explicit
NDD	$O(k \log n)$	$\Omega(n)$	Randomized
GROTESQUE	$O(k \cdot \log k \cdot \log n)$	$O\big(k(\log n + \log^2 k)\big)$	Randomized
SAFFRON	$O(k \cdot \log k \cdot \log n)$	$O(k \cdot \log k \cdot \log n)$	Randomized
BMC	$O(k \log n)$	$O(k^2 \cdot \log k \cdot \log n)$	Randomized

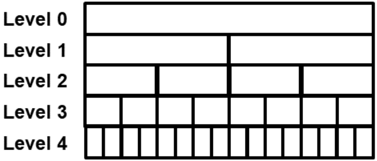
Goal: Design an algorithm that (i) requires $O(k \log n)$ tests, and (ii) has decoding time with a better scaling than BMC.

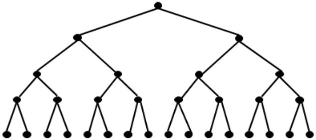
Outline

1. Splitting Technique
2. Noisy Splitting Algorithm
3. Analysis of the Algorithm
4. Summary
5. Addressing Previous Feedback

Splitting Technique

- Start with a tree, where each node is represented by a group of items.
- **Example (binary splitting):**





Splitting Technique (Noiseless Setting)

1. **Testing:** Conduct non-adaptive tests on the nodes.
2. **Decoding (level by level):**
   - Split each node into two nodes of equal sizes if the node's test outcome is $+$ve.
   - Return the set of final level nodes that is reached and appears only in $+$ve tests as $\widehat{S}$.

Example (tests always reveal correct defectivity):

Splitting Technique (Noiseless Setting)

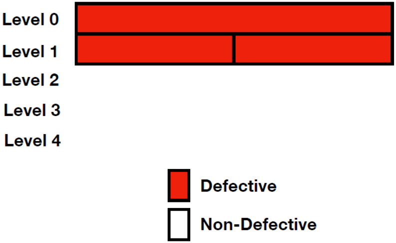
1. **Testing:** Conduct non-adaptive tests on the nodes.
2. **Decoding (level by level):**
   - Split each node into two nodes of equal sizes if the node's test outcome is +ve.
   - Return the set of final level nodes that is reached and appears only in +ve tests as $\widehat{\mathcal{S}}$.

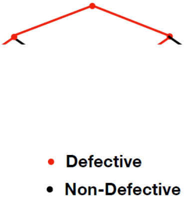
Example (tests always reveal correct defectivity):

Splitting Technique (Noiseless Setting)

1. **Testing:** Conduct non-adaptive tests on the nodes.
2. **Decoding (level by level):**
   - Split each node into two nodes of equal sizes if the node's test outcome is $+$ve.
   - Return the set of final level nodes that is reached and appears only in $+$ve tests as $\widehat{\mathcal{S}}$.

Example (tests always reveal correct defectivity):

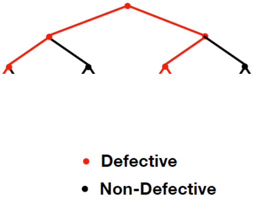




Splitting Technique (Noiseless Setting)

1. **Testing:** Conduct non-adaptive tests on the nodes.
2. **Decoding (level by level):**
   - Split each node into two nodes of equal sizes if the node's test outcome is $+$ve.
   - Return the set of final level nodes that is reached and appears only in $+$ve tests as $\widehat{\mathcal{S}}$.

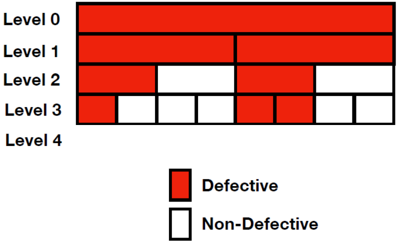
Example (tests always reveal correct defectivity):

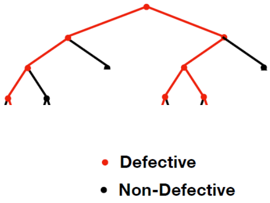


Splitting Technique (Noiseless Setting)

1. **Testing:** Conduct non-adaptive tests on the nodes.
2. **Decoding (level by level):**
   - Split each node into two nodes of equal sizes if the node's test outcome is $+$ve.
   - Return the set of final level nodes that is reached and appears only in $+$ve tests as $\widehat{\mathcal{S}}$.

Example (tests always reveal correct defectivity):

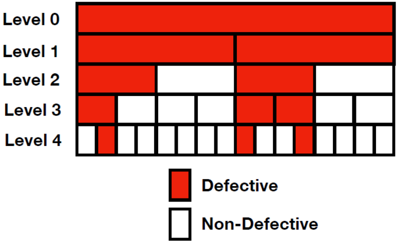


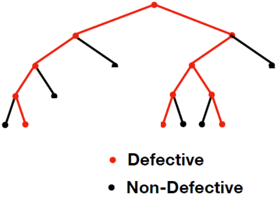


Splitting Technique (Noiseless Setting)

1. **Testing:** Conduct non-adaptive tests on the nodes.
2. **Decoding (level by level):**
   - Split each node into two nodes of equal sizes if the node's test outcome is $+$ve.
   - Return the set of final level nodes that is reached and appears only in $+$ve tests as $\widehat{\mathcal{S}}$.

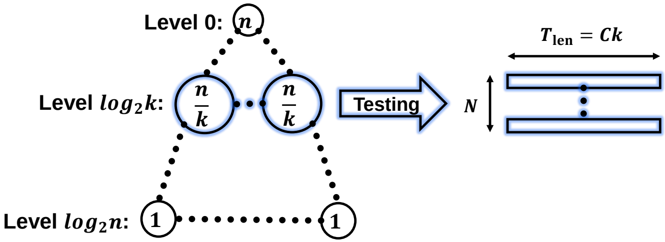
Example (tests always reveal correct defectivity):





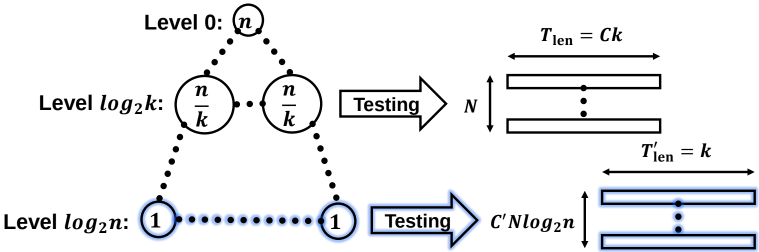
Noisy Algorithm: Testing Procedure

- Conduct testing from level $\log_2 k$ onwards.
- Each node is placed into a single test of a test sequence, chosen uniformly at random.
- Node placements between different test sequences are independent.



Noisy Algorithm: Testing Procedure

- Conduct testing from level $\log_2 k$ onwards.
- Each node is placed into a single test of a test sequence, chosen uniformly at random.
- Node placements between different test sequences are independent.

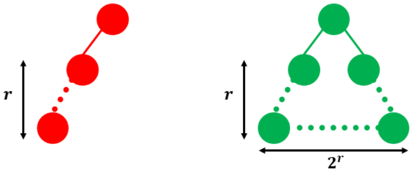


Noisy Algorithm: Decoding Procedure

- **Int. label:** By majority voting of N tests that the node is included in.
  - 1 int. label per node for all testing levels except the final level, which has $C' \log_2 n$ int. labels per node.
- **Final label:** Look at the int. labels of nodes up to r levels below the current node. The final label decides the defectivity of a node.
  - If $\exists$ a path with $> \frac{r}{2}$ +ve int. labels, then assign the final label to be +ve.
  - Otherwise, assign the final label to be -ve.

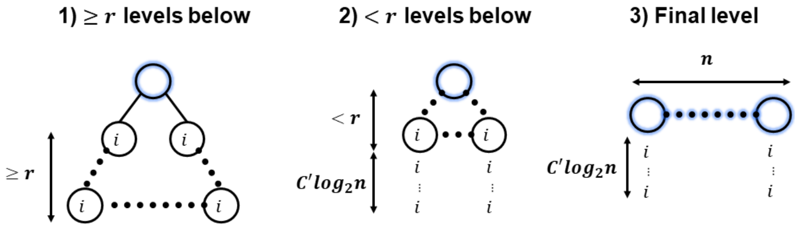
Noisy Algorithm: Decoding Procedure

- **Int. label:** By majority voting of N tests that the node is included in.
  - 1 int. label per node for all testing levels except the final level, which has $C' \log_2 n$ int. labels per node.
- **Final label:** Look at the int. labels of nodes up to r levels below the current node. The final label decides the defectivity of a node.
  - If $\exists$ a path with $> \frac{r}{2}$ +ve int. labels, then assign the final label to be +ve.
  - Otherwise, assign the final label to be -ve.



Noisy Algorithm: Decoding Procedure

- **Int. label:** By majority voting of N tests that the node is included in.
  - 1 int. label per node for all testing levels except the final level, which has $C' \log_2 n$ int. labels per node.
- **Final label:** Look at the int. labels of nodes up to r levels below the current node. The final label decides the defectivity of a node.
  - If $\exists$ a path with $> \frac{r}{2}$ +ve int. labels, then assign the final label to be +ve.
  - Otherwise, assign the final label to be -ve.



Main Result

Theorem

For any constants $\epsilon > 0$ and $t > 0$ satisfying $\epsilon t > 1$, $\exists$ choices of $C, C', N = O(1)$ and $r = O(\log k + \log\log n)$ such that with $O(k \log n)$ tests, our algorithm satisfies the following with probability at least $1 - O\Big(\big(k \log \frac{n}{k}\big)^{1-\epsilon t}\Big)$:

- *The returned estimate $\widehat{\mathcal{S}}$ equals $\mathcal{S}$;*
- *The decoding time is $O\Big(\big(k \log \frac{n}{k}\big)^{1+\epsilon}\Big)$.*

Remarks:

- ϵ and t are new variables that are introduced in the analysis.
- We need $\epsilon t > 1$ to get vanishing error probability.
- We can make ϵ arbitrarily small by choosing a large t.

Main Result

Theorem

For any constants $\epsilon > 0$ *and* $t > 0$ *satisfying* $\epsilon t > 1$, $\exists$ *choices of* $C, C', N = O(1)$ *and* $r = O(\log k + \log\log n)$ *such that with* $O(k \log n)$ *tests, our algorithm satisfies the following with probability at least* $1 - O\Big(\big(k \log \frac{n}{k}\big)^{1-\epsilon t}\Big)$*:*

- *The returned estimate* $\widehat{\mathcal{S}}$ *equals* $\mathcal{S}$*;*
- *The decoding time is* $O\Big(\big(k \log \frac{n}{k}\big)^{1+\epsilon}\Big)$.

Remarks:

- ϵ and t are new variables that are introduced in the analysis.
- We need $\epsilon t > 1$ to get vanishing error probability.
- We can make ϵ arbitrarily small by choosing a large t.

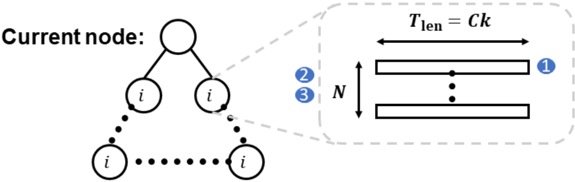
Analysis Outline

What we need to show:

- Low decoding time.
- Probability of wrong defective set output is vanishing.
- Number of tests is small.

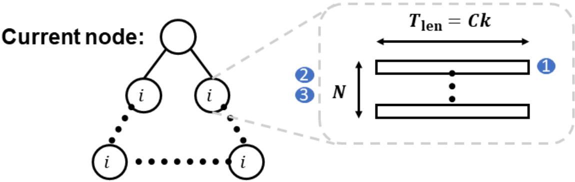
Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

- **Get upper bound on prob. of wrong final label:**
  1. Upper bound prob. of mistake in 1 seq. of tests:
     - Defective node: Only need to consider the noise to get $f_1^d(p, C)$.
     - Non-defective node: Need to consider both noise and the event of being placed with a def. node to get $f_1^{nd}(p, C)$.
  2. By independence between test seq., #tests with mistake is stochastically dominated by $\mathrm{Bin}(N, f_1^d(p, C))$ and $\mathrm{Bin}(N, f_1^{nd}(p, C))$.
  3. By Hoeffding's inequality, we can upper bound prob. of wrong int. label (i.e., $\geq \frac{N}{2}$ mistakes) by $e^{-f_2^d(N,p,C)}$ and $e^{-f_2^{nd}(N,p,C)}$.



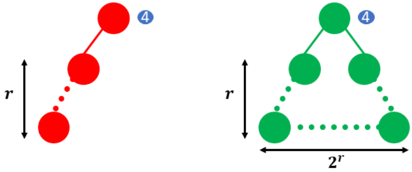
Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

- **Get upper bound on prob. of wrong final label:**
  1. Upper bound prob. of mistake in 1 seq. of tests:
     - Defective node: Only need to consider the noise to get $f_1^d(p, C)$.
     - Non-defective node: Need to consider both noise and the event of being placed with a def. node to get $f_1^{nd}(p, C)$.
  2. By independence between test seq., #tests with mistake is stochastically dominated by $\mathrm{Bin}(N, f_1^d(p, C))$ and $\mathrm{Bin}(N, f_1^{nd}(p, C))$.
  3. By Hoeffding's inequality, we can upper bound prob. of wrong int. label (i.e., $\geq \frac{N}{2}$ mistakes) by $e^{-f_2^d(N,p,C)}$ and $e^{-f_2^{nd}(N,p,C)}$.



Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

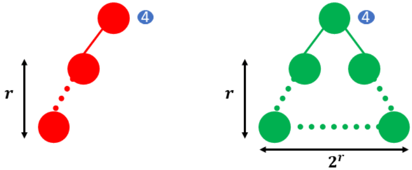
- **Get upper bound on prob. of wrong final label:**
  4. Upper bound prob. of wrong final label mistake:
     - Defective node (mistake: all paths have $\geq \frac{r}{2}$ -ve int. labels): $\binom{r}{r/2}\left(e^{-f_2^d(N,p,C)}\right)^{r/2} \leq 2^r\left(e^{-f_2^d(N,p,C)}\right)^{r/2} = \left(4e^{-f_2^d(N,p,C)}\right)^{r/2}$
     - Non-defective node (mistake: $\exists$ a path with $> \frac{r}{2}$ +ve int. labels): $2^r\left(4e^{-f_2^{nd}(N,p,C)}\right)^{r/2} = \left(16e^{-f_2^{nd}(N,p,C)}\right)^{r/2}$



  5. We introduce t such that for a sufficiently large N (i.e., $N \geq f_3(p, C, t)$), the prob. of wrong final label for both types are bounded above by 2^{-tr}.

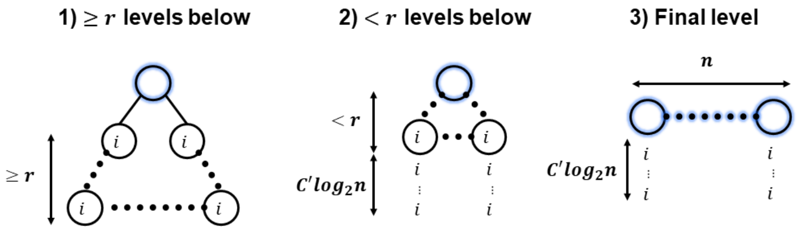
Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

- **Get upper bound on prob. of wrong final label:**
  4. Upper bound prob. of wrong final label mistake:
     - Defective node (mistake: all paths have $\geq \frac{r}{2}$ -ve int. labels): $\binom{r}{r/2}\left(e^{-f_2^d(N,p,C)}\right)^{r/2} \leq 2^r\left(e^{-f_2^d(N,p,C)}\right)^{r/2} = \left(4e^{-f_2^d(N,p,C)}\right)^{r/2}$
     - Non-defective node (mistake: $\exists$ a path with $> \frac{r}{2}$ +ve int. labels): $2^r\left(4e^{-f_2^{nd}(N,p,C)}\right)^{r/2} = \left(16e^{-f_2^{nd}(N,p,C)}\right)^{r/2}$



  5. We introduce t such that for a sufficiently large N (i.e., $N \geq f_3(p, C, t)$), the prob. of wrong final label for both types are bounded above by 2^{-tr}.

Analysis: Levels $\log_2 k$ to $\log_2 n - 1$



- **How about the other cases?**
  - **Case 2:** $p_{\text{final}} \leq 2^{-tr}$ still holds because $\#\text{paths} \leq 2^r$.
  - **Case 3:** Replace r with $C' \log_2 n$ in $p_{\text{final}} \leq 2^{-tr}$. Taking union bound over all n nodes at the final level, the prob. of any mistake at the final level is at most

$$n\left(2^{-tC' \log_2 n}\right) = O\left(n^{1-tC'}\right).$$

Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

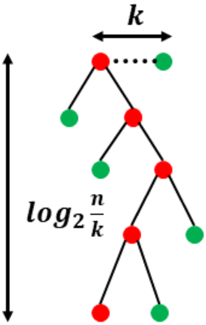
Bound on #nodes explored in the tree:

- Ideally, we don't want to explore too many nodes as it takes time.
- Getting the following 3 kinds of nodes correct implies no further exploration:
  1. nodes at level $\log_2 k$;
  2. all defective nodes below level $\log_2 k$;
  3. children nodes of those defective nodes.
- There are at most $2k \log_2 \left(\frac{n}{k}\right) + k$ of them.
  - At most k defective nodes per level.
  - Each def. node produces ≤ 1 non-def. node.

Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

Bound on #nodes explored in the tree:

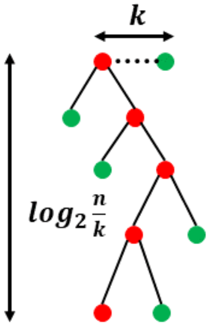
- Ideally, we don't want to explore too many nodes as it takes time.
- Getting the following 3 kinds of nodes correct implies no further exploration:
  1. nodes at level $\log_2 k$;
  2. all defective nodes below level $\log_2 k$;
  3. children nodes of those defective nodes.
- There are at most $2k \log_2 \left(\frac{n}{k}\right) + k$ of them.
  - At most k defective nodes per level.
  - Each def. node produces ≤ 1 non-def. node.



Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

Bound on #nodes explored in the tree:

- Ideally, we don't want to explore too many nodes as it takes time.
- Getting the following 3 kinds of nodes correct implies no further exploration:
  1. nodes at level $\log_2 k$;
  2. all defective nodes below level $\log_2 k$;
  3. children nodes of those defective nodes.
- There are at most $2k \log_2 \left(\frac{n}{k}\right) + k$ of them.
  - At most k defective nodes per level.
  - Each def. node produces ≤ 1 non-def. node.



Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

High prob. bound on #nodes explored in the tree:

- Taking union bound over the 3 kinds of nodes and further upper bounding it by an appropriate decaying function, we get

$$\underbrace{\left(2k\log_2\left(\frac{n}{k}\right)+k\right)}_{\#\text{nodes from the 3 kinds of nodes}}\underbrace{2^{-tr}}_{p_{\text{final}}} \le \underbrace{\left(k\log_2\left(\frac{n}{k}\right)\right)^{1-\epsilon t}}_{\text{decaying function}}.$$

- Choosing $r = \frac{1}{t}\log_2\left(3\left(k\log_2\frac{n}{k}\right)^{\epsilon t}\right)$ satisfies the above condition, i.e., we make no mistakes in labelling the 3 kinds of nodes.
- Hence, we conclude that using our choice of r, we explore at most $2k\log_2\left(\frac{n}{k}\right)+k = O\left(k\log\frac{n}{k}\right)$ nodes with probability $1-o(1)$.

Analysis: Levels $\log_2 k$ to $\log_2 n - 1$

High prob. bound on #nodes explored in the tree:

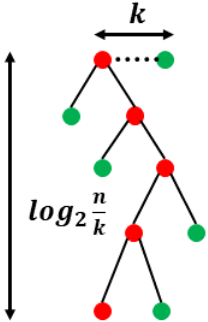
- Taking union bound over the 3 kinds of nodes and further upper bounding it by an appropriate decaying function, we get

$$\underbrace{\left(2k\log_2\left(\frac{n}{k}\right)+k\right)}_{\#\text{nodes from the 3 kinds of nodes}}\underbrace{2^{-tr}}_{p_{\text{final}}} \le \underbrace{\left(k\log_2\left(\frac{n}{k}\right)\right)^{1-\epsilon t}}_{\text{decaying function}}.$$

- Choosing $r = \frac{1}{t}\log_2\left(3\left(k\log_2\frac{n}{k}\right)^{\epsilon t}\right)$ satisfies the above condition, i.e., we make no mistakes in labelling the 3 kinds of nodes.
- Hence, we conclude that using our choice of r, we explore at most $2k\log_2\left(\frac{n}{k}\right)+k = O\left(k\log\frac{n}{k}\right)$ nodes with probability $1-o(1)$.

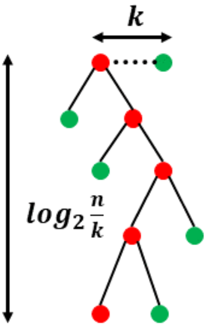
Analysis: Decoding time

- **Decoding time:** Count #test outcome checks.
  - For levels $\log_2 k$ to $\log_2 n - 1$, we explored $O\left(k \log \frac{n}{k}\right)$ nodes, where each node requires at most $\sum_{i=1}^{r} 2^i = O(2^r)$ int. label checks, which further requires $N = O(1)$ test outcome checks. $\#\text{checks} = O\left(\left(k \log \frac{n}{k}\right)^{1+\epsilon}\right)$.
  - At the final level, we have at most $2k$ explored nodes, where each node requires $C' \log_2 n$ int. label checks, which further requires $N = O(1)$ test outcome checks. $\#\text{checks} = O(k \log n)$.
  - Summing the checks in the points above give $O\left(\left(k \log \frac{n}{k}\right)^{1+\epsilon}\right)$.



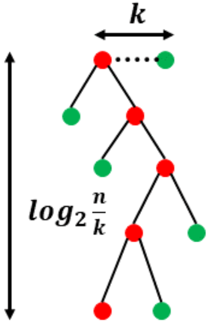
Analysis: Decoding time

- **Decoding time:** Count #test outcome checks.
  - For levels $\log_2 k$ to $\log_2 n - 1$, we explored $O\left(k \log \frac{n}{k}\right)$ nodes, where each node requires at most $\sum_{i=1}^{r} 2^i = O(2^r)$ int. label checks, which further requires $N = O(1)$ test outcome checks. $\#\text{checks} = O\left(\left(k \log \frac{n}{k}\right)^{1+\epsilon}\right)$.
  - At the final level, we have at most $2k$ explored nodes, where each node requires $C' \log_2 n$ int. label checks, which further requires $N = O(1)$ test outcome checks. $\#\text{checks} = O(k \log n)$.
  - Summing the checks in the points above give $O\left(\left(k \log \frac{n}{k}\right)^{1+\epsilon}\right)$.

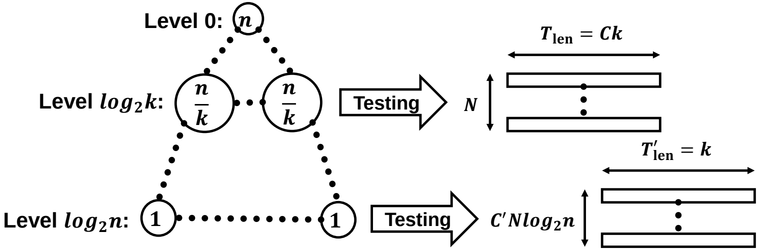


Analysis: Decoding time

- **Decoding time:** Count #test outcome checks.
  - For levels $\log_2 k$ to $\log_2 n - 1$, we explored $O\left(k \log \frac{n}{k}\right)$ nodes, where each node requires at most $\sum_{i=1}^{r} 2^i = O(2^r)$ int. label checks, which further requires $N = O(1)$ test outcome checks. $\#\text{checks} = O\left(\left(k \log \frac{n}{k}\right)^{1+\epsilon}\right)$.
  - At the final level, we have at most $2k$ explored nodes, where each node requires $C' \log_2 n$ int. label checks, which further requires $N = O(1)$ test outcome checks. $\#\text{checks} = O(k \log n)$.
  - Summing the checks in the points above give $O\left(\left(k \log \frac{n}{k}\right)^{1+\epsilon}\right)$.



Number of Tests



- **Number of tests:** At most

$$\underbrace{CNk\log_2\left(\frac{n}{k}\right)}_{\text{levels } \log_2 k, \ldots, \log_2 n-1} + \underbrace{C'Nk\log_2 n}_{\text{final level}} = O(k\log n).$$

Summary

Reference	Number of tests	Decoding time	Construction
Lower Bound	$\Omega\left(k \log \frac{n}{k}\right)$	–	–
Inan et al.	$O(k \log n)$	$\Omega(n)$	Explicit
Inan et al.	$O(k \log n)$	$O\left(k^3 \cdot \log k + k \log n\right)$	Explicit
NDD	$O(k \log n)$	$\Omega(n)$	Randomized
GROTESQUE	$O(k \cdot \log k \cdot \log n)$	$O\left(k(\log n + \log^2 k)\right)$	Randomized
SAFFRON	$O(k \cdot \log k \cdot \log n)$	$O(k \cdot \log k \cdot \log n)$	Randomized
BMC	$O(k \log n)$	$O(k^2 \cdot \log k \cdot \log n)$	Randomized
This talk	$O(k \log n)$	$O\left(\left(k \log \frac{n}{k}\right)^{1+\epsilon}\right)$	Randomized

- Our algorithm (i) uses order-optimal number of tests and (ii) has a near-linear dependence on k in the decoding time.

Jointly-Sparse Group Testing

- **Sparsity constraints:**
  - bounded (at most γ) tests-per-item;
  - bounded (at most ρ) items-per-test.
- **Lower bound:** It seems to be just the max of the 2 settings.
- **Upper bound:**
  - Test designs with double constraints are required.
  - For ρ-setting, it can be showed that a random test design with double constraints is superior to a design with a single constraint.
  - Working towards an optimal algorithm for the ρ-setting can help us understand the more general setting.
  - Some progress made to tighten the lower and upper bounds.

Jointly-Sparse Group Testing

- **Sparsity constraints:**
  - bounded (at most γ) tests-per-item;
  - bounded (at most ρ) items-per-test.
- **Lower bound:** It seems to be just the max of the 2 settings.
- **Upper bound:**
  - Test designs with double constraints are required.
  - For ρ-setting, it can be showed that a random test design with double constraints is superior to a design with a single constraint.
  - Working towards an optimal algorithm for the ρ-setting can help us understand the more general setting.
  - Some progress made to tighten the lower and upper bounds.