Approximate Range Emptiness in Constant Time and Optimal Space

Mayank Goswami, Allan Grønlund, Kasper Larsen, Rasmus Pagh

Max-Planck Institute for Informatics, (MADALGO-Aarhus)², IT University of Copenhagen

SODA 2015, San Diego
Input a set $S$ of $n$ elements from $[U]$. 
Approximate Range Emptiness

- Input a set $S$ of $n$ elements from $[U]$. Preprocess it to answer
- Query: $[a, b]$; is $[a, b] \cap S \neq \emptyset$?
Motivation: Exact versus Approximate Membership

**Membership:** Given a set $S = \{x_1, \cdots, x_n\}$ from a universe $[U]$, preprocess the set to answer membership queries for a queried element $q$ ($q \in S$?).

Minimum space required $B = \log_2(|U|)$ bits. There exist data structures using $B + o(B)$ bits and $O(1)$ query time.

Reduction in space if we only want $\epsilon$-approximate answers? Yes. Bloom Filters $O(n \log(1/\epsilon)$ space, $O(k)$ query. FPR $\epsilon$.

Here $k$ is the number of hash functions used, and depends on $\epsilon$.

Optimal Bloom Filters (Pagh et al.): Query time $O(1)$ irrespective of $\epsilon$ and space usage $(1 + o(1)) n \log(1/\epsilon)$.

\[1\] Currently 4757 citations!
**Membership:** Given a set $S = \{x_1, \cdots, x_n\}$ from a universe $[U]$, preprocess the set to answer membership queries for a queried element $q$ ($q \in S$?).

- Minimum space required $B = \lg \binom{U}{n}$ bits.
- There exist data structures using $B + o(B)$ bits and $O(1)$ query time.

---

1Currently 4757 citations!
Motivation: Exact versus Approximate Membership

- **Membership**: Given a set $S = \{x_1, \ldots, x_n\}$ from a universe $[U]$, preprocess the set to answer membership queries for a queried element $q$ ($q \in S$?).
  - Minimum space required $B = \lg \left( \frac{U}{n} \right)$ bits.
  - There exist data structures using $B + o(B)$ bits and $O(1)$ query time.
  - Reduction in space if we only want $\epsilon$-approximate answers?

---

1Currently 4757 citations!
Motivation: Exact versus Approximate Membership

- **Membership**: Given a set $S = \{x_1, \cdots, x_n\}$ from a universe $[U]$, preprocess the set to answer membership queries for a queried element $q$ ($q \in S$).
  - Minimum space required $B = \lg \left(\frac{U}{n}\right)$ bits.
  - There exist data structures using $B + o(B)$ bits and $O(1)$ query time.
- Reduction in space if we only want $\epsilon$-approximate answers?
  - Yes. **Bloom Filters**\(^1\) $O(n \lg(1/\epsilon))$ space, $O(k)$ query. FPR $\epsilon$.
  - Here $k$ is the number of hash functions used, and depends on $\epsilon$.
  - Optimal Bloom Filters (Pagh et. al.): Query time $O(1)$ irrespective of $\epsilon$ and space usage $(1 + o(1))n \lg(1/\epsilon)$.

\(^1\)Currently 4757 citations!
Approximate Range Emptiness

Range queries are more frequent in real life than membership queries.

- **Range emptiness**: Minimum space required $B = \lg \binom{U}{n}$ bits. Follows from membership.
- Alstrup et. al.: $O(n)$ words = $O(n \lg U)$ bits, $O(k)$ reporting, where $k$ is the number of reported points.
- Can also do emptiness (does there exist a point inside $[a, b]$?) in $O(1)$ time (stop at the first reported point).
Range queries are more frequent in real life than membership queries.

- **Range emptiness**: Minimum space required $B = \lg \binom{U}{n}$ bits. Follows from membership.
- Alstrup et. al.: $O(n)$ words $= O(n \lg U)$ bits, $O(k)$ reporting, where $k$ is the number of reported points.
- Can also do emptiness (does there exist a point inside $[a, b]$?) in $O(1)$ time (stop at the first reported point).

**Approximate range emptiness (ARE)**: False negatives not allowed. A fraction $\epsilon$ of false positives allowed.
- Of all the $u^2/2$ range queries, only an $\epsilon$ fraction may have false positives.
Main Question

Can we reduce space usage for range queries to something lower than $n \lg U$, by requiring approximate answers, similar to membership versus approximate membership queries?
One way to do ARE

- Let us say we want a data structure that answers only to ranges of size at most $L < U$
- One way to do approx. range emptiness query on $[a, b]$ is to
  - Build a Bloom Filter on $S$ with FPR $\epsilon/L$.
  - For every $x \in [a, b]$, run a membership query on the Bloom Filter.
  - By a union bound, the false positive rate is at most $\epsilon$. 

This uses space $n \lg(L/\epsilon)$.

Achieves a query time of $O(r)$, where $r$ is the size of the range.
One way to do ARE

Let us say we want a data structure that answers only to ranges of size at most $L < U$.

One way to do approx. range emptiness query on $[a, b]$ is to:

- Build a Bloom Filter on $S$ with FPR $\epsilon/L$.
- For every $x \in [a, b]$, run a membership query on the Bloom Filter.
- By a union bound, the false positive rate is at most $\epsilon$.

This uses space $n \lg(L/\epsilon)$.

Achieves a query time of $O(r)$, where $r$ is the size of the range.
One way to do ARE

- Let us say we want a data structure that answers only to ranges of size at most $L < U$
- One way to do approx. range emptiness query on $[a, b]$ is to
  - Build a Bloom Filter on $S$ with FPR $\epsilon/L$.
  - For every $x \in [a, b]$, run a membership query on the Bloom Filter.
  - By a union bound, the false positive rate is at most $\epsilon$.
- This uses space $n \lg(L/\epsilon)$.
- Achieves a query time of $O(r)$, where $r$ is the size of the range.
Results: Lower Bounds
Lower Bounds

We first show that the space error tradeoff cannot be improved significantly.

**Theorem**

Any data structure for the ARE problem answering all query intervals of a fixed length $L \leq u/5n$ with false positive rate $\varepsilon > 0$, must use at least

$$s \geq n \lg \left( \frac{L^{1-O(\varepsilon)}}{\varepsilon} \right) - O(n)$$

bits of space.
Extension to Two Sided Errors

**Theorem**

Any data structure for ARE with two sided error rate $\epsilon$ must use

$$s \geq n \log(L/\epsilon) - O(n) \quad \text{bits when } 0 < \epsilon < 1/\log U,$$

$$s = \Omega \left( \frac{n \log(L \log U)}{\log_1/\epsilon \log U} \right) \quad \text{bits when } \frac{1}{\log U} \leq \epsilon \leq \frac{1}{2} - \Omega(1)$$
Results: Upper Bounds
There is a data structure $D_a$ for the ARE problem that
- answers range emptiness for all ranges of length at most $L$,
- uses $n \log (L/\epsilon) + O(n \log^\delta (L/\epsilon))$ bits of space, $\delta$ any desired constant, and
- has a false positive probability at most $\epsilon$.

\[\text{the previous best used } O(n \log U) \text{ bits.}\]
Upper Bounds

- There is a data structure \( D_a \) for the ARE problem that
  - answers range emptiness for all ranges of length at most \( L \),
  - uses \( n \lg(L/\varepsilon) + O(n \lg^\delta (L/\varepsilon)) \) bits of space, \( \delta \) any desired constant, and
  - has a false positive probability at most \( \varepsilon \).

- A data structure \( D_e \) that
  - uses \( n \lg(U/n) + o(n \lg^\delta U/n) \) bits\(^2\),
  - answers exact range reporting in \( O(k) \) and exact emptiness in \( O(1) \) time, respectively.

\(^2\)the previous best used \( O(n \lg U) \) bits.
Upper Bounds: Reduction of Universe

- $f : [U] \rightarrow [R]$, where $R = nL/\epsilon$

On $[R]$ we use the exact range emptiness/reporting data structure. This would give us constant query time in $n \lg(R/n) + n \lg(\delta(R/n))$, or $n \lg(L/\epsilon) + n \lg(\delta(L/\epsilon))$ bits, which would be optimal.

How to construct $f$?

1. Choose $g : [U/R] \rightarrow [R]$ from a pairwise independent family.
2. Pairwise independence: $\Pr[g(x) = y, g(x') = y'] = 1/R^2$ for all $x \neq x'$ in $[U/R]$ and all $y, y'$ in $[R]$.
3. Define $f(x) = (g(\lfloor x/R \rfloor) + x) \mod R$. 

M. Goswami, A. Grønlund, K. Larsen, R. Pag
Upper Bounds: Reduction of Universe

- $f: [U] \rightarrow [R]$, where $R = nL/\epsilon$
- On $[R]$ we use the exact range emptiness/reporting data structure.

Choose $g: \frac{U}{R} \rightarrow [R]$ from a pairwise independent family.

Pairwise independence: $\Pr\{g(x) = y, g(x') = y'\} = \frac{1}{R^2}$ for all $x \neq x'$ in $\frac{U}{R}$ and all $y, y'$ in $[R]$.

Define $f(x) = (g(\lfloor x/R \rfloor) + x) \mod R$. 

Upper Bounds: Reduction of Universe

- $f : [U] \rightarrow [R]$, where $R = nL/\epsilon$
- On $[R]$ we use the exact range emptiness/reporting data structure.
- This would give us constant query time in $n \lg(R/n) + n \lg^\delta(R/n)$, or $n \lg(L/\epsilon) + n \lg^\delta(L/\epsilon)$ bits, which would be optimal.

How to construct $f$?

1. Choose $g : [U/R] \rightarrow [R]$ from a pairwise independent family.
2. Pairwise independence: $\Pr[g(x) = y, g(x') = y'] = 1/R^2$ for all $x \neq x'$ in $[U/R]$ and all $y, y'$ in $[R]$.
3. Define $f(x) = (g(\lfloor x/R \rfloor) + x) \mod R$. 

Upper Bounds: Reduction of Universe

- $f : [U] \rightarrow [R]$, where $R = nL/\epsilon$
- On $[R]$ we use the exact range emptiness/reporting data structure.
- This would give us constant query time in $n \log(R/n) + n \log \delta R/n$, or $n \log(L/\epsilon) + n \log \delta (L/\epsilon)$ bits, which would be optimal.
- How to construct $f$?
Upper Bounds: Reduction of Universe

- \( f : [U] \to [R] \), where \( R = nL/\epsilon \)
- On \([R]\) we use the exact range emptiness/reporting data structure.
- This would give us constant query time in 
  \( n \log(R/n) + n \log^\delta(R/n) \), or 
  \( n \log(L/\epsilon) + n \log^\delta(L/\epsilon) \) bits, which would be optimal.
- How to construct \( f \)?
  1. Choose \( g : [U/R] \to [R] \) from a pairwise independent family.
  2. Pairwise independence: \( \Pr[g(x) = y, g(x') = y'] = 1/R^2 \) for all 
     \( x \neq x' \) in \([U/R]\) and all \( y, y' \) in \([R]\).
Upper Bounds: Reduction of Universe

- \( f : [U] \rightarrow [R] \), where \( R = nL/\epsilon \)
- On \([R]\) we use the exact range emptiness/reporting data structure.
- This would give us constant query time in \( n \log(R/n) + n \log^\delta(R/n) \), or \( n \log(L/\epsilon) + n \log^\delta(L/\epsilon) \) bits, which would be optimal.
- How to construct \( f \)?
  1. Choose \( g : [U/R] \rightarrow [R] \) from a pairwise independent family.
  2. Pairwise independence: \( \Pr[g(x) = y, g(x') = y'] = 1/R^2 \) for all \( x \neq x' \) in \([U/R]\) and all \( y, y' \) in \([R]\).
  3. Define \( f(x) = (g(\lfloor x/R \rfloor) + x) \mod R \).
Upper Bounds: False Positives

- **Lemma:** $\text{Pr}[f(x_1) = f(x_2)] \leq 1/R$.
- Store $f(S) \subseteq [R]$ in an ERE data structure.
Upper Bounds: False Positives

- **Lemma:** $\Pr[f(x_1) = f(x_2)] \leq 1/R$.
- Store $f(S) \subseteq [R]$ in an ERE data structure.
- To answer range query on $[a, b]$, observe that $f([a, b])$ is the union of at most two intervals $l_1, l_2 \subseteq [R]$. 

$\sum_{x \in S} \sum_{y \in I} \Pr[f(x) = f(y)] \leq nL/r \leq \epsilon$. 

M. Goswami, A. Grønlund, K. Larsen, R. Pagh (Max-Planck Institute for Informatics)
Upper Bounds: False Positives

- **Lemma:** $\Pr[f(x_1) = f(x_2)] \leq 1/R$.
- Store $f(S) \subseteq [R]$ in an ERE data structure.
- To answer range query on $[a, b]$, observe that $f([a, b])$ is the union of at most two intervals $I_1, I_2 \subseteq [R]$.
- If either is non-empty in $f(S)$ we report non-empty, else report empty.
Lemma: \( \Pr[f(x_1) = f(x_2)] \leq 1/R. \)

Store \( f(S) \subseteq [R] \) in an ERE data structure.

To answer range query on \([a, b]\), observe that \( f([a, b]) \) is the union of at most two intervals \( I_1, I_2 \subseteq [R] \).

If either is non-empty in \( f(S) \) we report non-empty, else report empty.

No false negatives. False positives occur when \( x \in S \) and \( y \in [a, b] \) collide.

\[
\sum_{x \in S} \sum_{y \in I} \Pr[f(x) = f(y)] \leq nL/r \leq \epsilon.
\]
\( \mathcal{D}_e: \) The ERE Data Structure

- First \( \mathcal{D}_e^* \): Store \( n \) elts. from \([U]\) in \( n \lg U + O(n \lg^\delta U) \) bits, \( \delta > 0 \) any desired constant, and answer queries in constant time.
- Later we will reduce \( U \) above to \( U/n \).
$D_e$: The ERE Data Structure

- First $D_e^*$: Store $n$ elts. from $[U]$ in $n \lg U + O(n \lg^\delta U)$ bits, $\delta > 0$ any desired constant, and answer queries in constant time.
- Later we will reduce $U$ above to $U/n$.
- The data structure $D_e^*$ consists of:
  1. A sorted list of points of $S$. 

M. Goswami, A. Grønlund, K. Larsen, R. Pagh (Max-Planck Institute for Informatics)
Approximate Range Membership
SODA 2015, San Diego
15 / 20
$D_e$: The ERE Data Structure

- First $D_e^*$: Store $n$ elts. from $[U]$ in $n \lg U + O(n \lg \delta U)$ bits, $\delta > 0$ any desired constant, and answer queries in constant time.
- Later we will reduce $U$ above to $U/n$.
- The data structure $D_e^*$ consists of:
  1. A sorted list of points of $S$.
  2. A **weak prefix search** data structure.
$D_e$: The ERE Data Structure

- First $D_e^*$: Store $n$ elts. from $[U]$ in $n \log U + O(n \log^\delta U)$ bits, $\delta > 0$ any desired constant, and answer queries in constant time.
- Later we will reduce $U$ above to $U/n$.
- The data structure $D_e^*$ consists of:
  1. A sorted list of points of $S$.
  2. A **weak prefix search** data structure.

Prefix $p=101$

Answer $= [4, 8]$
Using the weak prefix data structure

Prefix p=101

Answer = [ 4, 8 ]

- Given $[a, b]$, compute the longest common prefix of $a$ and $b$ in $O(1)$ time.
- $h(S) \cap [a, b]$ is non-empty iff:
  1. A largest point in $h(S)$ prefixed by $p \circ 0$ exists, and is not smaller than $a$, or
  2. A smallest point in $h(S)$ prefixed by $p \circ 1$ exists, and is not larger than $b$. 
To reduce \( \log U \) to \( \log(U/n) \) use a standard trick: split \([U]\) into \( n \) subranges \( s_1, \ldots, s_n \) of size \( U/n \)...
ARE data structure

- To reduce $\log U$ to $\log(U/n)$ use a standard trick: split $[U]$ into $n$ subranges $s_1, \ldots, s_n$ of size $U/n$...
- Summarizing:
  - Map elements to a smaller universe.
ARE data structure

- To reduce $\lg U$ to $\lg(U/n)$ use a standard trick: split $[U]$ into $n$ subranges $s_1, \cdots, s_n$ of size $U/n$...
- Summarizing:
  - Map elements to a smaller universe.
  - Use a rank-select on union of top intervals of length $U/n$.
ARE data structure

- To reduce $\log U$ to $\log(U/n)$ use a standard trick: split $[U]$ into $n$ subranges $s_1, \cdots, s_n$ of size $U/n$...
- Summarizing:
  - Map elements to a smaller universe.
  - Use a rank-select on union of top intervals of length $U/n$.
  - Use a weak-prefix search data structure for each interval of size $U/n$. 

There is a data structure $D_a$ for the ARE problem that answers range emptiness for all ranges of length at most $L$, uses $n \log(L/\epsilon) + O(n \log \delta(L/\epsilon))$ bits of space, $\delta$ any desired constant, and has a false positive probability at most $\epsilon$. 

M. Goswami, A. Grønlund, K. Larsen, R. Pagh (Max-Planck Institute for Informatics) 
Approximate Range Membership 
SODA 2015, San Diego
ARE data structure

- To reduce $\log U$ to $\log(U/n)$ use a standard trick: split $[U]$ into $n$ subranges $s_1, \cdots, s_n$ of size $U/n$...

- Summarizing:
  - Map elements to a smaller universe.
  - Use a rank-select on union of top intervals of length $U/n$.
  - Use a weak-prefix search data structure for each interval of size $U/n$.
  - Store some other rank-select structures to locate the individual weak-prefix search data structure.

There is a data structure $D_a$ for the ARE problem that answers range emptiness for all ranges of length at most $L$, uses $n \log(L/\epsilon) + O(n \log(\delta(L/\epsilon)))$ bits of space, $\delta$ any desired constant, and has a false positive probability at most $\epsilon$.

M. Goswami, A. Grønlund, K. Larsen, R. Pagh (Max-Planck Institute for Informatics)
ARE data structure

- To reduce $\lg U$ to $\lg(\frac{U}{n})$ use a standard trick: split $[U]$ into $n$ subranges $s_1, \cdots, s_n$ of size $\frac{U}{n}$...

- Summarizing:
  - Map elements to a smaller universe.
  - Use a rank-select on union of top intervals of length $\frac{U}{n}$.
  - Use a weak-prefix search data structure for each interval of size $\frac{U}{n}$.
  - Store some other rank-select structures to locate the individual weak-prefix search data structure.

- There is a data structure $D_a$ for the ARE problem that
  - answers range emptiness for all ranges of length at most $L$,
  - uses $n \lg (L/\varepsilon) + O(n \lg^\delta (L/\varepsilon))$ bits of space, $\delta$ any desired constant, and
  - has a false positive probability at most $\varepsilon$. 

M. Goswami, A. Grønlund, K. Larsen, R. Pagh (Max-Planck Institute for Informatics)
Approximate Range Membership
SODA 2015, San Diego
Lower bound proof

We will prove the $\epsilon$ one-sided error (no false negatives) version.

- The proof is an encoding argument.
Lower bound proof

We will prove the $\epsilon$ one-sided error (no false negatives) version.

- The proof is an encoding argument.
- Assume a data structure for ARE for ranges of size at most $L$ exists.
- We will use the data structure to encode the set $S$ into a bit string.

The length of this bit string depends on the space usage and false positive rate of the data structure. We know we need $\log(u \cdot n)$ bits; this gives us the lower bound.

For simplicity, we only encode $L$-well separated point sets $S$. 
We will prove the $\epsilon$ one-sided error (no false negatives) version.

- The proof is an encoding argument.
- Assume a data structure for ARE for ranges of size at most $L$ exists.
- We will use the data structure to encode the set $S$ into a bit string.
- The length of this bit string depends on the space usage and false positive rate of the data structure.
- We know we need $\log \binom{u}{n}$ bits; this gives us the lower bound.
Lower bound proof

We will prove the $\epsilon$ one-sided error (no false negatives) version.

- The proof is an encoding argument.
- Assume a data structure for ARE for ranges of size at most $L$ exists.
- We will use the data structure to encode the set $S$ into a bit string.
- The length of this bit string depends on the space usage and false positive rate of the data structure.
- We know we need $\log \binom{u}{n}$ bits; this gives us the lower bound.
- For simplicity, we only encode $L$-well separated point sets $S$. 
A set $S$ is $L$-well separated if:

- $x_{i+1} - x_i \geq 2L$.
- $x_1 \geq 2L - 1$ and $x_n \leq U - 2L$. 

How many $L$-well separated sets are there?

Inductive construction: for the $i$th point, we have at least $U - 4iL - 4(i-1)L = U - 4iL$ choices.

Lemma: There are at least $M = \left(\frac{U - 4nL}{n!}\right)^n$ $L$-well separated sets of size $n$ in a universe of size $U$. Encoding one such set requires $\lg M$ bits.

Size of encoding($s, \epsilon$) $\geq \lg M$ gives the lower bound.
**L-well separated**

- A set $S$ is $L$-well separated if:
  - $x_{i+1} - x_i \geq 2L$.
  - $x_1 \geq 2L - 1$ and $x_n \leq U - 2L$.

- How many $L$-well separated sets are there?
A set \( S \) is \( L \)-well separated if:

- \( x_{i+1} - x_i \geq 2L \).
- \( x_1 \geq 2L - 1 \) and \( x_n \leq U - 2L \).

How many \( L \)-well separated sets are there?

Inductive construction: for the \( i \)th point, we have at least

\[
U - 4L - 4(i - 1)L = U - 4iL \text{ choices.}
\]

**Lemma**

There are at least

\[
M = \frac{(U - 4nL)^n}{n!}
\]

\( L \)-well separated sets of size \( n \) in a universe of size \( U \). Encoding one such set requires \( \lg M \) bits.

Size of encoding \( (s, \epsilon) \geq \lg M \) gives the lower bound.
Conclusion/Open problems

- Disappointing. No space reduction is possible like the Bloom Filter case. Stop looking for upper bounds to the general problem.

Open problems:
- What about the 2D version? Exact range emptiness is well-understood: $O(n \log \log n)$, $O(\log \log n)$ query. Constant query time for approximate version?
- What if the $n$ elements or the queries from $S$ come from a (known/unknown) distribution? Can we save space then? Can we prove a lower bound for this? VLDB paper...

Questions?
Disappointing. No space reduction is possible like the Bloom Filter case. Stop looking for upper bounds to the general problem.

Open problems:

What about the 2D version? Exact range emptiness is well-understood: $O(n \log \log n)$, $O(\log \log n)$ query. Constant query time for approximate version?
Conclusion/Open problems

- Disappointing. No space reduction is possible like the Bloom Filter case. Stop looking for upper bounds to the general problem.

- Open problems:
  - What about the 2D version? Exact range emptiness is well-understood: $O(n \log \log n)$, $O(\log \log n)$ query. Constant query time for approximate version?
  - What if the $n$ elements or the queries from $S$ come from a (known/unknown) distribution? Can we save space then? Can we prove a lower bound for this? VLDB paper..
Conclusion/Open problems

- Disappointing. No space reduction is possible like the Bloom Filter case. Stop looking for upper bounds to the general problem.

- Open problems:
  - What about the 2D version? Exact range emptiness is well-understood: $O(n \log \log n)$, $O(\log \log n)$ query. Constant query time for approximate version?
  - What if the $n$ elements or the queries from $S$ come from a (known/unknown) distribution? Can we save space then? Can we prove a lower bound for this? VLDB paper..

Questions?