Lower Bounds and Algorithms in the Linear Decision Tree Model

AURÉLIEN OOMS
ADVISOR: PROF. JEAN CARDINAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DU MASTER EN SCIENCES INFORMATIQUES
ANNÉE ACADÉMIQUE 2014 - 2015

UNIVERSITÉ LIBRE DE BRUXELLES

1. Introduction

Keywords

- Lower Bounds

1. Introduction

Keywords

- Lower Bounds
- Algorithms

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

Master's thesis contents

- Sorting, Merging and Sorting under Partial Information

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

Master's thesis contents

- Sorting, Merging and Sorting under Partial Information
- Linial's algorithm (Linial [4] + efficient implementation)

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

Master's thesis contents

- Sorting, Merging and Sorting under Partial Information
- Linial's algorithm (Linial [4] + efficient implementation)
- Sorting $X + Y$ (naive approach + Fredman [2])

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

Master's thesis contents

- Sorting, Merging and Sorting under Partial Information
- Linial's algorithm (Linial [4] + efficient implementation)
- Sorting $X + Y$ (naive approach + Fredman [2])
- 3SUM, k-SUM, k-LDT (Grønlund and Pettie [3])

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

Master's thesis contents

- Sorting, Merging and Sorting under Partial Information
- Linial's algorithm (Linial [4] + efficient implementation)
- Sorting $X + Y$ (naive approach + Fredman [2])
- 3SUM, k-SUM, k-LDT (Grønlund and Pettie [3])
- Application of Meiser's Algorithm (Meiser [5])

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

Master's thesis contents

- Sorting, Merging and Sorting under Partial Information
- Linial's algorithm (Linial [4] + efficient implementation)
- Sorting $X + Y$ (naive approach + Fredman [2])
- 3SUM, k-SUM, k-LDT (Grønlund and Pettie [3])
- Application of Meiser's Algorithm (Meiser [5])
- Solve k-SUM using only $o(n)$-linear queries

1. Introduction

Keywords

- Lower Bounds
- Algorithms
- Linear Decision Tree Model

Master's thesis contents

- Sorting, Merging and Sorting under Partial Information
- Linial's algorithm (Linial [4] + efficient implementation)
- Sorting $X + Y$ (naive approach + Fredman [2])
- 3SUM, k-SUM, k-LDT (Grønlund and Pettie [3])
- Application of Meiser's Algorithm (Meiser [5])
- Solve k-SUM using only $o(n)$-linear queries

2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$

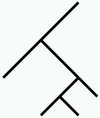
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



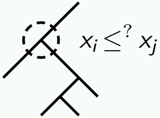
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



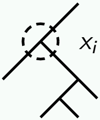
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$x_i - x_j \leq^? 0$

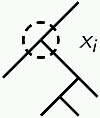
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$x_i - x_j - 3x_k \leq^? 0$

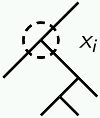
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$x_i - x_j - 3x_k \leq^? 4$

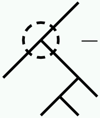
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$-4 + x_i - x_j - 3x_k \leq^? 0$

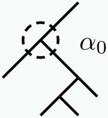
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n \leq^? 0$

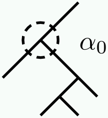
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

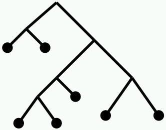
We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n \leq^? 0$



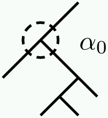
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

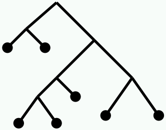
We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n \leq^? 0$$



• = feasible solutions

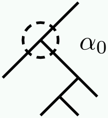
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

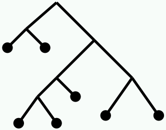
We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n \leq^? 0$$



$\bullet$ = feasible solutions

$\mathrm{ITLB} = \log(\# \bullet)$

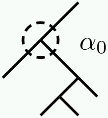
2. Decision Tree Model and ITLB

Definition (Decision Tree Model)

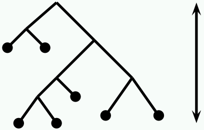
We are allowed to ask questions to an oracle "$\leq$" that are answered "yes" or "no". Each question asked to the oracle costs us a single unit. Every other operation can be carried out for free.

Linear decision tree model and ITLB

input $\mathcal{S} = \{x_1, \ldots, x_n\}$



$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n \leq^? 0$



$h \geq \mathrm{ITLB}$

$\bullet$ = feasible solutions

$\mathrm{ITLB} = \log(\# \bullet)$

3. Efficient Implementation of Linial's Algorithm

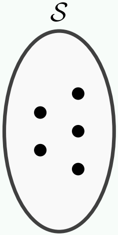
Merging under Partial Information

input $\mathcal{P} = (\mathcal{S} = \{x_1, \ldots, x_N\}, <_{\mathcal{P}})$

3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

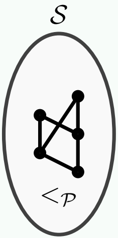
input $\mathcal{P} = (\mathcal{S} = \{x_1, \ldots, x_N\}, <_{\mathcal{P}})$



3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

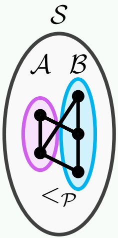
input $\mathcal{P} = (\mathcal{S} = \{x_1, \ldots, x_N\}, <_{\mathcal{P}})$



3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

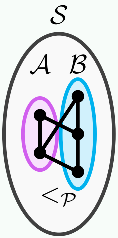
input $\mathcal{P} = (\mathcal{S} = \mathcal{A} \cup \mathcal{B}, <_{\mathcal{P}})$



3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

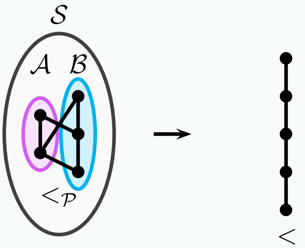
input $\mathcal{P} = (\mathcal{S} = \{a_1 <_\mathcal{P} \cdots <_\mathcal{P} a_m\} \cup \{b_1 <_\mathcal{P} \cdots <_\mathcal{P} b_n\}, <_\mathcal{P})$



3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

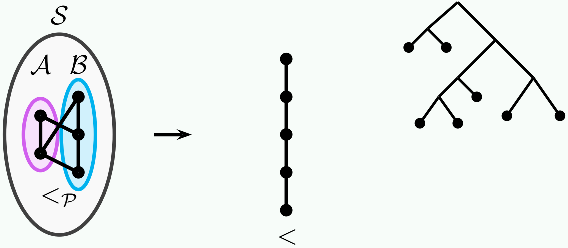
input $\mathcal{P} = (\mathcal{S} = \{a_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} a_m\} \cup \{b_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} b_n\}, <_{\mathcal{P}})$



3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

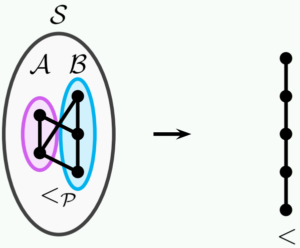
input $\mathcal{P} = (\mathcal{S} = \{a_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} a_m\} \cup \{b_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} b_n\}, <_{\mathcal{P}})$

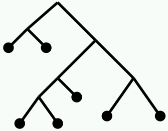


3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

input $\mathcal{P} = (\mathcal{S} = \{a_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} a_m\} \cup \{b_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} b_n\}, <_{\mathcal{P}})$



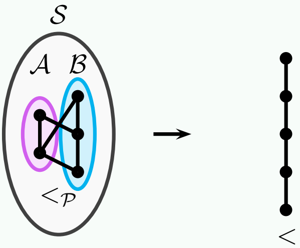


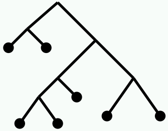
$\# \bullet = e(\mathcal{P}) \leq n!$

3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

input $\mathcal{P} = (\mathcal{S} = \{a_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} a_m\} \cup \{b_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} b_n\}, <_{\mathcal{P}})$





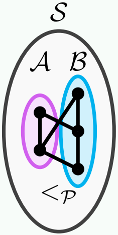
$\# \bullet = e(\mathcal{P}) \leq n!$

$\mathrm{ITLB} = \log e(\mathcal{P})$

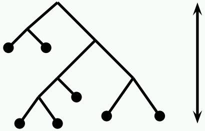
3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

input $\mathcal{P} = (\mathcal{S} = \{a_1 <_\mathcal{P} \cdots <_\mathcal{P} a_m\} \cup \{b_1 <_\mathcal{P} \cdots <_\mathcal{P} b_n\}, <_\mathcal{P})$



$\rightarrow$



$h \geq \log e(\mathcal{P})$

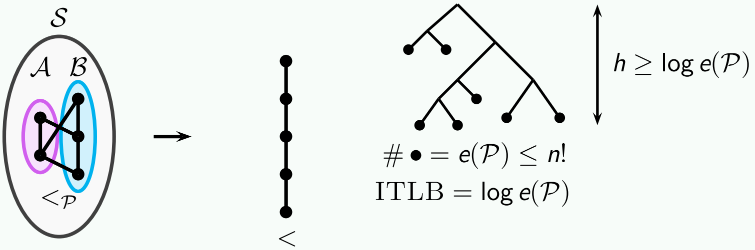
$\# \bullet = e(\mathcal{P}) \leq n!$

$\mathrm{ITLB} = \log e(\mathcal{P})$

3. Efficient Implementation of Linial's Algorithm

Merging under Partial Information

input $\mathcal{P} = (\mathcal{S} = \{a_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} a_m\} \cup \{b_1 <_{\mathcal{P}} \cdots <_{\mathcal{P}} b_n\}, <_{\mathcal{P}})$



Theorem (Linial [4])

Given a poset $\mathcal{P} = (\{x_1, \ldots, x_N\}, <_{\mathcal{P}})$ covered by two chains $\mathcal{A}$ and $\mathcal{B}$, we can always find a query $x_i \overset{?}{<} x_j$ with $x_i \in \mathcal{A}, x_j \in \mathcal{B}$ such that the probability that $x_i < x_j$ lies in the interval $[1/3, 2/3]$.

3. Efficient Implementation of Linial's Algorithm

Using dynamic programming

$$\mathcal{A} = \{a_1 < \cdots < a_m\}, \mathcal{B} = \{b_1 < \cdots < b_n\}, \mathcal{P} = (\mathcal{A} \cup \mathcal{B}, <_{\mathcal{P}})$$

3. Efficient Implementation of Linial's Algorithm

Using dynamic programming

$$\mathcal{A} = \{a_1 < \cdots < a_m\}, \mathcal{B} = \{b_1 < \cdots < b_n\}, \mathcal{P} = (\mathcal{A} \cup \mathcal{B}, <_{\mathcal{P}})$$

$$e(\mathcal{P}) = \begin{cases} 1 & \text{if } |\mathcal{A}| = 0 \text{ or } |\mathcal{B}| = 0 \\ e(\mathcal{P} \setminus \{a_1\}) & \text{if } a_1 <_{\mathcal{P}} b_1 \\ e(\mathcal{P} \setminus \{b_1\}) & \text{if } b_1 <_{\mathcal{P}} a_1 \\ e(\mathcal{P} \setminus \{a_1\}) + e(\mathcal{P} \setminus \{b_1\}) & \text{if } a_1 \text{ and } b_1 \text{ are incomparable in } \mathcal{P} \end{cases}$$

3. Efficient Implementation of Linial's Algorithm

Using dynamic programming

$$\mathcal{A} = \{a_1 < \cdots < a_m\}, \mathcal{B} = \{b_1 < \cdots < b_n\}, \mathcal{P} = (\mathcal{A} \cup \mathcal{B}, <_{\mathcal{P}})$$

$$e(\mathcal{P}) = \begin{cases} 1 & \text{if } |\mathcal{A}| = 0 \text{ or } |\mathcal{B}| = 0 \\ e(\mathcal{P} \setminus \{a_1\}) & \text{if } a_1 <_{\mathcal{P}} b_1 \\ e(\mathcal{P} \setminus \{b_1\}) & \text{if } b_1 <_{\mathcal{P}} a_1 \\ e(\mathcal{P} \setminus \{a_1\}) + e(\mathcal{P} \setminus \{b_1\}) & \text{if } a_1 \text{ and } b_1 \text{ are incomparable in } \mathcal{P} \end{cases}$$

$e(\mathcal{P})$	$e(\mathcal{P} \setminus \{b_1\})$	$e(\mathcal{P} \setminus \{b_1, b_2\})$	$\cdots$	1
$e(\mathcal{P} \setminus \{a_1\})$	$\ddots$			$\vdots$
$e(\mathcal{P} \setminus \{a_1, a_2\})$		$\ddots$		$\vdots$
$\vdots$			$\ddots$	$\vdots$
1	$\cdots$	$\cdots$	$\cdots$	1

3. Efficient Implementation of Linial's Algorithm

Using dynamic programming

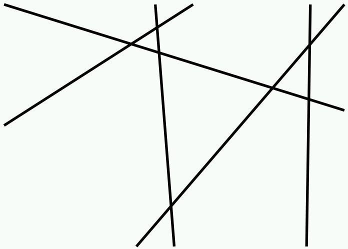
$$\mathcal{A} = \{a_1 < \cdots < a_m\}, \mathcal{B} = \{b_1 < \cdots < b_n\}, \mathcal{P} = (\mathcal{A} \cup \mathcal{B}, <_{\mathcal{P}})$$

$$e(\mathcal{P}) = \begin{cases} 1 & \text{if } |\mathcal{A}| = 0 \text{ or } |\mathcal{B}| = 0 \\ e(\mathcal{P} \setminus \{a_1\}) & \text{if } a_1 <_{\mathcal{P}} b_1 \\ e(\mathcal{P} \setminus \{b_1\}) & \text{if } b_1 <_{\mathcal{P}} a_1 \\ e(\mathcal{P} \setminus \{a_1\}) + e(\mathcal{P} \setminus \{b_1\}) & \text{if } a_1 \text{ and } b_1 \text{ are incomparable in } \mathcal{P} \end{cases}$$

$e(\mathcal{P})$	$e(\mathcal{P}(b_1 < a_1))$	$e(\mathcal{P}(b_2 < a_1))$	$\cdots$	1
$e(\mathcal{P}(a_1 < b_1))$	$\ddots$			$\vdots$
$e(\mathcal{P}(a_2 < b_1))$		$\ddots$		$\vdots$
$\vdots$			$\ddots$	$\vdots$
1	$\cdots$	$\cdots$	$\cdots$	1

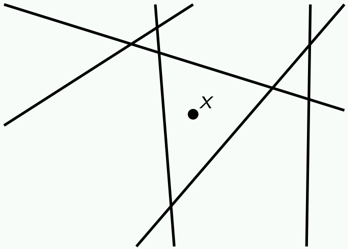
4. Application of Meiser's Algorithm

Point Location in an Arrangement of Hyperplanes



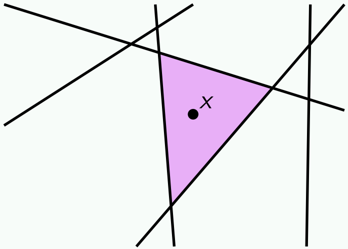
4. Application of Meiser's Algorithm

Point Location in an Arrangement of Hyperplanes



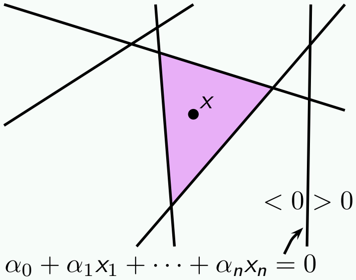
4. Application of Meiser's Algorithm

Point Location in an Arrangement of Hyperplanes



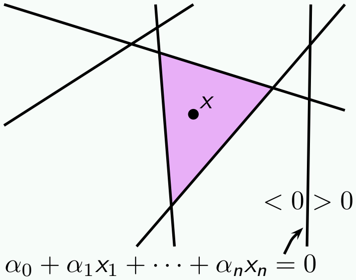
4. Application of Meiser's Algorithm

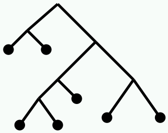
Point Location in an Arrangement of Hyperplanes



4. Application of Meiser's Algorithm

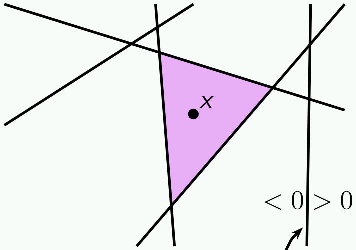
Point Location in an Arrangement of Hyperplanes



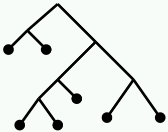


4. Application of Meiser's Algorithm

Point Location in an Arrangement of Hyperplanes



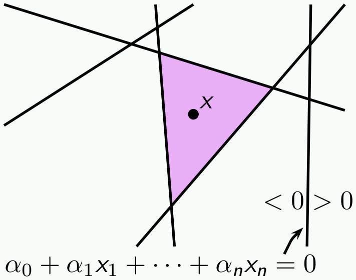
$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n = 0$

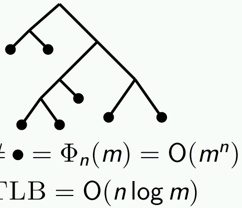


$\# \bullet = \Phi_n(m) = O(m^n)$

4. Application of Meiser's Algorithm

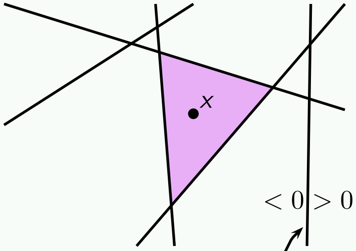
Point Location in an Arrangement of Hyperplanes



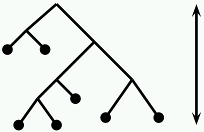


4. Application of Meiser's Algorithm

Point Location in an Arrangement of Hyperplanes



$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n = 0$

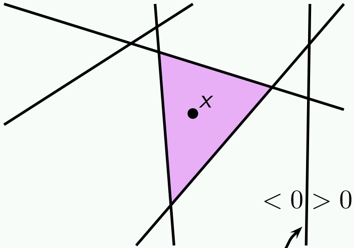


$\# \bullet = \Phi_n(m) = O(m^n)$

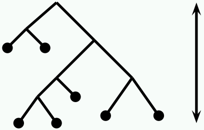
$\mathrm{ITLB} = O(n \log m)$

4. Application of Meiser's Algorithm

Point Location in an Arrangement of Hyperplanes



$$\alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n = 0$$



$h = \Omega(n \log m)$

$\# \bullet = \Phi_n(m) = O(m^n)$

$\mathrm{ITLB} = O(n \log m)$

k-SUM

Given a n-tuple $\mathcal{S} = (x_1, \ldots, x_n)$, $x_i \in \mathbb{R}$, decide whether there exists a k-tuple $(x_{i_1}, \ldots, x_{i_k})$ such that $\sum_{j=1}^{k} x_{i_j} = 0$.

4. Application of Meiser's Algorithm

Algorithm (Idea of the algorithm)

input $x \in \mathbb{R}^n$, the point to be located.

1. Compute the position of x relative to a subset $\mathcal{H}^*$ of $\mathcal{H}$, finding the cell C of $\mathcal{H}^*$ containing x.
2. For any hyperplane H_i not meeting C, deduce $pv_i(x)$ then discard the hyperplane.
3. Recurse on hyperplanes that are left.

4. Application of Meiser's Algorithm

Algorithm (Idea of the algorithm)

input $x \in \mathbb{R}^n$, the point to be located.

1. Compute the position of x relative to a subset $\mathcal{H}^*$ of $\mathcal{H}$, finding the cell C of $\mathcal{H}^*$ containing x.
2. For any hyperplane H_i not meeting C, deduce $pv_i(x)$ then discard the hyperplane.
3. Recurse on hyperplanes that are left.

Theorem (Bürgisser et al. [1])

If we choose $\mathrm{O}(\frac{n^2}{\epsilon} \log^2 \frac{n}{\epsilon})$ hyperplanes uniformly at random from $\mathcal{H}$ and denote this selection $\mathcal{H}^*$ and if there is no hyperplane in $\mathcal{H}^*$ intersecting a given simplex, then, with high probability, the number of hyperplanes of $\mathcal{H}$ intersecting the simplex is less or equal to $\epsilon|\mathcal{H}|$.

4. Application of Meiser's Algorithm

Algorithm (Meiser's algorithm)

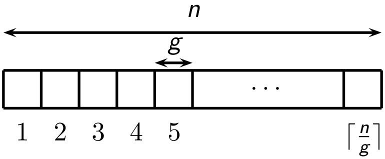
input $x \in \mathbb{R}^n$, the point to be located.

1. Compute the position of x relative to a subset $\mathcal{H}^*$, $|\mathcal{H}^*| = O(n^2 \log^2 n)$, of $\mathcal{H}$, finding the cell C of $\mathcal{H}^*$ containing x.
2. Build simplex S containing x and inscribed in C.
3. For any hyperplane H_i not meeting S, deduce $pv_i(x)$ then discard the hyperplane.
4. Recurse on the hyperplanes that are left.

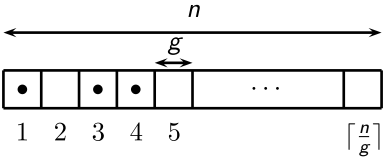
Theorem (Bürgisser et al. [1])

If we choose $O(\frac{n^2}{\epsilon} \log^2 \frac{n}{\epsilon})$ hyperplanes uniformly at random from $\mathcal{H}$ and denote this selection $\mathcal{H}^*$ and if there is no hyperplane in $\mathcal{H}^*$ intersecting a given simplex, then, with high probability, the number of hyperplanes of $\mathcal{H}$ intersecting the simplex is less or equal to $\epsilon|\mathcal{H}|$.

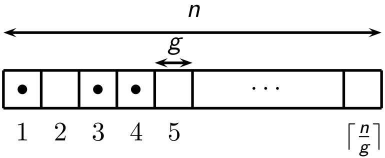
5. Solve k-SUM using only $o(n)$-linear queries



5. Solve k-SUM using only $o(n)$-linear queries



5. Solve k-SUM using only $o(n)$-linear queries



For any g

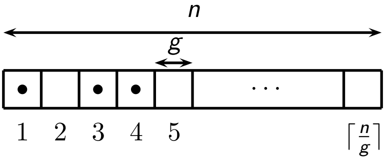
Complexity

$$O\left(\left(\frac{n}{g}\right)^k (kg)^3 \log^3 (kg)\right)$$

Query size

$$kg$$

5. Solve k-SUM using only $o(n)$-linear queries



For $g = n^{\frac{k-1}{k}} = n^{1-\frac{1}{k}}$

Complexity

$$O\left(n(kn^{1-\frac{1}{k}})^3 \log^3(kn^{1-\frac{1}{k}})\right)$$

Query size

$$kn^{1-\frac{1}{k}}$$

6. Conclusion

Conclusion

- Lots of related problems

Conclusion

- Lots of related problems
- Some of them are well studied

Conclusion

- Lots of related problems
- Some of them are well studied
- Others have only weak lower bounds with no matching algorithm

6. Conclusion

Conclusion

- Lots of related problems
- Some of them are well studied
- Others have only weak lower bounds with no matching algorithm
- Still a lot of difficult open problems

6. Conclusion

Conclusion

- Lots of related problems
- Some of them are well studied
- Others have only weak lower bounds with no matching algorithm
- Still a lot of difficult open problems

An Open Question

- Is there an $O(n^{2-\epsilon})$ algorithm for 3SUM?

7. References

[1] Bürgisser, P., Clausen, M., and Shokrollahi, M. A. (1997). *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer.

[2] Fredman, M. L. (1976). How good is the information theory bound in sorting? *Theoretical Computer Science*, 1(4):355–361.

[3] Grønlund, A. and Pettie, S. (2014). Threesomes, degenerates, and love triangles. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 621–630. IEEE.

[4] Linial, N. (1984). The information-theoretic bound is good for merging. *SIAM Journal on Computing*, 13(4):795–801.

[5] Meiser, S. (1993). Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303.