# The Algebraic Computation Tree Model

Mohammad Farshi[1]

Combinatorial and Geometric ALGorithms (CGALG) Lab.,
Department of Computer Science,
Yazd University
http://cs.yazd.ac.ir/cgalg/

1394-1

[1] Thanks to M. Rajaati and M. Alambardar for preliminary version of the slides.

# Outline

1. Algebraic computation trees
2. Algebraic decision trees
3. Lower bounds
   - Linear decision trees
   - The general lower bound
   - Some applications
4. Lower bound of computing spanners
   - A reduction from the element uniqueness problem
   - A lower bound for a set of pairwise distinct points

## Algebraic Computation Tree Model

Algorithms that:

- Exact arithmetics on real numbers
- Use primitive computations (comparison, $+, -, *, /, \sqrt{\ldots}$)

Time complexity: depends on the number of inputs.

# Algebraic computation trees

### Problem $P$:

$n$ : Size of input

$$P : \mathbb{R}^n \rightarrow S$$

$S$ : solution space.

### Example: Sorting $n$ numbers (increasing)

$S$ : all non-decreasing sequences of $n$ numbers

### Definition 3.1.1. An algebraic computation tree

An algebraic computation tree on a sequence $s_1, s_2, \ldots, s_n$ of $n$ variables is a finite tree $T$ in which each node has at most two children, and that satisfies the following three conditions:

# Algebraic computation trees

**Problem $P$:**

$n$ : Size of input

$$P : \mathbb{R}^n \to S$$

$S$ : solution space.

**Example: Sorting $n$ numbers (increasing)**

$S$ : all non-decreasing sequences of $n$ numbers

**Definition 3.1.1. An algebraic computation tree**

An algebraic computation tree on a sequence $s_1, s_2, ..., s_n$ of $n$ variables is a finite tree $T$ in which each node has at most two children, and that satisfies the following three conditions:

# Algebraic computation trees

## Problem $P$:

$n$ : Size of input

$$P : \mathbb{R}^n \to S$$

$S$ : solution space.

## Example: Sorting $n$ numbers (increasing)

$S$ : all non-decreasing sequences of $n$ numbers

## Definition 3.1.1. An algebraic computation tree

An algebraic computation tree on a sequence $s_1, s_2, \ldots, s_n$ of $n$ variables is a finite tree $T$ in which each node has at most two children, and that satisfies the following three conditions:

## 1. leaf

Each leaf is labeled with the combinatorial description, in terms of the variables $s_1, s_2, ..., s_n$, of an element in $S$.

## 2. nodes with one child (computation)

Each node $u$ having one child is labeled with a variable $Z(u)$ and an assignment of the form
(a) $Z(u) := A_1 \& A_2$, where $\& \in \{+, -, *, \backslash\}$, or
(b) $Z(u) := \sqrt{A_1}$
where, for i = 1, 2,
(1) $A_i = Z(u')$ for some proper ancestor $u'$ of $u$ in $T$, or
(2) $A_i \in \{s_1, s_2, ..., s_n\}$, or
(3) $A_i$ is a real number constant.

# Algebraic computation trees

## 1. leaf

Each leaf is labeled with the combinatorial description, in terms of the variables $s_1, s_2, ..., s_n$, of an element in $S$.

## 2. nodes with one child (computation)

Each node $u$ having one child is labeled with a variable $Z(u)$ and an assignment of the form
(a) $Z(u) := A_1 \& A_2$, where $\& \in \{+, -, *, \backslash\}$, or
(b) $Z(u) := \sqrt{A_1}$
where, for i = 1, 2,
(1) $A_i = Z(u')$ for some proper ancestor $u'$ of $u$ in $T$, or
(2) $A_i \in \{s_1, s_2, ..., s_n\}$, or
(3) $A_i$ is a real number constant.

# Algebraic computation trees

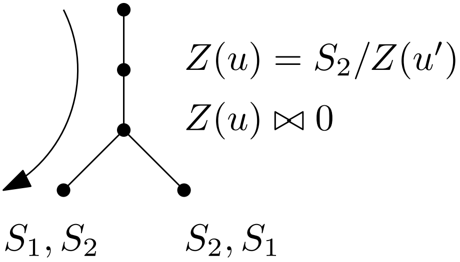
## 3. nodes with two children (comparison)

Each node $u$ having two children is labeled with a comparison of the form $A \bowtie 0$, where
(1) $A = Z(u')$ for some proper ancestor $u'$ of $u$ in $T$, or
(2) $A \in \{s_1, s_2, ..., s_n\}$,
The two outgoing edges leading to the left and right children of u are labeled with $\leq$ and $>$, respectively.

# Algebraic computation trees

## Algebraic computation tree

- An algebraic computation tree $T$ constructed based on an algorithm $A_T$ that solves a computation problem $P : \mathbb{R}^n \to S$.
- For each input, $A_T$ traverses $T$ from root to a leaf.
- Each node of the tree corresponds to the operation that $A_T$ perform at that step.



# Algebraic computation trees

## well-defined algorithm $A_T$

We require that no computation leads to a division by 0 or taking the square root of a negative number

## Algebraic computation tree algorithm

An algorithm $A$ that makes only comparisons and the arithmetic operations $+, -, *, \backslash$ and $\sqrt{}$ is called an algebraic computation-tree algorithm, if there is an algebraic computation tree $T$ such that $A = A_T$

**Definition 3.1.2. solvable in the ACT model**

Let $P : \mathbb{R}^n \to S$ be a computation problem. We say that $P$ is solvable in the algebraic computation-tree model if there exists an algebraic computation-tree $T$ such that, for any $(s_1, s_2, ..., s_n) \in \mathbb{R}^n$ the corresponding algorithm $A_T$ returns the value of $P(s_1, s_2, ..., s_n)$. We say then that $T$ solves the computation problem $P$.

# Algebraic computation trees

**Definition 3.1.3. Time complexity of algorithm $A_T$**

Let $T$ be an algebraic computation-tree. The time complexity of algorithm $A_T$ is defined as the height of the tree $T$.

**Definition 3.1.4. Time complexity of $P$**

Let $P : \mathbb{R}^n \rightarrow S$ be a computation problem that is solvable in the algebraic computation-tree model. The time complexity of $P$ is defined as the minimum height of any algebraic computation-tree that solves $P$.

# Algebraic decision trees

## Section 3.2: Algebraic decision trees

We introduce a restricted type of algebraic computation-trees, called algebraic decision trees, that solve decision problems.

## Algebraic decision trees

An algebraic computation-tree that solves a decision problem is called a algebraic decision tree. This restricted type of decision tree is formally defined by replacing Condition 1 in Definition 3.1.1 by the following condition:

$\ast$ Each leaf is labeled with either YES or NO.

### Section 3.2: Algebraic decision trees

We introduce a restricted type of algebraic computation-trees, called algebraic decision trees, that solve decision problems.

### Algebraic decision trees

An algebraic computation-tree that solves a decision problem is called a algebraic decision tree. This restricted type of decision tree is formally defined by replacing Condition 1 in Definition 3.1.1 by the following condition:
$*$ Each leaf is labeled with either YES or NO.

# Algebraic decision trees

## YES-instances of a problem

Let $P : \mathbb{R}^n \to \{YES, NO\}$ be a decision problem. A point $(s_1, s_2, ..., s_n) \in \mathbb{R}^n$ is called a YES-instance for $P$, if the value of $P(s_1, s_2, ..., s_n)$ is YES.
$V_P$: all YES-instances of $P$.

## The problem corresponding to $V \subseteq \mathbb{R}^n$

$\forall V \subseteq \mathbb{R}^n \quad \exists\, P : \mathbb{R}^n \to \{YES, NO\}$ s.t. $V_P = V$.

Therefore, we can identify each $P$ by its YES-instances.

## Example: Element Uniqueness Problem (EUP)

$$V_P = \left\{ (s_1, \ldots, s_n) \in \mathbb{R}^n \middle| \prod_{1 \le i < j \le n} (s_i - s_j) \neq 0 \right\}.$$

## YES-instances of a problem

Let $P : \mathbb{R}^n \to \{YES, NO\}$ be a decision problem. A point $(s_1, s_2, ..., s_n) \in \mathbb{R}^n$ is called a YES-instance for $P$, if the value of $P(s_1, s_2, ..., s_n)$ is YES.
$V_P$: all YES-instances of $P$.

## The problem corresponding to $V \subseteq \mathbb{R}^n$

$\forall V \subseteq \mathbb{R}^n \quad \exists\, P : \mathbb{R}^n \to \{YES, NO\}$ s.t. $V_P = V$.

Therefore, we can identify each $P$ by its YES-instances.

## Example: Element Uniqueness Problem (EUP)

$$V_P = \left\{ (s_1, \ldots, s_n) \in \mathbb{R}^n \middle| \prod_{1 \le i < j \le n} (s_i - s_j) \neq 0 \right\}.$$

# Algebraic decision trees

## YES-instances of a problem

Let $P : \mathbb{R}^n \to \{YES, NO\}$ be a decision problem. A point $(s_1, s_2, ..., s_n) \in \mathbb{R}^n$ is called a YES-instance for $P$, if the value of $P(s_1, s_2, ..., s_n)$ is YES.
$V_P$: all YES-instances of $P$.

## The problem corresponding to $V \subseteq \mathbb{R}^n$

$\forall V \subseteq \mathbb{R}^n \quad \exists\, P : \mathbb{R}^n \to \{YES, NO\}$ s.t. $V_P = V$.

Therefore, we can identify each $P$ by its YES-instances.

## Example: Element Uniqueness Problem (EUP)

$$V_P = \left\{ (s_1, \ldots, s_n) \in \mathbb{R}^n \middle| \prod_{1 \le i < j \le n} (s_i - s_j) \neq 0 \right\}.$$

### YES-instances of a problem

Let $P : \mathbb{R}^n \rightarrow \{YES, NO\}$ be a decision problem. A point $(s_1, s_2, ..., s_n) \in \mathbb{R}^n$ is called a YES-instance for $P$, if the value of $P(s_1, s_2, ..., s_n)$ is YES.
$V_P$: all YES-instances of $P$.

### The problem corresponding to $V \subseteq \mathbb{R}^n$

$\forall V \subseteq \mathbb{R}^n \quad \exists\, P : \mathbb{R}^n \rightarrow \{YES, NO\}$ s.t. $V_P = V$.

Therefore, we can identify each $P$ by its YES-instances.

### Example: Element Uniqueness Problem (EUP)

$$V_P = \left\{ (s_1, \ldots, s_n) \in \mathbb{R}^n \middle| \prod_{1 \le i < j \le n} (s_i - s_j) \neq 0 \right\}.$$

## decidable in the algebraic decision tree model

A decision problem $P : \mathbb{R}^n \rightarrow \{YES, NO\}$ is called decidable in the algebraic decision tree model, if there exists an algebraic decision tree $T$ such that, for any $(s_1, s_2, ..., s_n) \in \mathbb{R}^n$, the corresponding algorithm $A_T$ returns YES if $(s_1, s_2, ..., s_n) \in V_P$ and NO if $(s_1, s_2, ..., s_n) \notin V_P$
We say then that $T$ decides the decision problem $P$.

## Definition 3.2.1. Time complexity of $P$

Let $V$ be a subset of $\mathbb{R}^n$ and let $P$ be the corresponding decision problem. If $P$ is decidable in the algebraic decision tree model, then we define the time complexity of $V$ as the minimum height of any algebraic decision tree that decides $P$.

## decidable in the algebraic decision tree model

A decision problem $P : \mathbb{R}^n \rightarrow \{YES, NO\}$ is called decidable in the algebraic decision tree model, if there exists an algebraic decision tree $T$ such that, for any $(s_1, s_2, ..., s_n) \in \mathbb{R}^n$, the corresponding algorithm $A_T$ returns YES if $(s_1, s_2, ..., s_n) \in V_P$ and NO if $(s_1, s_2, ..., s_n) \notin V_P$
We say then that $T$ decides the decision problem $P$.

## Definition 3.2.1. Time complexity of $P$

Let $V$ be a subset of $\mathbb{R}^n$ and let $P$ be the corresponding decision problem. If $P$ is decidable in the algebraic decision tree model, then we define the time complexity of $V$ as the minimum height of any algebraic decision tree that decides $P$.

## Section 3.3: Lower bounds for ADT algorithms

- We present a general technique for proving lower bounds on the time complexity for solving decision problems.
- Since a given computation problem often "contains" a related decision problem, this also gives a general approach for proving lower bounds on the time complexity for solving computation problems.
- We will show that the topological structure of a set $V \subseteq \mathbb{R}^n$ yields a lower bound on the time complexity of $V$.

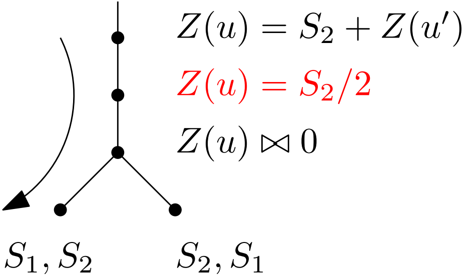
# Linear decision trees (LDT)

## Section 3.3.1: Linear decision trees

This restricted class of algorithms is formally defined by replacing Condition 2 in Definition 3.1.1 by the following condition:

## 2. nodes with one child (computation)

Each node u having one child is labeled with a variable $Z(u) := A_1 \& A_2$,
1. $\& \in \{+, -\}$, (same as before)
2. $\& \in \{*, \backslash\}$, and
(a) $A_1 = Z(u')$ for some proper ancestor $u'$ of $u$ in $T$, or $A_1 \in \{s_1, s_2, ..., s_n\}$, or $A_1$ is a real number constant.
(b) $A_2$ is a real number constant.

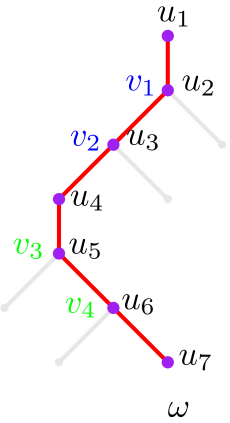



Observe that two input elements (or previously computed values) cannot be multiplied or divided, thus

avoiding nonlinear functions of the input elements

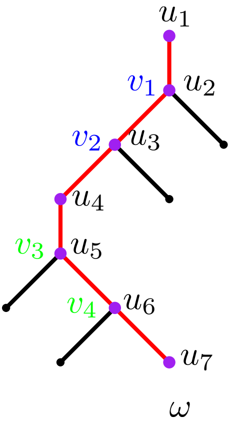
## For any leaf $\omega$ of $T$:

$R(\omega)$ = the set of those inputs on which algorithm $A_T$ terminates in leaf $\omega$.



## For any leaf $\omega$ of $T$:

$R(\omega)$ = the set of those inputs on which algorithm $A_T$ terminates in leaf $\omega$.



### Lemma 3.3.1.

The set $R(\omega)$ is connected, i.e., for any two points $p$ and $q$ in $R(\omega)$, there is a continuous curve in $\mathbb{R}^n$ between $p$ and $q$ that is completely contained in $R(\omega)$.

### Proof:

$R(\omega) = \cap$ half-planes $\Rightarrow$ Convex $\Rightarrow$ Connected.

## Lemma 3.3.1.

The set $R(\omega)$ is connected, i.e., for any two points $p$ and $q$ in $R(\omega)$, there is a continuous curve in $\mathbb{R}^n$ between $p$ and $q$ that is completely contained in $R(\omega)$.

## Proof:

$R(\omega) = \cap$ half-planes $\Rightarrow$ Convex $\Rightarrow$ Connected.

$A, B$ : two distinct connected component of $V$.
$p \in A, q \in B$
$w_p$ : leaf of $T$ that $A_T$ terminates with input $p$.
$w_q$ : leaf of $T$ that $A_T$ terminates with input $q$.

**Lemma 3.3.2.**

The leaves $w_p$ and $w_q$ are distinct.

**Proof.**

- Assume $w_p = w_q = w$.
- $p, q \in R(w)$.
- $R(w)$ is convex, therefore line seqment $pq$ is in $R(w)$.
- This contradict with the assumption that $A$ and $B$ are two distinct connected component.

### Theorem 3.3.3. time complexity in the LDT model

Let $P$ be a decision problem that is decidable in the linear decision tree model and let $V_P \subseteq \mathbb{R}^n$ be the corresponding set of YES-instances. The time complexity of $V_P$ in the linear decision tree model is greater than or equal to $\log(CC(V_P))$.

### Proof.

Let $T$ be an arbitrary linear decision tree that decides $V_P$. By Lemma 3.3.2, $T$ has at least $CC(V_P)$ leaves. Hence, the height of this tree is greater than or equal to $\log(CC(V_P))$.

## Theorem 3.3.3. time complexity in the LDT model

Let $P$ be a decision problem that is decidable in the linear decision tree model and let $V_P \subseteq \mathbb{R}^n$ be the corresponding set of YES-instances. The time complexity of $V_P$ in the linear decision tree model is greater than or equal to $\log(CC(V_P))$.

## Proof.

Let $T$ be an arbitrary linear decision tree that decides $V_P$. By Lemma 3.3.2, $T$ has at least $CC(V_P)$ leaves. Hence, the height of this tree is greater than or equal to $\log(CC(V_P))$.

# Example: EUP

**Theorem 3.3.4.**

The time complexity of the element uniqueness problem for $n$ real numbers in the linear decision tree model is greater than or equal to $n \log n - O(n)$.

**Proof.**

Let $\pi$ and $\rho$ be two distinct permutations of $1, 2, \dots, n$, and consider the points $p := (\pi(1), \pi(2), ..., \pi(n))$ and $r := (\rho(1), \rho(2), ..., \rho(n))$ in $\mathbb{R}^n$. Clearly, both $p$ and $r$ belong to the set $V_P$. We will show that these two points belong to different connected components of $V_P$. This will prove that $CC(V_P) \geq n!$.

# Example: EUP

## Theorem 3.3.4.

The time complexity of the element uniqueness problem for $n$ real numbers in the linear decision tree model is greater than or equal to $n \log n - O(n)$.

## Proof. (Cont.)

- $\pi$ and $\rho$ are distinct $\Rightarrow$ $\exists i, j$ s.t. $\pi(i) < \pi(j)$ and $\rho(i) > \rho(j)$.
- $p$ and $r$ are on different side of hyperplane $x_i = x_j$.
- Any curve between $p$ and $r$ must pass through this hyperplane. $\Rightarrow$ The curve are not included in $V_P$.
- $p$ and $r$ belong to different connected components.
- Done!

The lower bound in LDT Model does not hold in stronger model, i.e. ADT Model.

### Section 3.3.2 The general lower bound

In this section, we will prove that the arguments of Section 3.3.1 can, nevertheless, be generalized. As we will see, the number of connected components of the set $V_P$ of YES-instances still gives a lower bound on the time complexity of the (decidable) decision problem $P$.

### Theorem 3.3.5. (Algebraic Topology)

Let $k$ and $g$ be positive integers, and let $F_1, F_2, \ldots, F_k$ be polynomials in $n$ variables, each having degree less than or equal to $g$. Let

$$W := (x_1, x_2, ..., x_n) \in \mathbb{R}^n : F_i(x_1, x_2, ..., x_n) = 0$$

for all $1 \leq i \leq k$. The set $W$ has at most $g(2g-1)^{(n-1)}$ connected components.

Observe that the upper bound on the number of connected components of the set $W$ depends only on

the number of variables and

the degrees of the polynomials.

It does not depend on

the number of polynomials.

Consider an arbitrary algebraic decision tree $T$ , and let $w$ be any leaf of $T$. Later in this section, we will show that the set $R(w) \subseteq \mathbb{R}^n$ of all inputs on which algorithm $A_T$ terminates in $w$ can be described by a system of polynomial equations and inequalities, each having degree less than or equal to 2.

Our goal is to derive an upper bound on the number of connected components of $R(w)$.
This will be done by transforming

the system of equations and inequalities that describe $R(w)$

into

a system containing polynomial equations only

and then applying Theorem 3.3.5.

**Theorem 3.3.6.**

Let $a, b$ and $c$ be nonnegative integers, and let $E_1, \ldots, E_a, N_1, \ldots, N_b$, $P_1, \ldots, P_c$ be polynomials in $n$ variables, each having degree less than or equal to 2. Let $W$ be the set of all points $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$. such that the following is true:

1. $E_i(x_1, x_2, \ldots, x_n) = 0$ for all $i$ with $1 \leq i \leq a$,
2. $N_i(x_1, x_2, \ldots, x_n) \leq 0$ for all $i$ with $1 \leq i \leq b$, and
3. $P_i(x_1, x_2, \ldots, x_n) > 0$ for all $i$ with $1 \leq i \leq c$.

The set $W$ has at most $3^{n+b+c}$ connected components.

## Proof.

It can be shown that the number $CC(W)$ of connected components of $W$ is finite.
Let $d := CC(W)$. For each $j$ with $1 \leq j \leq d$, let $p_j \in \mathbb{R}^n$ be an arbitrary point in the $j$-th connected component of $W$. Define

$$\epsilon := min\{P_i(p_j) : 1 \leq i \leq c, 1 \leq j \leq d\}$$

Clearly, $\epsilon > 0$. Let $W_\epsilon$ be the set of all points $(x_1, x_2, ..., x_n) \in \mathbb{R}^n$, such that
1. $E_i(x_1, x_2, ..., x_n) = 0$ for all $i$ with $1 \leq i \leq a$,
2. $N_i(x_1, x_2, ..., x_n) \leq 0$ for all $i$ with $1 \leq i \leq b$, and
3. $P_i(x_1, x_2, ..., x_n) > \epsilon$ for all $i$ with $1 \leq i \leq c$.

Then, $W_\epsilon \subseteq W$ and $W_\epsilon$ contains the points $p_1, p_2, ..., p_d$.

## Proof. (Cont.)

We transform the equations and inequalities that define $W_\epsilon$ into a system of polynomial equations by introducing $b+c$ new variables $x_{n+1}, \ldots, x_{n+b+c}$. Let $W_\epsilon$ be the set of all points $(x_1, ..., x_{n+b+c}) \in \mathbb{R}^{n+b+c}$ such that

1. $E_i(x_1, x_2, ..., x_n) = 0$ for all $i$ with $1 \leq i \leq a$,
2. $N_i(x_1, x_2, ..., x_n) + x_{n+i}^2 = 0$ for all $i$ with $1 \leq i \leq b$, and
3. $P_i(x_1, x_2, ..., x_n) - x_{n+b+i}^2 - \epsilon = 0$ for all $i$ with $1 \leq i \leq c$.

The projection of $W'$ onto the first $n$ coordinates is exactly the set $W_\epsilon$, that is,

$$W_\epsilon = \{(x_1, x_2, ..., x_n) : \exists x_1, ..., x_{n+b+c} \in \mathbb{R}, (x_1, ..., x_{n+b+c}) \in W'\}$$

## Proof. (Cont.)

For each $j$ with $1 \leq j \leq d$, let $p'_j$ be a point in $W'$ such that its projection onto the first n coordinates is the point $p_j$. Since the points $p_1, p_2, \ldots, p_d$ are in pairwise distinct connected components of $W$ and since $W' \subseteq W$, it follows that the points $p'_1, p'_2, \ldots, p'_d$ are in pairwise distinct connected components of $W'$ Hence, $CC(W') \geq d$.
The set $W'$ is defined by polynomial equations in $n + b + c$ variables, each having degree less than or equal to 2.
Therefore, by Theorem 3.3.5, we have

$$CC(W') \leq 2 * 3^{n+b+c-1} \leq 3^{n+b+c}$$

Now we are ready to prove the lower bound for ADT algorithms.

## Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P)) - n\log 3}{1+2\log 3}$.

## Proof.

- $T$: ADT that decides $P$. $w$: a leaf of $T$.
- $R(w)$: Inputs that terminate at $w$.
- $u_1, u_2, \ldots, u_{k+1}$: the path from root $u_1$ to $u_{k+1} = w$.
- $r$ : # nodes in this path with one child
- $s$ : # nodes in this path labled by $\sqrt{\ldots}$
- We define $k + s$ polynomial equations and inequalities in variables $x_1, \ldots, x_{n+k}$, ($x_1, \ldots x_n$ for $s_1, \ldots, s_n$ and $x_{n+1}, \ldots, x_{n+k}$ for $u_1, \ldots, u_k$)

### Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P)) - n\log 3}{1+2\log 3}$.

### Proof. (Cont.)

For $1 \leq i \leq k$, consider node $u_i$:
**Case 1:** $u_i$ has one child (computation node)

- We add one equation and probably an inequality to our system.

### Example 1

$Z(u_i) := s_a / Z(u_l)$
$x_{n+i} * x_{n+l} - x_a = 0$

### Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P))-n\log 3}{1+2\log 3}$.

### Proof. (Cont.)

For $1 \leq i \leq k$, consider node $u_i$:
**Case 1:** $u_i$ has one child (computation node)

- We add one equation and probably an inequality to our system.

### Example 1

$Z(u_i) := s_a/Z(u_l)$
$x_{n+i} * x_{n+l} - x_a = 0$

### Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P)) - n\log 3}{1+2\log 3}$.

### Proof. (Cont.)

For $1 \leq i \leq k$, consider node $u_i$:
**Case 1:** $u_i$ has one child (computation node)

- We add one equation and probably an inequality to our system.

### Example 2

$Z(u_i) := \sqrt{s_a}$
$x_{n+i}^2 - x_a = 0$ and
$-x_{n+i} \leq 0$

| Assignment | Equation/inequality |
|---|---|
| $Z(u_i) := Z(u_j) + Z(u_\ell)$ | $x_{n+i} - x_{n+j} - x_{n+\ell} = 0$ |
| $Z(u_i) := Z(u_j) - Z(u_\ell)$ | $x_{n+i} - x_{n+j} + x_{n+\ell} = 0$ |
| $Z(u_i) := Z(u_j) * Z(u_\ell)$ | $x_{n+i} - x_{n+j}x_{n+\ell} = 0$ |
| $Z(u_i) := Z(u_j)/Z(u_\ell)$ | $x_{n+i}x_{n+\ell} - x_{n+j} = 0$ |
| $Z(u_i) := \sqrt{Z(u_j)}$ | $x_{n+i}^2 - x_{n+j} = 0$ and $-x_{n+i} \leq 0$ |
| $Z(u_i) := s_a + Z(u_\ell)$ | $x_{n+i} - x_a - x_{n+\ell} = 0$ |
| $Z(u_i) := s_a - Z(u_\ell)$ | $x_{n+i} - x_a + x_{n+\ell} = 0$ |
| $Z(u_i) := Z(u_\ell) - s_a$ | $x_{n+i} - x_{n+\ell} + x_a = 0$ |
| $Z(u_i) := s_a * Z(u_\ell)$ | $x_{n+i} - x_a x_{n+\ell} = 0$ |
| $Z(u_i) := s_a/Z(u_\ell)$ | $x_{n+i}x_{n+\ell} - x_a = 0$ |
| $Z(u_i) := Z(u_\ell)/s_a$ | $x_{n+i}x_a - x_{n+\ell} = 0$ |
| $Z(u_i) := s_a + s_b$ | $x_{n+i} - x_a - x_b = 0$ |
| $Z(u_i) := s_a - s_b$ | $x_{n+i} - x_a + x_b = 0$ |
| $Z(u_i) := s_a * s_b$ | $x_{n+i} - x_a x_b = 0$ |
| $Z(u_i) := s_a/s_b$ | $x_{n+i}x_b - x_a = 0$ |
| $Z(u_i) := \sqrt{s_a}$ | $x_{n+i}^2 - x_a = 0$ and $-x_{n+i} \leq 0$ |
| $Z(u_i) := c + Z(u_\ell)$ | $x_{n+i} - c - x_{n+\ell} = 0$ |
| $Z(u_i) := c - Z(u_\ell)$ | $x_{n+i} - c + x_{n+\ell} = 0$ |
| $Z(u_i) := Z(u_\ell) - c$ | $x_{n+i} - x_{n+\ell} + c = 0$ |
| $Z(u_i) := c * Z(u_\ell)$ | $x_{n+i} - cx_{n+\ell} = 0$ |

## Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P)) - n\log 3}{1+2\log 3}$.

## Proof. (Cont.)

**Case 2:** $u_i$ has two children (comparison node)

- if the path in $T$ to $w$ proceeds from $u_i$ to its left child
  - The comparison in $u_i$: $Z(u_j) \bowtie 0$, we add $x_{n+j} \leq 0$
  - The comparison in $u_i$: $s_a \bowtie 0$, we add $x_a \leq 0$
- if the path in $T$ to $w$ proceeds from $u_i$ to its right child
  - The comparison in $u_i$: $Z(u_j) \bowtie 0$, we add $x_{n+j} > 0$
  - The comparison in $u_i$: $s_a \bowtie 0$, we add $x_a > 0$

## Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P))-n\log 3}{1+2\log 3}$.

## Proof. (Cont.)

$r = \#$ computation nodes on the path to $w$
$s = \#$ computation nodes on this path labeled $\sqrt{\dots}$
$t = \#$ times this path proceeds to its left child

- $r$ polynomial equations
- $s+t$ polynomial $\leq$-inequalities
- $k-r-t$ polynomial $>$-inequalities

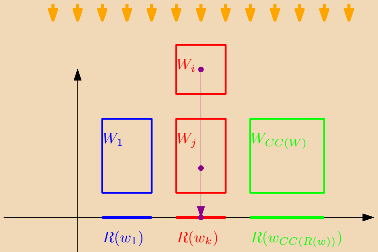
in the variables $x_1, \dots, x_{n+k}$
By Theorem 3.3.6 $W$ has at most $3^{n+2k+s-r}$ connected components.

## Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P)) - n\log 3}{1+2\log 3}$.

## Proof. (Cont.)

The projection of $W$ onto the first $n$ coordinates is equal to the set $R(w)$. This implies that
$CC(R(w)) \leq CC(W) \leq 3^{n+2k+s-r}$



## Theorem 3.3.7. time complexity in the ADT model

$P$: decision problem that is decidable in the ADT model
$V_P \subseteq \mathbb{R}^n$: the set of YES-instances of $P$.
Time complexity of $V_P$ in ADT model $\geq \frac{\log(CC(V_P))-n\log 3}{1+2\log 3}$.

## Proof. (Cont.)

$h$: height of $T$.
Since $k \leq h$ and $s \leq r$, $CC(R(w)) \leq CC(W) \leq 3^{n+2h}$.
$V_P$: YES-instances of $P$

$$V_P = \bigcup_{w\text{:YES-leaf of } T} R(w) \Rightarrow CC(V_P) \leq \sum_{w\text{:YES-leaf of } T} CC(R(w))$$

$T$ has at most $2^h$ leaves $\Rightarrow CC(V_P) \leq 3^{n+2h} \times 2^h$

$$h \geq \frac{\log(CC(V_P)) - n\log 3}{1 + 2\log 3}$$

## Section 3.3.3 Some applications

Lower bound of some problems.

## Theorem 3.3.9. Time complexity of the EUP

The time complexity of the element uniqueness problem for $n$ real numbers in ADT model is $\Omega(n \log n)$.

## Proof.

$V$ has at least $n!$ components. By Theorem 3.3.7,

$$\frac{\log(n!) - n\log 3}{1 + 2\log 3} = \Omega(n \log n)$$

lower bound.

## Corollary 3.3.10. Time complexity of sorting and closest pair problems

The following two problems have time complexity $\Omega(n \log n)$ in ACT model:
(1) the sorting problem for $n$ real numbers, and
(2) the closest pair problem on a set $S$ of $n$ points in $\mathbb{R}^d$.

## Proof.

The lower bound for the closest pair problem follows immediately from Theorem 3.3.9 because the input sequence contains two equal elements if and only if the distance of the closest pair is zero.

## Corollary 3.3.10. Time complexity of sorting and closest pair problems

The following two problems have time complexity $\Omega(n \log n)$ in ACT model:
(1) the sorting problem for $n$ real numbers, and
(2) the closest pair problem on a set $S$ of $n$ points in $\mathbb{R}^d$.

## Proof.

$A$: arbitrary ACT algorithm that solves the sorting problem in $T(n)$ time.
$B$: solves the EUP
$B$ sort the numbers and then compares all pairs of elements that are neighbors in the sorted sequence $\Rightarrow$
Time complexity of $B$=$T(n) + O(n)$
$T(n) + O(n) = \Omega(n \log n) \Rightarrow T(n) = \Omega(n \log n)$

Problems whose YES-instances have infinite connected components are non-decidable.

**Theorem 3.3.11. (non-decidable problems)**

There is no algebraic decision tree algorithm that, when given an arbitrary real number $x$ as input, returns YES if $x \in \mathbb{N}$, and NO if $x \notin \mathbb{N}$.

## Section 3.4. A lower bound for constructing spanners

We will use Theorem 3.3.7 to prove an $\Omega(n \log n)$ lower bound for constructing $t$-spanners.

We will focus on algorithms that construct Steiner $t$-spanners with $o(n \log n)$ edges for one-dimensional multisets, that is, multisets of real numbers. We will prove that even this one-dimensional case has an $\Omega(n \log n)$ lower bound. Of course, this implies the same lower bound for any dimension $d \geq 1$.

## 3.4.1. A reduction from the EUP

$A$: An arbitrary algebraic computation-tree algorithm that, constructs a Steiner $t$-spanner for the multiset $S = \{s_1, s_2, ..., s_n\}$ of $n$ points on the one-dimensional real line.
Each vertex of the output of $A$ is labeled as either being an element of $S$ or being a Steiner point.

**Claim:** Algorithm $A$ can be used to solve the EUP.

### EUP Algorithm

Step 1: $G$ : output of algorithm $A$ on the input sequence $s_1, s_2, ..., s_n$ and arbitrary $t > 1$.
Step 2: $G'$ : subgraph of $G$ such that $G'$ contains the same vertices as $G$, and $G'$ contains all edges of $G$ having length zero.
Step 3: Compute the connected components of the graph $G'$
Step 4: For each connected component of $G'$, check whether it contains two or more distinct non-Steiner elements (i.e., elements having distinct indices). If this is the case for some connected component, return NO. Otherwise, return YES.

**Theorem 3.4.1.**

In the algebraic computation-tree model, any algorithm that, when given a multiset $S$ of $n$ points in $\mathbb{R}^d$, ($d$ constant) and a real number $t > 1$, constructs a Steiner $t$-spanner for $S$, takes $\Omega(n \log n)$ time in the worst case.

If the points are known to be pairwise distinct, then the EUP can be solved in O(1) time,
because the output is always YES.
In the next section, we will consider algorithms that construct Steiner spanners for inputs consisting of pairwise distinct points.

## 3.4.2. Lower bound for pairwise distinct points

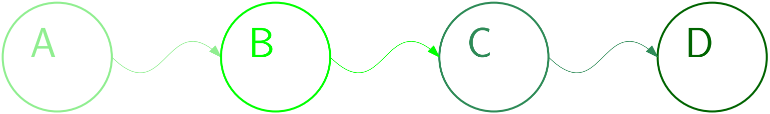
In the algebraic computation-tree model, the lower bound of $\Omega(n \log n)$ for the Steiner $t$-spanner construction problem holds even if the input is known to consist of pairwise distinct points. The proof effectively uses a lower bound of $\Omega(n \log n)$ for the mingap problem.

## We can not apply Theorem 3.3.7 because:

The set of all inputs of $A$ has $\Omega(n!)$ components.

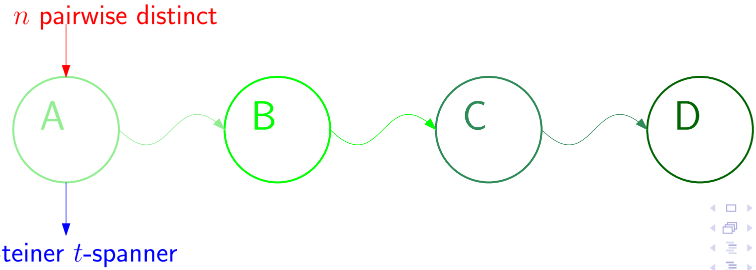
### 3.4.2. Lower bound for pairwise distinct points

In the algebraic computation-tree model, the lower bound of $\Omega(n \log n)$ for the Steiner $t$-spanner construction problem holds even if the input is known to consist of pairwise distinct points. The proof effectively uses a lower bound of $\Omega(n \log n)$ for the mingap problem.



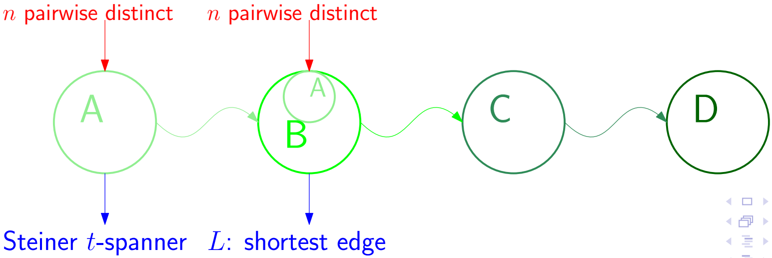
### Algorithm $A$

$A$ denotes an arbitrary algebraic computation-tree algorithm that, when given a set $S$ of $n$ pairwise distinct real numbers, and a real number $t > 1$, constructs a Steiner t-spanner for $S$ with $o(n \log n)$ edges.



## Algorithm $B$

$B$ takes pairwise distinct real numbers as input. This algorithm runs algorithm $A$ on this input, and returns the length $L$ of a shortest edge of nonzero length in the graph that $A$ computes.



**Lemma 3.4.2.**

The real number $L$ that is returned by algorithm $B$ satisfies $0 < L \leq t \times \text{mingap}(s_1, s_2, ..., s_n)$.

**Proof:**

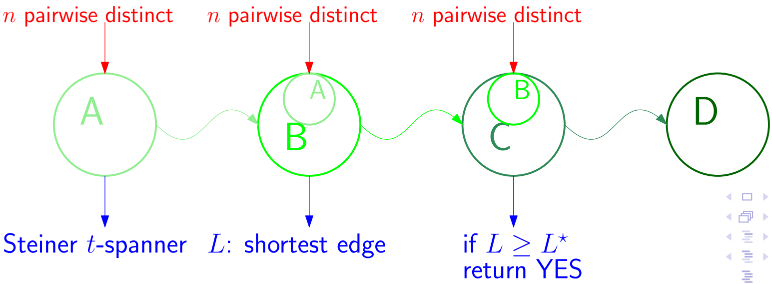
Let $|s_i - s_j| = \text{mingap}\{s_1, \ldots, s_n\}$.
$\exists$ a path between $s_i$ and $s_j$ of length $t \times \text{mingap}\{s_1, \ldots, s_n\}$.
Each edge of this path has length $\leq t \times \text{mingap}\{s_1, \ldots, s_n\}$. DONE!

$$\text{Time: } T_B(n,t) \leq T_A(n,t) + o(n \log n).$$

$L^\star$ : Minimum value return by $B$ among all runs of $B$ on all permutations of $1, 2, \ldots, n$. $L^\star$ is indepenent of the input, we we can assume that we know $L^\star$.

**Algorithm $C$**

Algorithm $C$ takes pairwise distinct real numbers as input. It runs algorithm $B$ on this input, and returns YES if and only if the output $L$ of $B \geq L^\star$.



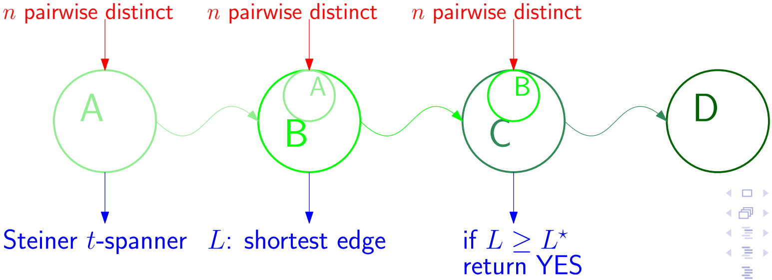
# Lower bound for constructing spanners

## Pairwise distinct points

$L^\star$ : Minimum value return by $B$ among all runs of $B$ on all permutations of $1, 2, \ldots, n$. $L^\star$ is indepenent of the input, we we can assume that we know $L^\star$.

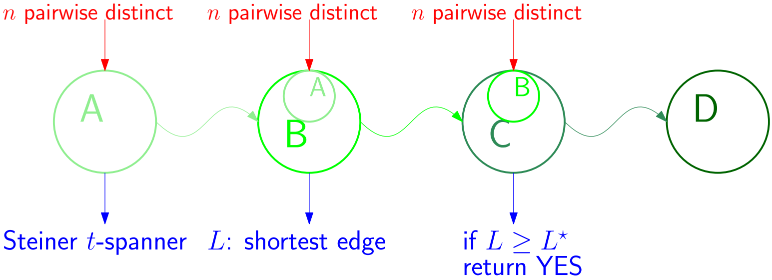
### Algorithm $C$

Algorithm $C$ takes pairwise distinct real numbers as input. It runs algorithm $B$ on this input, and returns YES if and only if the output $L$ of $B \geq L^\star$.



## Algorithm $C$

Algorithm $C$ pairwise distinct real numbers as input. It runs algorithm $B$ on this input, and returns YES if and only if the output $L$ of $B \geq L^{\star}$.




Running time of algorithm $C$ is a constant factor of $B$'s running time.

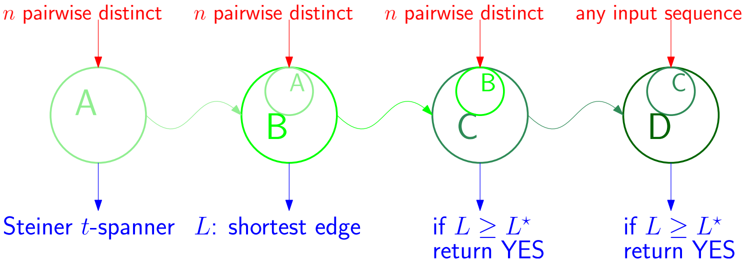
# Lower bound for constructing spanners
## Pairwise distinct points

### Algorithm $D$: Same as $C$ except:

- Accept any input (not just paiwise distinct elements.)
- In any place that $C$ perform division $z = x/y$, it checks the denominator to be nonzero.
- In any place that $C$ perform $z = \sqrt{y}$, it checks $y$ to be non-negative.

### Algorithm $D$:

has the same output as $C$, on an input of pairwise distict points.
For a set of non-pairwise distinct elements, $C$ has no input and the output of $D$ is meaningless.
is well-defined on any input sequence of real numbers.
is the algebraic decision tree algorithm we are looking for.



$n$ pairwise distinct
$n$ pairwise distinct
$n$ pairwise distinct
any input sequence
A
A
B
B
C
C
D
Steiner $t$-spanner
$L$: shortest edge
if $L \geq L^\star$ return YES
if $L \geq L^\star$ return YES

### Analysis of algorithm $D$

We now prove that the worst-case running time of algorithm $D$ is $\Omega(n \log n)$.
This will imply the same lower bound on the running time of our target algorithm $A$.

**Lemma 3.4.3.: The set $W$ of YES-instances of $D$ has at least $n!$ connected components.**

Proof: Let

$$p = (\pi(1), \pi(2), ..., \pi(n))$$

$$r = (\rho(1), \rho(2), ..., \rho(n))$$

Consider $i$ and $j$ s. t. $\pi(i) < \pi(j)$ and $\rho(i) > \rho(j)$
We show that $p$ and $r$ belongs to different connected components.
Let $C$: is a curve that connects $p$ and $r$.
We show: $\exists q \in C$ such that $q \notin W$.

**Lemma 3.4.3.: The set $W$ of YES-instances of $D$ has at least $n!$ connected components.**

Proof (cont.): $C$ passes through hyperplane $x_i = x_j$.
$q = (q_1, q_2, \ldots, q_n)$: first point on $C$, such that

$$\text{mingap}(q_1, q_2, ..., q_n) \leq \frac{L^\star}{2t}.$$

We will show that the coordinates of $q$ are pairwise distinct.
If we run algorithm $B$ on input $q_1, q_2, ..., q_n, t$, then

$$L \leq t \times \frac{L^\star}{2t} < L^\star$$

This means that $q \notin W$ and we are done.

**Lemma 3.4.3.: The set $W$ of YES-instances of $D$ has at least $n!$ connected components.**

Proof (cont.): $C(\tau)$: parameterize the curve $C$, $0 \leq \tau \leq 1$, where $C(0) = p$ and $C(1) = r$.
$C(\tau)_k$: $k$-th coordinate of the point $C(\tau)$.
We define

$$\tau_0 := \min_{0\leq\tau\leq 1}\{\text{mingap}(C(\tau)_1, C(\tau)_2, \ldots, C(\tau)_n) \leq \frac{L^\star}{2t}\}$$

Let $q = C(\tau_0)$. We have

$$\text{mingap}(q_1, q_2, \ldots, q_n) \leq \frac{L^\star}{2t}$$

Also, by Lemma 3.4.2, and since $C(0) = p \in W$,

$$\text{mingap}(C(0)_1, C(0)_2, \ldots, C(0)_n) \geq \frac{L^\star}{t} > \frac{L^\star}{2t}$$

**Lemma 3.4.3.: The set $W$ of YES-instances of $D$ has at least $n!$ connected components.**

Proof (cont.): $q$ is the first point on $C$ s.t.
mingap$(q) \leq L/2t$
$C$ is continuous
Therefore, mingap$(q) > 0$.
Now, run algorithm $D$ on the input sequence
$q = \{q_1, q_2, \ldots, q_n\}$.

$$L \leq t \times \text{mingap}(q_1, q_2, \ldots, q_n).$$
$$L \leq t \times \frac{L^\star}{2t} < L^\star$$

So, algorithm $D$ returns NO. This implies that $q \notin W$. This completes the proof.

**Theorem 3.4.4.**

Let $d \geq 1$ be an integer constant. In the algebraic computation-tree model, any algorithm that, when given a set $S$ of $n$ pairwise distinct points in $\mathbb{R}^n$ and a real number $t > 1$, constructs a Steiner $t$-spanner for $S$, takes $\Omega(n \log n)$ time in the worst case.

**Open problem:**

Let $d \geq 2$ be an integer constant. Prove that, in the algebraic computation-tree model, any algorithm that, when given a set $S$ of $n$ points in $\mathbb{R}^d$ that are in general position and a real number $t > 1$, constructs a Steiner $t$-spanner for $S$, takes $\Omega(n \log n)$ time in the worst case.

## Theorem 3.4.4.

Let $d \geq 1$ be an integer constant. In the algebraic computation-tree model, any algorithm that, when given a set $S$ of $n$ pairwise distinct points in $\mathbb{R}^n$ and a real number $t > 1$, constructs a Steiner $t$-spanner for $S$, takes $\Omega(n \log n)$ time in the worst case.

## Open problem:

Let $d \geq 2$ be an integer constant. Prove that, in the algebraic computation-tree model, any algorithm that, when given a set $S$ of $n$ points in $\mathbb{R}^d$ that are in general position and a real number $t > 1$, constructs a Steiner $t$-spanner for $S$, takes $\Omega(n \log n)$ time in the worst case.