Tree isomorphism

Alexander Smal

St.Petersburg State University of Information Technologies, Mechanics and Optics

Joint Advanced Student School 2008

Motivation

In some applications the chemical structures are often trees with millions of vertices:

- gene splicing,
- protein analysis,
- molecular biology.

Difference between $O(n)$, $O(n \log n)$, and $O(n^2)$ isomorphism algorithms is not just theoretical importance.

Graph isomorphism

Definition

Isomorphism of graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is a bijection between the vertex sets $\varphi : V_1 \to V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2.$$

Graph isomorphism

Definition

Isomorphism of graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is a bijection between the vertex sets $\varphi : V_1 \to V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2.$$

Facts

- No algorithm, other than brute force, is known for testing whether two arbitrary graphs are isomorphic.

Graph isomorphism

Definition

Isomorphism of graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is a bijection between the vertex sets $\varphi : V_1 \to V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2.$$

Facts

- No algorithm, other than brute force, is known for testing whether two arbitrary graphs are isomorphic.
- It is still an open question (!) whether graph isomorphism is $\mathcal{NP}$ complete.

Graph isomorphism

Definition

Isomorphism of graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is a bijection between the vertex sets $\varphi : V_1 \rightarrow V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2.$$

Facts

- No algorithm, other than brute force, is known for testing whether two arbitrary graphs are isomorphic.
- It is still an open question (!) whether graph isomorphism is $\mathcal{NP}$ complete.
- Polynomial time isomorphism algorithms for various graph subclasses such as trees are known.

Rooted trees

Definition

Rooted tree (V, E, r) is a tree (V, E) with selected root $r \in V$.

Definition

Rooted tree (V, E, r) is a tree (V, E) with selected root $r \in V$.

Definition

Isomorphism of rooted trees $T_1(V_1, E_1, r_1)$ and $T_2(V_2, E_2, r_2)$ is a bijection between the vertex sets $\varphi : V_1 \to V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2$$

and $\varphi(r_1) = r_2$.

Definition

Rooted tree (V, E, r) is a tree (V, E) with selected root $r \in V$.

Definition

Isomorphism of rooted trees $T_1(V_1, E_1, r_1)$ and $T_2(V_2, E_2, r_2)$ is a bijection between the vertex sets $\varphi : V_1 \to V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2$$

and $\varphi(r_1) = r_2$.

Definition

Rooted tree (V, E, r) is a tree (V, E) with selected root $r \in V$.

Definition

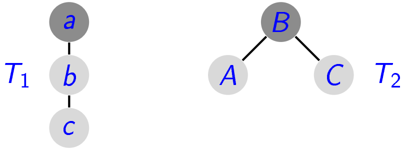
Isomorphism of rooted trees $T_1(V_1, E_1, r_1)$ and $T_2(V_2, E_2, r_2)$ is a bijection between the vertex sets $\varphi : V_1 \to V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2$$

and $\varphi(r_1) = r_2$.

Example

T_1 and T_2 are isomorphic as *graphs* . . .



Rooted trees

Definition

Rooted tree (V, E, r) is a tree (V, E) with selected root $r \in V$.

Definition

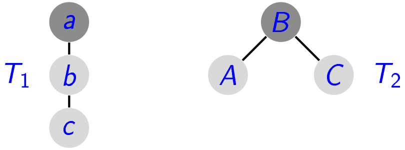
Isomorphism of rooted trees $T_1(V_1, E_1, r_1)$ and $T_2(V_2, E_2, r_2)$ is a bijection between the vertex sets $\varphi : V_1 \to V_2$ such that

$$\forall u, v \in V_1 \quad (u, v) \in E_1 \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2$$

and $\varphi(r_1) = r_2$.

Example

T_1 and T_2 are isomorphic as *graphs* but not as *rooted trees*!



Rooted trees (part 2)

Lemma

If there is $O(n)$ algorithm for rooted trees isomorphism there is $O(n)$ algorithm for ordinary trees isomorphism.

Rooted trees (part 2)

Lemma

If there is $O(n)$ algorithm for rooted trees isomorphism there is $O(n)$ algorithm for ordinary trees isomorphism.

Proof.

1. Let $\mathcal{A}$ to be $O(n)$ algorithm for rooted trees.

Rooted trees (part 2)

Lemma

If there is $O(n)$ algorithm for rooted trees isomorphism there is $O(n)$ algorithm for ordinary trees isomorphism.

Proof.

1. Let $\mathcal{A}$ to be $O(n)$ algorithm for rooted trees.
2. Let T_1 and T_2 to be ordinary trees.

Rooted trees (part 2)

Lemma

If there is $O(n)$ algorithm for rooted trees isomorphism there is $O(n)$ algorithm for ordinary trees isomorphism.

Proof.

1. Let $\mathcal{A}$ to be $O(n)$ algorithm for rooted trees.
2. Let T_1 and T_2 to be ordinary trees.
3. Lets find centers of this trees. There are three possibilities:

Rooted trees (part 2)

Lemma

If there is $O(n)$ algorithm for rooted trees isomorphism there is $O(n)$ algorithm for ordinary trees isomorphism.

Proof.

1. Let $\mathcal{A}$ to be $O(n)$ algorithm for rooted trees.
2. Let T_1 and T_2 to be ordinary trees.
3. Lets find centers of this trees. There are three possibilities:
    1. each tree has only one center (c_1 and c_2 respectively)
       return $\mathcal{A}(T_1, c_1, T_2, c_2)$

Rooted trees (part 2)

Lemma

If there is $O(n)$ algorithm for rooted trees isomorphism there is $O(n)$ algorithm for ordinary trees isomorphism.

Proof.

1. Let $\mathcal{A}$ to be $O(n)$ algorithm for rooted trees.
2. Let T_1 and T_2 to be ordinary trees.
3. Lets find centers of this trees. There are three possibilities:
   1. each tree has only one center (c_1 and c_2 respectively)
      return $\mathcal{A}(T_1, c_1, T_2, c_2)$
   2. each tree has exactly two centers (c_1, c_1' and c_2, c_2' respectively)
      return $\mathcal{A}(T_1, c_1, T_2, c_2)$ **or** $\mathcal{A}(T_1, c_1', T_2, c_2)$

Rooted trees (part 2)

Lemma

If there is $O(n)$ algorithm for rooted trees isomorphism there is $O(n)$ algorithm for ordinary trees isomorphism.

Proof.

1. Let $\mathcal{A}$ to be $O(n)$ algorithm for rooted trees.
2. Let T_1 and T_2 to be ordinary trees.
3. Lets find centers of this trees. There are three possibilities:
    1. each tree has only one center (c_1 and c_2 respectively)
       return $\mathcal{A}(T_1, c_1, T_2, c_2)$
    2. each tree has exactly two centers (c_1, c_1' and c_2, c_2' respectively)
       return $\mathcal{A}(T_1, c_1, T_2, c_2)$ **or** $\mathcal{A}(T_1, c_1', T_2, c_2)$
    3. trees has different count of centers
       return False

□

Diameter and center

Definition
The **diameter** of tree is the length of the longest path.

Diameter and center

Definition

The **diameter** of tree is the length of the longest path.

Definition

A **center** is a vertex v such that the longest path from v to a leaf is minimal over all vertices in the tree.

Diameter and center

Definition

The **diameter** of tree is the length of the longest path.

Definition

A **center** is a vertex v such that the longest path from v to a leaf is minimal over all vertices in the tree.

Algorithm

1: *Choose a random root* r.
2: *Find a vertex* v_1 *— the farthest form* r.
3: *Find a vertex* v_2 *— the farthest form* v_1.
4: *Diameter is a length of path from* v_1 *to* v_2.
5: *Center is a median element(s) of path from* v_1 *to* v_2.

Diameter and center

Definition

The **diameter** of tree is the length of the longest path.

Definition

A **center** is a vertex v such that the longest path from v to a leaf is minimal over all vertices in the tree.

Algorithm

1: *Choose a random root* r.
2: *Find a vertex* v_1 *— the farthest form* r.
3: *Find a vertex* v_2 *— the farthest form* v_1.
4: *Diameter is a length of path from* v_1 *to* v_2.
5: *Center is a median element(s) of path from* v_1 *to* v_2.

It is $O(n)$ algorithm.

The idea

Let's try to find *complete invariant* of rooted trees isomorphism.

The idea

Let's try to find *complete invariant* of rooted trees isomorphism.

Definition

Isomorphism invariant is a function $f(T)$ such that $f(T_1) = f(T_2)$ for all pairs of isomorphic trees T_1 and T_2.

The idea

Let's try to find *complete invariant* of rooted trees isomorphism.

Definition

Isomorphism invariant is a function $f(T)$ such that $f(T_1) = f(T_2)$ for all pairs of isomorphic trees T_1 and T_2.

Definition

Complete isomorphism invariant is a function $f(T)$ such that two trees T_1 and T_2 are isomorphic if and only if $f(T_1) = f(T_2)$.

The idea

Let's try to find *complete invariant* of rooted trees isomorphism.

Definition

Isomorphism invariant is a function $f(T)$ such that $f(T_1) = f(T_2)$ for all pairs of isomorphic trees T_1 and T_2.

Definition

Complete isomorphism invariant is a function $f(T)$ such that two trees T_1 and T_2 are isomorphic if and only if $f(T_1) = f(T_2)$.

So if we find complete isomorphism invariant we can obtain algorithm from it.

The idea

Let's try to find *complete invariant* of rooted trees isomorphism.

Definition
Isomorphism invariant is a function $f(T)$ such that $f(T_1) = f(T_2)$ for all pairs of isomorphic trees T_1 and T_2.

Definition
Complete isomorphism invariant is a function $f(T)$ such that two trees T_1 and T_2 are isomorphic if and only if $f(T_1) = f(T_2)$.

So if we find complete isomorphism invariant we can obtain algorithm from it.

Note
Starting from the next slide *tree* means *rooted tree*!

Candidate 1

Observation

The level number of a vertex is a tree isomorphism invariant.

Candidate 1

Observation

The level number of a vertex is a tree isomorphism invariant.

Conjecture

Two trees are isomorphic if and only if they have the same number of levels and the same number of vertices on each level.

Candidate 1

Observation

The level number of a vertex is a tree isomorphism invariant.

Conjecture

Two trees are isomorphic if and only if they have the same number of levels and the same number of vertices on each level.

Observation

The number of the leaves is a tree isomorphism invariant.

Candidate 1

Observation

The level number of a vertex is a tree isomorphism invariant.

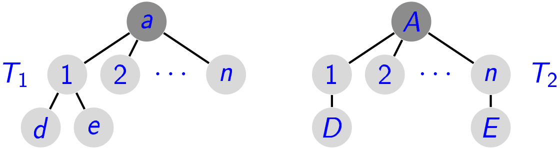
Conjecture

Two trees are isomorphic if and only if they have the same number of levels and the same number of vertices on each level.

Observation

The number of the leaves is a tree isomorphism invariant.

Contrary instance



Candidate 2

What's wrong with candidate 1?

We didn't take into account the *spectrum degree* of a tree.

Candidate 2

What's wrong with candidate 1?

We didn't take into account the *spectrum degree* of a tree.

Definition

Degree spectrum of tree is the sequence of non-negative integers $\{d_j\}$, where d_j is the number of vertices that have j children.

Candidate 2

What's wrong with candidate 1?

We didn't take into account the *spectrum degree* of a tree.

Definition

Degree spectrum of tree is the sequence of non-negative integers $\{d_j\}$, where d_j is the number of vertices that have j children.

Conjecture

Two trees are isomorphic if and only if they have the same degree spectrum.

Candidate 2 (part 2)

Observation

Since a tree isomorphism preserves longest paths from the root, the number of levels in a tree is a tree isomorphism invariant.

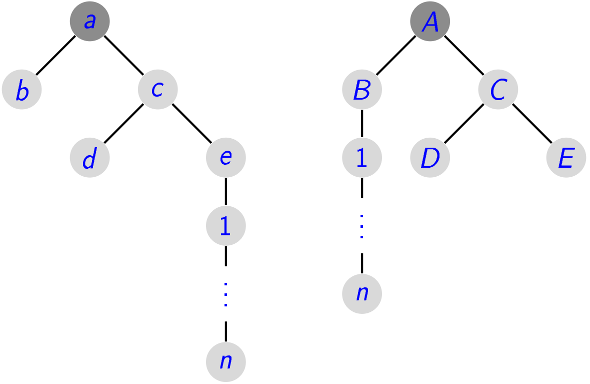
Candidate 2 (part 2)

Observation

Since a tree isomorphism preserves longest paths from the root, the number of levels in a tree is a tree isomorphism invariant.

Contrary instance

T_1



T_2

Candidate 3

Conjecture

Two trees are isomorphic if and only if they have the same degree spectrum at each level.

Candidate 3

Conjecture

Two trees are isomorphic if and only if they have the same degree spectrum at each level.

If two trees have the same degree spectrum at each level, then they must automatically have the same numbers of levels, the same numbers of vertices at each level, and the same global degree spectrum!

Candidate 3

Conjecture

Two trees are isomorphic if and only if they have the same degree spectrum at each level.

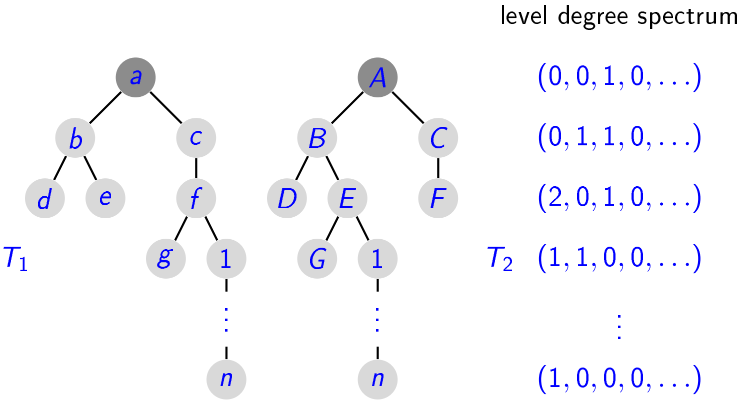
If two trees have the same degree spectrum at each level, then they must automatically have the same numbers of levels, the same numbers of vertices at each level, and the same global degree spectrum!

Observation

The number of leaf descendants of a vertex and the level number of a vertex are both tree isomorphism invariants.

Candidate 3 (part 2)

Contrary instance



AHU algorithm

Algorithm by Aho, Hopcroft and Ullman

- Determine tree isomorphism in time $O(|V|)$.
- Uses *complete history of degree spectrum of the vertex descendants* as a complete invariant.

AHU algorithm

Algorithm by Aho, Hopcroft and Ullman

- Determine tree isomorphism in time $O(|V|)$.
- Uses *complete history of degree spectrum of the vertex descendants* as a complete invariant.

The idea of AHU algorithm

The AHU algorithm associates with each vertex a tuple that describes the complete history of its descendants.

AHU algorithm

Algorithm by Aho, Hopcroft and Ullman

- Determine tree isomorphism in time $O(|V|)$.
- Uses *complete history of degree spectrum of the vertex descendants* as a complete invariant.

The idea of AHU algorithm

The AHU algorithm associates with each vertex a tuple that describes the complete history of its descendants.

Hard question

Why our previous invariants are not complete?

AHU algorithm

Algorithm by Aho, Hopcroft and Ullman

- Determine tree isomorphism in time $O(|V|)$.
- Uses *complete history of degree spectrum of the vertex descendants* as a complete invariant.

The idea of AHU algorithm

The AHU algorithm associates with each vertex a tuple that describes the complete history of its descendants.

Hard question

Why our previous invariants are not complete?

Let's discuss AHU algorithm. We start from $O(|V|^2)$ version and then I tell how to make it faster ($O(|V|)$).

Understanding AHU algorithm

Knuth tuples

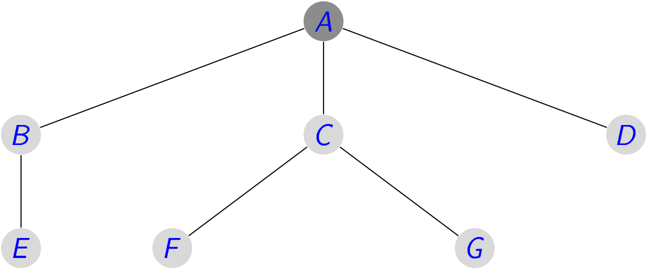
Let's assign parenthetical tuples to all tree vertices.

Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

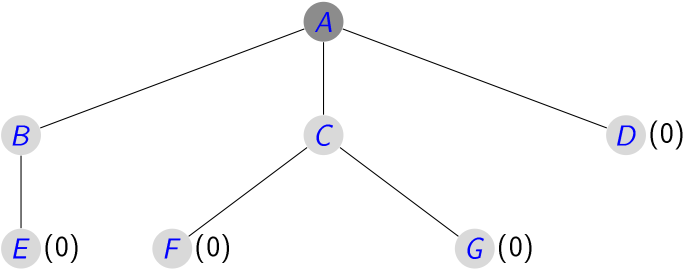


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

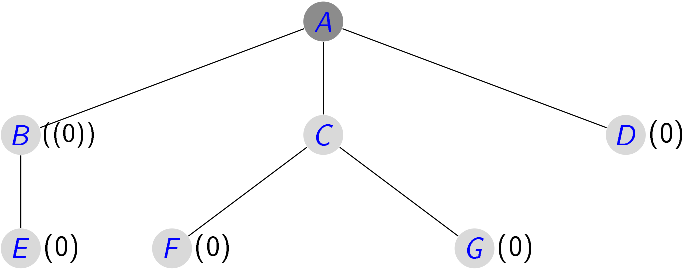


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

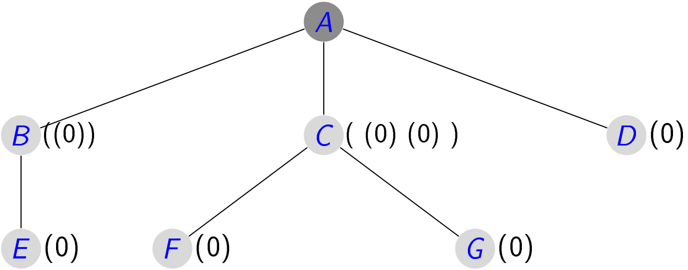


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

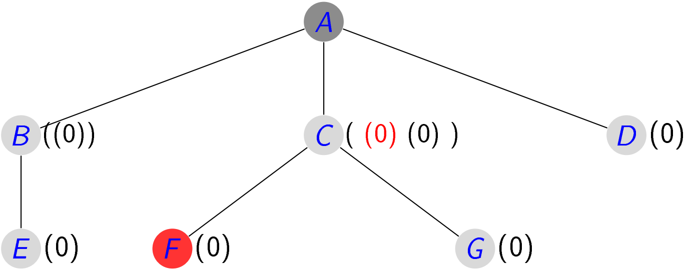


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

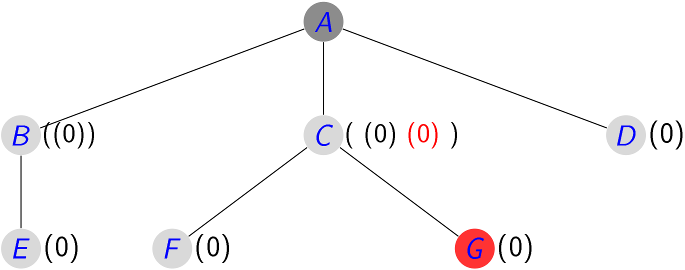


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

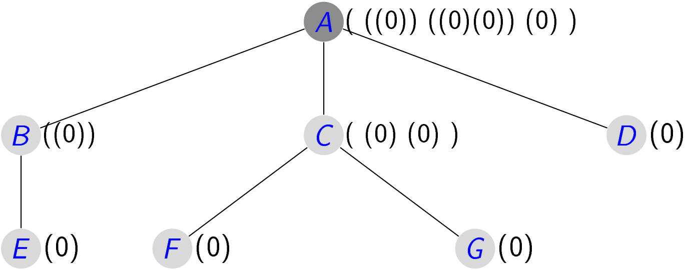


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

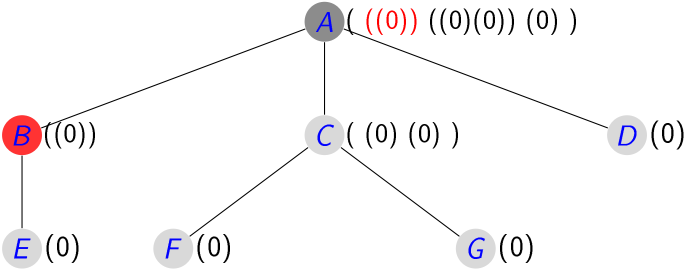


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

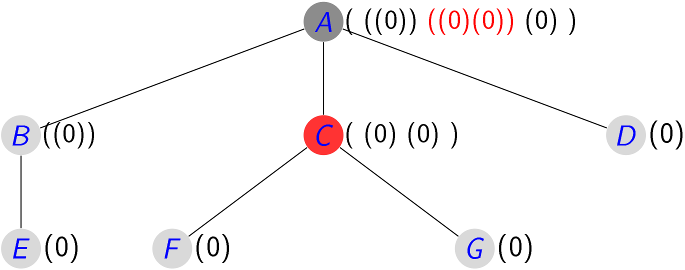


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example

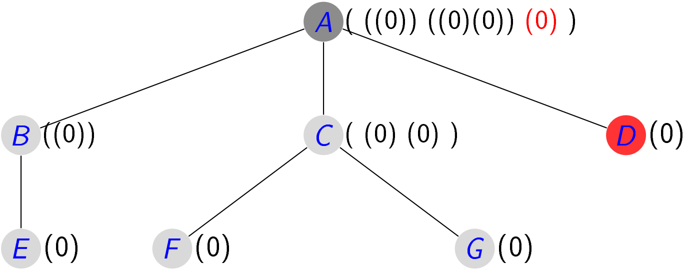


Understanding AHU algorithm

Knuth tuples

Let's assign parenthetical tuples to all tree vertices.

Knuth tuples example



Understanding AHU algorithm (part 2)

There is algorithm Assign-Knuth-Tuples(v) that visits every vertex once or twice.

Assign-Knuth-Tuples(v)

```
if v is a leaf then
    Give v the tuple name (0)
else
    for all child w of v do
        Assign-Knuth-Tuples(w)
    end for
end if
Concatenate the names of all children of v to temp
Give v the tuple name temp
```

Understanding AHU algorithm (part 3)

Observation

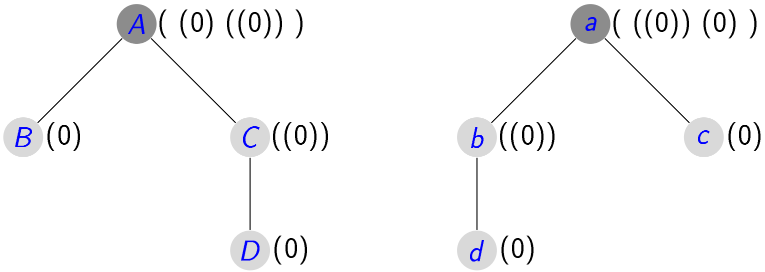
There is no order on parenthetical tuples.

Understanding AHU algorithm (part 3)

Observation

There is no order on parenthetical tuples.

Example

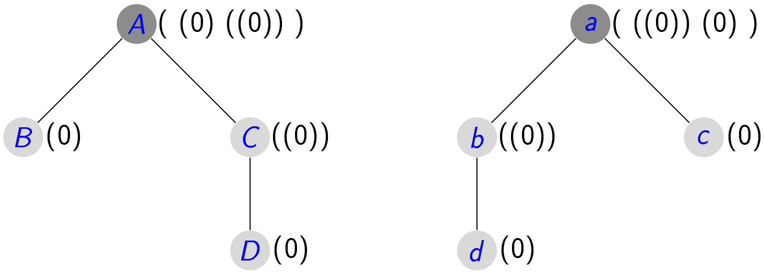


Understanding AHU algorithm (part 3)

Observation

There is no order on parenthetical tuples.

Example



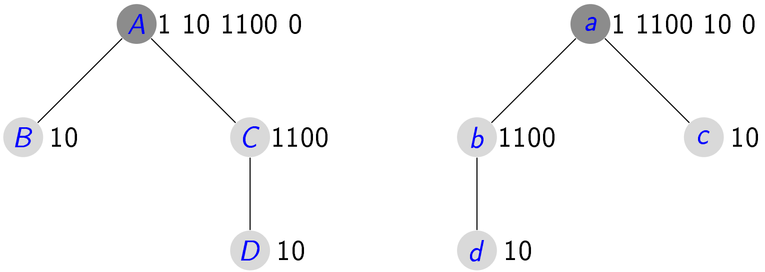
Let's convert parenthetical tuples to *canonical names*. We should drop all "0"-s and replace "(" and ")" with "1" and "0" respectively.

Understanding AHU algorithm (part 3)

Observation

There is no order on parenthetical tuples.

Example



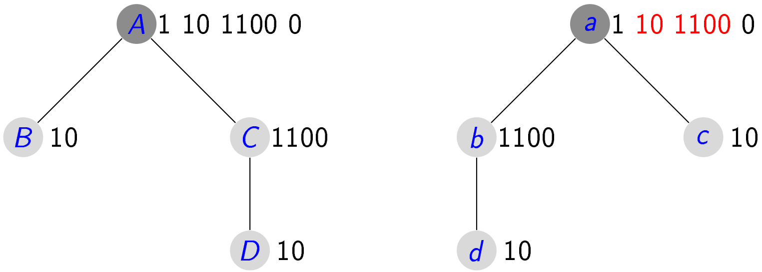
Let's convert parenthetical tuples to *canonical names*. We should drop all "0"-s and replace "(" and ")" with "1" and "0" respectively.

Understanding AHU algorithm (part 3)

Observation

There is no order on parenthetical tuples.

Example



Let's convert parenthetical tuples to *canonical names*. We should drop all "0"-s and replace "(" and ")" with "1" and "0" respectively.

Understanding AHU algorithm (part 4)

ASSIGN-CANONICAL-NAMES(*v*)

```
if v is a leaf then
    Give v the tuple name "10"
else
    for all child w of v do
        ASSIGN-CANONICAL-NAMES(v)
    end for
end if
Sort the names of the children of v
Concatenate the names of all children of v to temp
Give v the name 1temp0
```

Understanding AHU algorithm (part 5)

We should discuss some important questions.

Understanding AHU algorithm (part 5)

We should discuss some important questions.

Invariant?

Is canonical name of a root is a *tree isomorphism invariant*?

Understanding AHU algorithm (part 5)

We should discuss some important questions.

Invariant?

Is canonical name of a root is a *tree isomorphism invariant*?

Complete invariant?

Is canonical name of a root is a *complete tree isomorphism invariant*?

Understanding AHU algorithm (part 5)

We should discuss some important questions.

Invariant?

Is canonical name of a root is a *tree isomorphism invariant*?

Complete invariant?

Is canonical name of a root is a *complete tree isomorphism invariant*?

AHU-TREE-ISOMORPHISM($T_1(V_1, E_1, r_1),\ T_2(V_2, E_2, r_2)$)

```
ASSIGN-CANONICAL-NAMES(r1)
ASSIGN-CANONICAL-NAMES(r2)
if name(r1) = name(r2) then
    return True
else
    return False
end if
```

AHU algorithm improvement

Observation

To compute the root name of a tree of n vertices in one long strand, takes time proportional to $1 + 2 + \cdots + n$, which is $\Omega(n^2)$.

AHU algorithm improvement

Observation

To compute the root name of a tree of n vertices in one long strand, takes time proportional to $1 + 2 + \cdots + n$, which is $\Omega(n^2)$.

Observation

For all levels i, the canonical name of level i is a tree isomorphism invariant.

AHU algorithm improvement

Observation

To compute the root name of a tree of n *vertices in one long strand, takes time proportional to* $1 + 2 + \cdots + n$, *which is* $\Omega(n^2)$.

Observation

For all levels i, *the canonical name of level* i *is a tree isomorphism invariant.*

Observation

Two trees T_1 *and* T_2 *are isomorphic if and only if for all levels* i *canonical level names of* T_1 *and* T_2 *are identical.*

AHU algorithm improvement

Observation

To compute the root name of a tree of n vertices in one long strand, takes time proportional to $1 + 2 + \cdots + n$, which is $\Omega(n^2)$.

Observation

For all levels i, the canonical name of level i is a tree isomorphism invariant.

Observation

Two trees T_1 and T_2 are isomorphic if and only if for all levels i canonical level names of T_1 and T_2 are identical.

The idea 1

Assign canonical names level, sort by level, and check by level that the canonical level names agree.

AHU algorithm improvement

Observation

To compute the root name of a tree of n vertices in one long strand, takes time proportional to $1 + 2 + \cdots + n$, which is $\Omega(n^2)$.

Observation

For all levels i, the canonical name of level i is a tree isomorphism invariant.

Observation

Two trees T_1 and T_2 are isomorphic if and only if for all levels i canonical level names of T_1 and T_2 are identical.

The idea 1

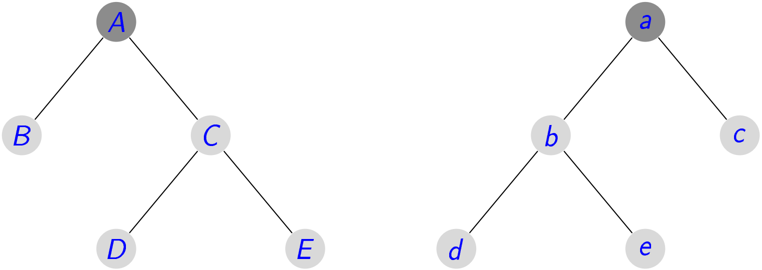
Assign canonical names level, sort by level, and check by level that the canonical level names agree.

The idea 2

Assign canonical names level and if canonical level names agree than replace canonical names with integers.

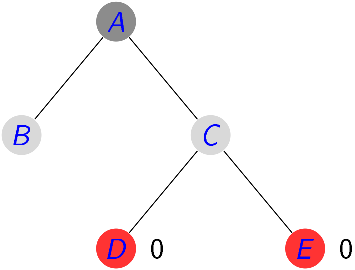
AHU algorithm example

Example



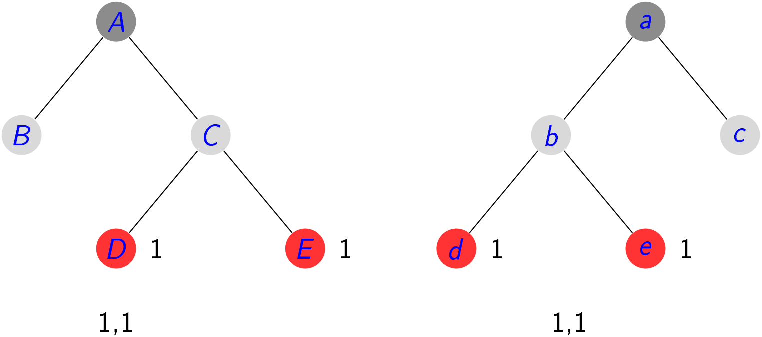
AHU algorithm example

Example



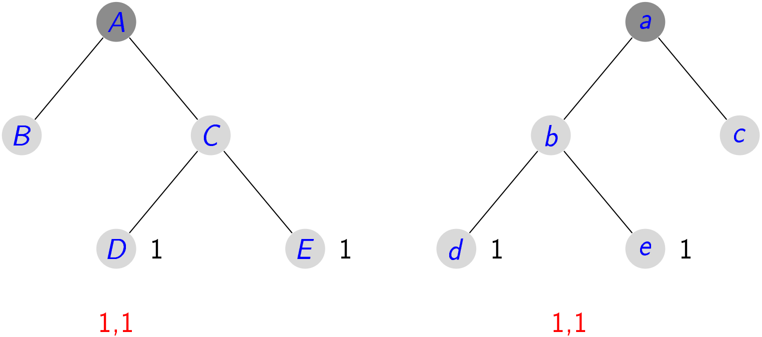
AHU algorithm example

Example



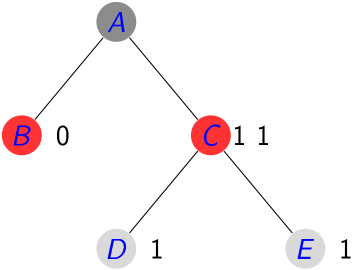
AHU algorithm example

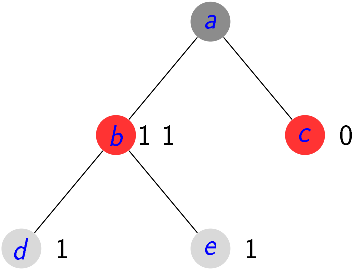
Example



AHU algorithm example

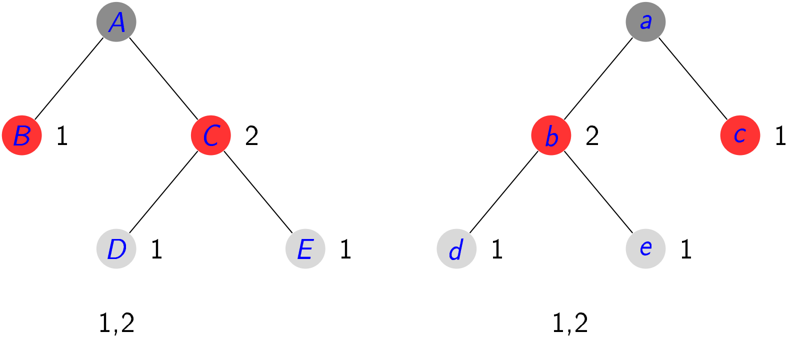
Example





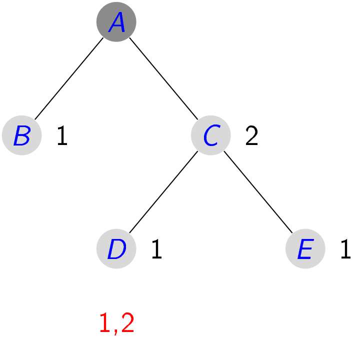
AHU algorithm example

Example



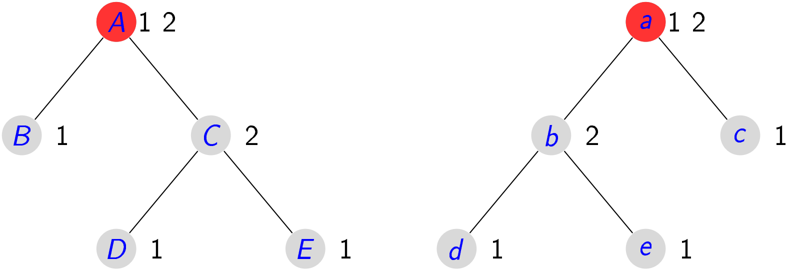
AHU algorithm example

Example



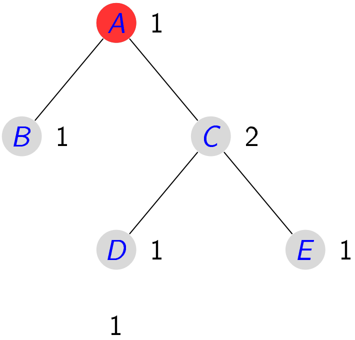
AHU algorithm example

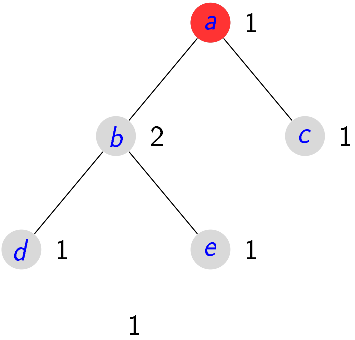
Example



AHU algorithm example

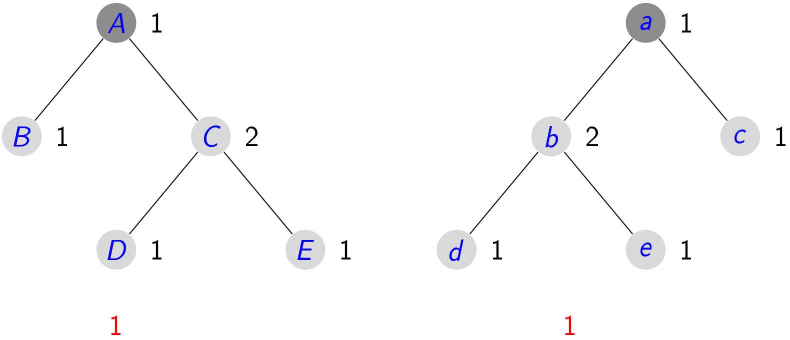
Example





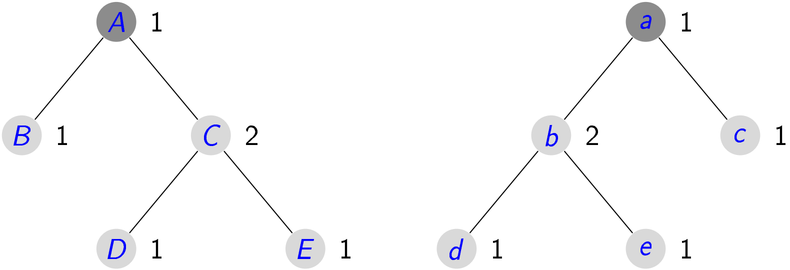
AHU algorithm example

Example



AHU algorithm example

Example



OK

Resume

- We have three unsuccessful tries to construct complete tree isomorphism invariant.
- We discussed $O(|V|^2)$ version of AHU algorithm.
- We discussed ways of improvement of AHU algorithm to make it work in $O(|V|)$ time.

Thank you for your attention!

Any questions?