Elementary Introduction to Graph Theory

(01EIG 2025/2026)

Lecture 3

Francesco Dolce
dolcefra@fit.cvut.cz

October 10, 2025

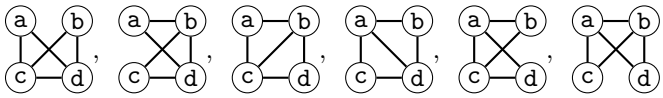
updated: October 10, 2025

PDF available at the address: dolcefra.pages.fit/ens/2526/EIG-lecture-03.pdf

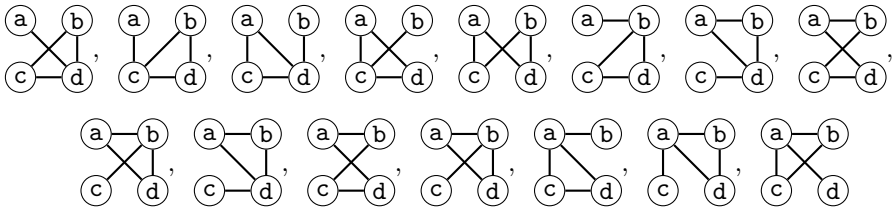
Solution of Exercise in previous Lecture. The number of connected graphs on 4 vertices, S_4, is 38. These 38 graphs are:

a) 1 complete graph;

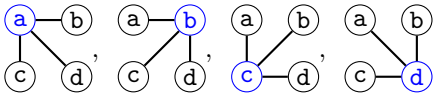
b) $6 = \binom{6}{1}$ graphs with one edge missing;



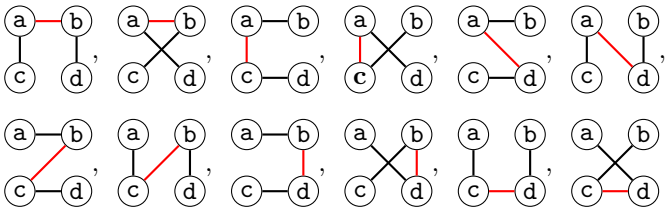
c) $15 = \binom{6}{2}$ graphs with two edges missing;



d) 4 graphs with one vertex with degree 3, connected to the other 3 vertices of degree 1;



e) $12 = 6 \cdot 2$ graphs that are paths of length 3, since we have 6 possibilities for the middle segment and 2 possibilities for the end segments;



1 Adjacency matrix

The *adjacency matrix* A_G of a simple graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the $n \times n$ matrix defined by

$$[A_G]_{i,j} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases}.$$

Clearly A_G is a non-negative symmetric matrix.

Example 1 Let $G = (\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}, \{\{\mathtt{a}, \mathtt{b}\}, \{\mathtt{a}, \mathtt{c}\}, \{\mathtt{a}, \mathtt{d}\}, \{\mathtt{b}, \mathtt{c}\}\})$. The adjacency matrix of G is

$$A_G = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Note that the definition of A_G can be generalised to non-simple graphs by replacing 1 with the number of parallel edges and counting each loop twice.

Theorem 1 *Let $k \in \mathbb{N}_0$ and let A_G be the adjacency matrix of a graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$. Then $\left[A_G^k\right]_{i,j}$ is the number of walks of length k with endpoints v_i and v_j in G.*

Proof. The case $k = 0$ is trivial since we have $A_G^0 = I_n$, and the only walks of length 0 are the ones connecting a vertex to itself.

Let us prove it by induction on $k \geq 1$.

($k = 1$) It follows directly from the definition of adjacency matrix, since a walk of length 1 is exactly an edge.

($k - 1 \to k$) The (i, j)-element of A_G^k is given by

$$\left[A_G^k\right]_{i,j} = \left[A_G^{k-1} \cdot A_G\right]_{i,j} = \sum_{\ell=1}^{n} \left[A_G^{k-1}\right]_{i,\ell} \cdot [A_G]_{\ell,j} = \sum_{\substack{\ell=1 \\ \{v_\ell, v_j\} \in E}}^{n} \left[A_G^{k-1}\right]_{i,\ell}$$

but $\left[A_G^{k-1}\right]_{i,\ell}$ is, by induction hypothesis, the number of walks of length $k-1$ with endpoints v_i and v_ℓ. Since we are summing only the walks that can be prolonged from v_ℓ to v_j (for every possibile choice of v_ℓ) we can conclude (see Figure 1).

■

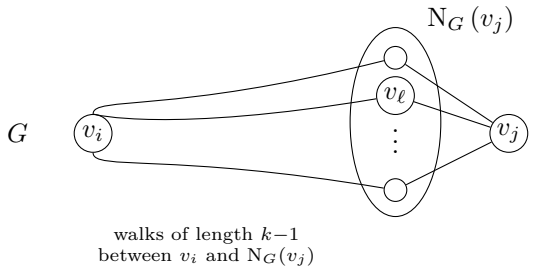


Figure 1: Illustration of the proof of Theorem 1.

Corollary 1 *A graph $G = (V, E)$ is connected if and only if $\sum_{k=0}^{\#V-1} A_G^k$ is a positive matrix.*

Proof.

($\Rightarrow$) Let $v_i, v_j \in V$. Since G is connected the two vertices are linked in G. Thus, there is a walk of length $k \leq \#V - 1$ in G with endpoints v_i and v_j. Hence, $\left[A_G^k\right]_{i,j} \geq 1$.

($\Leftarrow$) Let i, j with $0 < i, j < n$. Since $\left[\sum_{k=0}^{n-1} A_G^k\right]_{i,j} > 0$, there exists ℓ, with $0 \leq \ell \leq n-1$ such that $\left[A_G^\ell\right]_{i,j} \geq 1$. Thus, there is a walk of length ℓ in G with endpoints v_i and v_j.

■

Example 2 Let A_G be the adjacency matrix seen in Example 1. We have

$$A_G^2 = \begin{pmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad A_G^3 = \begin{pmatrix} 2 & 4 & 4 & 3 \\ 4 & 3 & 3 & 2 \\ 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 0 \end{pmatrix}.$$

Indeed, we have, e.g., three walks of length 2 from `a` to `a`, namely `(a,b,a)`, `(a,c,a)` and `(a,d,a)`; and four walks of length 3 from `a` to `c`, namely: `(a,b,a,c)`, `(a,c,a,c)`, `(a,c,b,c)` and `(a,d,a,c)`.

Notice also that

$$\sum_{k=0}^{3} A_G^k = I_4 + A_G + A_G^2 + A_G^3 = \begin{pmatrix} 6 & 6 & 6 & 4 \\ 6 & 6 & 5 & 3 \\ 6 & 5 & 5 & 2 \\ 4 & 3 & 2 & 2 \end{pmatrix}.$$

is a positive matrix.

2 Isomorphism of graphs

Two graphs $G = (V, E)$ and $H = (U, F)$ are *identical* if $V = U$ and $E = F$.

They are *isomorphic*, denoted $G \sim H$ (or $G \cong H$), if there exists a bijection $\varphi : V \to U$, called an *isomorphism*, such that for every $u, v \in V$

$$\{u, v\} \in E \quad \Leftrightarrow \quad \{\varphi(u), \varphi(v)\} \in F.$$

Clearly when $\varphi = \mathrm{id}$, the two graphs are identical.

Example 3 The graphs $G = \left(V, \binom{V}{2}\right)$ and $H = \left(U, \binom{U}{2}\right)$ with $V = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ and $U = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{t}\}$ are isomorphic (see Figure 2), with bijection, e.g.,

$$\varphi : \begin{cases} \mathtt{a} \mapsto \mathtt{x} \\ \mathtt{b} \mapsto \mathtt{y} \\ \mathtt{c} \mapsto \mathtt{z} \\ \mathtt{d} \mapsto \mathtt{t} \end{cases}$$

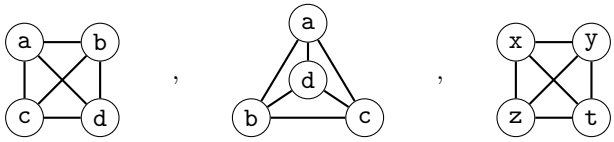


Figure 2: Three isomorphic graphs (the first two are identical).

Thus, two graphs are isomorphic if they are "the same" up to relabelling the vertices. Clearly, degree and distance are preserved under isomorphism.

Proposition 1 *Let $G = (V, E)$ and $H = (U, F)$ be two isomorphic graphs with isomorphism φ. Then, for every $u, v \in V$ we have*

1. $d_G(v) = d_H(\varphi(v))$,
2. $d_G(u, v) = d_H(\varphi(u), \varphi(v))$.

Proof.

1. From the definition of isomorphism it follows that $u \in \mathrm{N}_G(v)$ if and only if $\varphi(u) \in \mathrm{N}_H(\varphi(v))$. Thus the cardinality of the two sets, i.e., the two degrees $d_G(v)$ and $d_H(\varphi(v))$, coincide.
2. Exercise.

■

Example 4 The graphs represented in Figure 3 are isomorphic. A possible bijection is

$$\varphi : \begin{cases} \mathtt{a} \mapsto 1 \\ \mathtt{b} \mapsto 4 \\ \mathtt{c} \mapsto 2 \\ \mathtt{d} \mapsto 5 \\ \mathtt{e} \mapsto 3 \\ \mathtt{f} \mapsto 6 \end{cases} .$$

One can check, e.g., that $d_G(\mathtt{a}) = d_H(1) = 3$, and that $d_G(\mathtt{a}, \mathtt{e}) = d_H(1, 3) = 2$.

Figure 3: Two isomorphic graphs.

We do not normally distinguish between isomorphic graphs. When dealing with a graph up to isomorphism we can forget the labels in the drawing.

If two graphs are isomorphic, then clearly they have the same order and the same size. The opposite is not true.

Exercise. Show that the two unlabeled graphs in Figure 4, which have both order 4 and size 3, are not isomorphic.

Figure 4: Two non-isomorphic graphs with the same order and size.

The graph isomorphism problem (deciding whether two graphs are isomorpic) is $\mathcal{NP}$, but it is not known to belong to either $\mathcal{P}$ or $\mathcal{NP}$-complete.

3 Tree isomorphism problem

We will see a polynomial-time algorithm for isomorphism of trees. To do it we proceed in three steps, giving three different algorithms:

1. algorithm for isomorphism of ordered trees;
2. algorithm for isomorphism of rooted trees;
3. general algorithm for isomorphism of trees.

Two trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ are isomorphic, denoted

$$T_1 \sim T_2,$$

if the two trees are isomorphic as graphs, i.e., there exists a bijective mapping $\varphi : V_1 \to V_2$ such that for every $u, v \in V_1$

$$\{u, v\} \in E_1 \quad \Leftrightarrow \quad \{\varphi(u), \varphi(v)\} \in E_2.$$

Two rooted trees $T_1(r_1)$ and $T_2(r_2)$ are isomorphic, denoted

$$T_1(r_1) \sim_R T_2(r_2),$$

if $T_1 \sim T_2$ with isomorphism φ, and the morphism is sending root to root, i.e., $\varphi(r_1) = r_2$.

Two ordered trees $T_1(r_1, \preceq_1)$ and $T_2(r_2, \preceq_2)$ are isomorphic, denoted

$$T_1(r_1, \preceq_1) \sim_O T_2(r_2, \preceq_2),$$

if $T_1(r_1) \sim_R T_2(r_2)$ with isomorphism φ, and the mapping φ preserves the partial order of children of each vertex, i.e., for every $u, v \in V_1$

$$u \preceq_1 v \quad \Leftrightarrow \quad \varphi(u) \preceq_2 \varphi(v).$$

Example 5 The two trees T_1 and T_2 in Figure 5 are isomorphic. Indeed an isomorphism is given by

$$\varphi : \begin{cases} \mathtt{a} \mapsto \mathtt{E} \\ \mathtt{b} \mapsto \mathtt{D} \\ \mathtt{c} \mapsto \mathtt{C} \\ \mathtt{D} \mapsto \mathtt{B} \\ \mathtt{e} \mapsto \mathtt{A} \end{cases} .$$

As rooted trees, though, $T_1(\mathtt{d})$ and $T_2(\mathtt{C})$ are not isomorphic since there is no isomorphism sending $\mathtt{d}$ to $\mathtt{C}$ (why?)

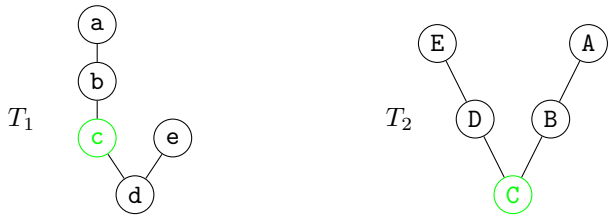


Figure 5: $T_1 \sim T_2$ but $T_1(\mathtt{d}) \not\sim_R T_2(\mathtt{C})$.

Example 6 The two rooted trees $T_3(\mathtt{r})$ and $T_4(\mathtt{R})$ in Figure 6 are isomorphic. Indeed it is enough to consider the bijection

$$\varphi : \begin{cases} \mathtt{r} \mapsto \mathtt{R} \\ \mathtt{a} \mapsto \mathtt{C} \\ \mathtt{b} \mapsto \mathtt{B} \\ \mathtt{c} \mapsto \mathtt{D} \\ \mathtt{d} \mapsto \mathtt{A} \end{cases} .$$

They are not isomorphic as ordered trees (why?)

The two ordered trees $T_4(\mathtt{R}, \preceq_4)$ and $T_5(\mathtt{s}, \preceq_5)$ in the same figure are isomorphic (find the isomorphism!).

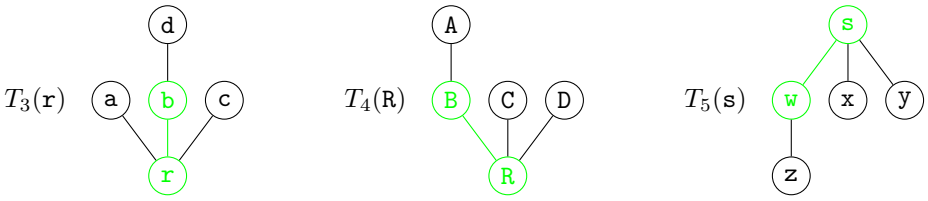


Figure 6: Three ordered trees.

4 Isomorphism of ordered trees

We will code a ordered tree by a sequence of 0s and 1s such that:

- the encoding uses only properties fixed by the mapping φ;
- the code is uniquely (up to isomorphism) decodable.

These two properties of the encoding imply that two ordered trees are isomorphic if and only if their codes are the same.

We code an ordered tree $T(r, \preceq)$ in the following way:

- each leaf gets the code $\mathtt{01}$;
- given a node v with children $v_1 \prec v_2 \prec \ldots \prec v_k$ having encoding $C(v_1), C(v_2), \ldots, C(v_k)$, we define $C(v) = \mathtt{0}\, C(v_1)C(v_2)\cdots C(v_k)\, \mathtt{1}$;
- $C(T) = C(r)$, i.e., the code of a tree coincides with the code of its root.

Example 7 Let $T_3(\mathtt{r}, \preceq_3)$ be the ordered tree in Example 6. Then

$$C(\mathtt{a}) = C(\mathtt{c}) = C(\mathtt{d}) = \mathtt{01}; \qquad C(\mathtt{b}) = \mathtt{0}.C(\mathtt{d}).\mathtt{1} = \mathtt{0011};$$

and

$$C(T_3) = C(\mathtt{r}) = \mathtt{0}.C(\mathtt{a}).C(\mathtt{b}).C(\mathtt{c}).\mathtt{1} = \mathtt{0010011011}.$$

One can check that $C(T_4) = C(T_5) = \mathtt{0001101011} \neq C(T_3)$.

We can also decode a code C of an ordered tree $T(r, \prec)$. If $C(T) = \mathtt{0}w\mathtt{1}$, then w corresponds to the concatenations of codes of ordered trees with roots corresponding to the children of r.

In particular, if $w = C(v_1)C(v_2)\cdots C(v_k)$, then $C(v_1)$ is the shortest prefix of w with the same number of 0s and 1s. We can cut $C(v_1)$ and continue analogously.

Example 8 Let $C(T_3) = \mathtt{0.01001101.1}$ as in Example **??**. The shortest prefix of $w = \mathtt{01\,0011\,01}$ having the same number of 0s and 1s is $C(v_1) = \mathtt{01}$. Similarly $C(v_2) = \mathtt{0011}$ and $C(v_3) = \mathtt{01}$. The vertex v_2 has a child v_4 with encoding $C(v_4) = \mathtt{01}$. Thus v_1, v_3, v_4 are leaves and the T_3 is isomorphic to the ordered tree $T = (\{r, v_1, v_2, v_3, v_4\}, \{\{r, v_1\}, \{r, v_2\}, \{r, v_3\}, \{v_2, v_4\}\})$.

5 Isomorphism of rooted trees

We would like to modify the encoding of ordered trees so that it can be used for deciding the isomorphism of rooted trees.

Which properties used in the previous case can we also use in the case of rooted trees?

1) a vertex is a root, ✓

2) a vertex is a leaf, ✓

3) a vertex is a child of a vertex, ✓

4) children of the same vertex are ordered. ✘

So we have to compensate for the fact that children of a vertex do not have a fixed (under isomorphism) order. To do that we:

- choose an ordering $\leq$ on strings over $\{\mathtt{0},\mathtt{1}\}$ (e.g., lexicographical or radix),
- if v has children $v_1, v_2, \ldots, v_k$ such that $C(v_1) \leq C(v_2) \leq \ldots \leq C(v_k)$, then we define $C(v) = \mathtt{0}\, C(v_1)C(v_2)\cdots C(v_k)\, \mathtt{1}$.

Example 9 Let $T_1(\mathtt{d})$ and $T_2(\mathtt{C})$ be the rooted tree in Example 5. We have

$$C(\mathtt{a}) = C(\mathtt{e}) = \mathtt{01}, \quad C(\mathtt{b}) = \mathtt{0011} \quad \text{and} \quad C(\mathtt{C}) = \mathtt{000110}.$$

If we use the lexicographic order $\leq_{lex}$ we have

$$C(\mathtt{c}) = \mathtt{000111} <_{lex} \mathtt{01} = C(\mathtt{e}),$$

so

$$C(T_1) = C(\mathtt{d}) = \mathtt{0}.C(\mathtt{c}).C(\mathtt{e}).\mathtt{1} = \mathtt{0000111011}$$

Similarly, we have

$$C(\mathtt{A}) = C(\mathtt{E}) = \mathtt{01}, \quad C(\mathtt{B}) = C(\mathtt{D}) = \mathtt{0011}, \quad \text{and} C(\mathtt{D}) = \mathtt{0011} = C(\mathtt{B}).$$

Hence, $C(T_2) = C(\mathtt{C}) = \mathtt{0001100111} \neq C(T_1)$.

On the other hand, one can check that the three trees in Example 6 have all the same encoding (using the lexicographic order) $\mathtt{0001101011}$.

Exercise. Prove, using the radix order, that:

a) $T_1(\mathtt{d}) \not\sim_R T_2(\mathtt{c})$,

b) $T_3(\mathtt{r}) \sim_R T_4(\mathtt{R}) \sim_R T_5(\mathtt{s})$.

6 Isomorphism of general trees

Again, we would like to adapt the encoding of rooted trees to decide isomorphism of general trees. Problems in this case, i.e., in switching from rooted to general trees, are much more serious. Indeed, we don't have the root, thus we cannot establish the child/parent relationship between nodes.

Therefore, we have to find a suitable (i.e., preserved under isomorphism) substitute for the root. To do that we use the centre of a tree.

While for a general graph $G = (V, E)$ the centre of G can be any subset of V (see, e.g., $\mathcal{C}(C_3)$ where the centre is the entire set of vertices), in the case of trees the situation is simpler.

Theorem 2 *Let $T = (V, E)$ be a tree. Then either $\mathcal{C}(T) = \{x\}$ or $\mathcal{C}(T) = \{x, y\}$, with $x, y \in V$. Moreover, in the latter case we have $\{x, y\} \in E$.*

Example 10 Let T_1 and T_2 be the trees in Example 5, and T_3, T_4, T_5 the ones in Example 6. One can check that

$$\mathcal{C}(T_1) = \{\texttt{c}\},\ \ \mathcal{C}(T_2) = \{\texttt{C}\},\ \ \mathcal{C}(T_3) = \{\texttt{b}, \texttt{r}\},\ \ \mathcal{C}(T_4) = \{\texttt{B}, \texttt{R}\},\ \ \mathcal{C}(T_5) = \{\texttt{w}, \texttt{s}\}.$$

Since the only graph property used in the definition of centre of a tree is the distance, and since distance of vertices is preserved under the isomorphism, we can use the centre of a tree instead of the root. In particular, when the centre is a singleton we consider it as the root of the tree. When the centre is a set of cardinality two, we use these two vertices as roots of two new trees obtained as subgraphs of the original one.

Algorithm 1: `IsoTrees`(T_1, T_2)

Input: Two trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$
Output: `TRUE` if $T_1 \sim T_2$; `FALSE` if not

Find centres $\mathcal{C}(T_1)$ and $\mathcal{C}(T_2)$
if $\#\mathcal{C}(T_1) \neq \#\mathcal{C}(T_2)$ **then**
  return `FALSE` // $T_1 \not\sim T_2$
else if $\Big(\mathcal{C}(T_1) = \{x\}$ *and* $\mathcal{C}(T_2) = \{y\}\Big)$ **then**
  return `IsoRootTrees`$(T_1(x), T_2(y))$ // $T_1 \sim T_2 \ \Leftrightarrow\ T_1(x) \sim_R T_2(y)$
else
  Let $\mathcal{C}(T_1) = \{x_1, x_2\}$ and $\mathcal{C}(T_2) = \{y_1, y_2\}$.
  We use 4 rooted trees:
  $T_1^{(1)}(x_1)$: the component of $T_1 \setminus \{x_1, x_2\}$ containing x_1;
  $T_1^{(2)}(x_2)$: the component of $T_1 \setminus \{x_1, x_2\}$ containing x_2;
  $T_2^{(1)}(y_1)$: the component of $T_2 \setminus \{y_1, y_2\}$ containing y_1.
  $T_2^{(2)}(y_2)$: the component of $T_2 \setminus \{y_1, y_2\}$ containing y_2.
  return $\Big($ $\big($`IsoRootTrees`$(T_1^{(1)}(x_1), T_2^{(1)}(y_1))$ `and` `IsoRootTrees`$(T_1^{(2)}(x_2), T_2^{(2)}(y_2))\big)$ `or` $\big($`IsoRootTrees`$(T_1^{(1)}(x_1), T_2^{(2)}(y_2))$ `and` `IsoRootTrees`$(T_1^{(2)}(x_2), T_2^{(1)}(y_1))\big)$ $\Big)$
  // $T_1 \sim T_2 \ \Leftrightarrow\ \Big(T_1^{(1)}(x_1) \sim_R T_2^{(1)}(y_1) \ \wedge T_1^{(2)}(x_2) \sim_R T_2^{(2)}(y_2)\Big) \vee \Big(T_1^{(1)}(x_1) \sim_R T_2^{(2)}(y_2) \ \wedge T_1^{(2)}(x_2) \sim_R T_2^{(1)}(y_1)\Big)$

Example 11 Let T_1, T_2, T_3, and T_4 as in Examples 5 and 6.

- Clearly $T_1 \not\sim T_3$ since $\#\mathcal{C}(T_1) \neq \#\mathcal{C}(T_3)$.
- We have $T_1 \sim T_2$ since one can easily check that $T_1(\texttt{c}) \sim_R T_2(\texttt{C})$.

- We have $T_3 \sim T_4$ since one can build the rooted trees $T_3^{(1)}(\mathtt{b}), T_3^{(2)}(\mathtt{r}), T_4^{(1)}(\mathtt{B})$ and $T_4^{(2)}(\mathtt{R})$ (see Figure 7) and easily check that $T_3^{(1)}(\mathtt{b}) \sim_R T_4^{(1)}(\mathtt{B})$ and $T_3^{(2)}(\mathtt{r}) \sim_R T_4^{(2)}(\mathtt{R})$.

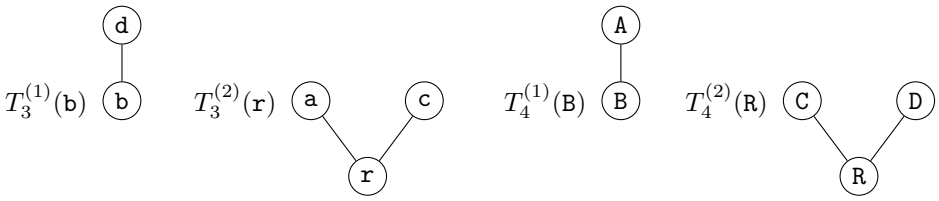


Figure 7: The trees obtained by deleting the central edges in T_3 and T_4.