Searching for a Graph Isomorphism

G.A.Noskov

2015

Outline

- Formulation of the GI
- Searching for a GI
- SGI versus GI
- Algorithm for SGI
- Disease
- A gift
- Solving GI a.a.surely
- E-test
- Questions
- Conclusion

Formulation of the GI

- A graph is a pair $G = (V, E)$ of sets such that $E \subseteq \binom{V}{2}$;
  thus, the elements of E (the edges) are 2-element subsets of V.

Formulation of the GI

- A graph is a pair $G = (V, E)$ of sets such that $E \subseteq \binom{V}{2}$; thus, the elements of E (the edges) are 2-element subsets of V.
- An isomorphism f from G to H is a bijective mapping (a permutation) from V_G to V_H such that the edge connections are respected, i.e. $f(E_G) = E_H$.

Formulation of the GI

- A graph is a pair $G = (V, E)$ of sets such that $E \subseteq \binom{V}{2}$; thus, the elements of E (the edges) are 2-element subsets of V.
- An isomorphism f from G to H is a bijective mapping (a permutation) from V_G to V_H such that the edge connections are respected, i.e. $f(E_G) = E_H$.
- $\mathcal{G}_n$ = all graphs on $[n] = \{1, \ldots, n\}$.

Formulation of the GI

- A graph is a pair $G = (V, E)$ of sets such that $E \subseteq \binom{V}{2}$; thus, the elements of E (the edges) are 2-element subsets of V.
- An isomorphism f from G to H is a bijective mapping (a permutation) from V_G to V_H such that the edge connections are respected, i.e. $f(E_G) = E_H$.
- $\mathcal{G}_n$ = all graphs on $[n] = \{1, \ldots, n\}$.
- Let S_n denote the symmetric group on $[n]$. S_n acts also on 2^V - the set of 2-element subsets of V. For every permutation $f \in S_n$ and every graph $G = ([n], E)$ define $fG = ([n], fE)$.

Formulation of the GI

- A graph is a pair $G = (V, E)$ of sets such that $E \subseteq \binom{V}{2}$; thus, the elements of E (the edges) are 2-element subsets of V.
- An isomorphism f from G to H is a bijective mapping (a permutation) from V_G to V_H such that the edge connections are respected, i.e. $f(E_G) = E_H$.
- $\mathcal{G}_n$ = all graphs on $[n] = \{1, \ldots, n\}$.
- Let S_n denote the symmetric group on $[n]$. S_n acts also on 2^V - the set of 2-element subsets of V. For every permutation $f \in S_n$ and every graph $G = ([n], E)$ define $fG = ([n], fE)$.
- Thus S_n acts on the set $\mathcal{G}_n$ and isomorphism classes in $\mathcal{G}_n$ are precisely the orbits of this action $S_n \curvearrowright \mathcal{G}_n$.

Formulation of the GI

- A graph is a pair $G = (V, E)$ of sets such that $E \subseteq \binom{V}{2}$; thus, the elements of E (the edges) are 2-element subsets of V.
- An isomorphism f from G to H is a bijective mapping (a permutation) from V_G to V_H such that the edge connections are respected, i.e. $f(E_G) = E_H$.
- $\mathcal{G}_n$ = all graphs on $[n] = \{1, \ldots, n\}$.
- Let S_n denote the symmetric group on $[n]$. S_n acts also on 2^V - the set of 2-element subsets of V. For every permutation $f \in S_n$ and every graph $G = ([n], E)$ define $fG = ([n], fE)$.
- Thus S_n acts on the set $\mathcal{G}_n$ and isomorphism classes in $\mathcal{G}_n$ are precisely the orbits of this action $S_n \curvearrowright \mathcal{G}_n$.

Graph Isomorphism problem

- GI-problem: Decide whether two given graphs $G, H \in \mathcal{G}_n$ are isomorphic, i.e. are in the same orbit of the action $S_n \curvearrowright \mathcal{G}_n$.

Graph Isomorphism problem

- GI-problem: Decide whether two given graphs $G, H \in \mathcal{G}_n$ are isomorphic, i.e. are in the same orbit of the action $S_n \curvearrowright \mathcal{G}_n$.
- Outstanding particular case: Is GI in the class $\mathcal{P}$?

Graph Isomorphism problem

- GI-problem: Decide whether two given graphs $G, H \in \mathcal{G}_n$ are isomorphic, i.e. are in the same orbit of the action $S_n \curvearrowright \mathcal{G}_n$.
- Outstanding particular case: Is GI in the class $\mathcal{P}$?
- $\mathcal{P}$ denotes the class of problem that have decision algorithms with running time bounded by some polynomial (in the input length).

Graph Isomorphism problem

- GI-problem: Decide whether two given graphs $G, H \in \mathcal{G}_n$ are isomorphic, i.e. are in the same orbit of the action $S_n \curvearrowright \mathcal{G}_n$.
- Outstanding particular case: Is GI in the class $\mathcal{P}$?
- $\mathcal{P}$ denotes the class of problem that have decision algorithms with running time bounded by some polynomial (in the input length).

- MathSciNet: Publications results for "Anywhere=(Graph isomorphism)- 554.

- MathSciNet: Publications results for "Anywhere=(Graph isomorphism)- 554.
- First publication:

- MathSciNet: Publications results for "Anywhere=(Graph isomorphism)- 554.
- First publication:
- Corneil, Derek Gordon GRAPH ISOMORPHISM. Thesis (Ph.D.)-University of Toronto (Canada). 1968.

- MathSciNet: Publications results for "Anywhere=(Graph isomorphism)- 554.
- First publication:
- Corneil, Derek Gordon GRAPH ISOMORPHISM. Thesis (Ph.D.)-University of Toronto (Canada). 1968.

SGI

- Booth, Kellogg S.; Colbourn, C. J. (1977), Problems polynomially equivalent to graph isomorphism.

SGI

- Booth, Kellogg S.; Colbourn, C. J. (1977), Problems polynomially equivalent to graph isomorphism.
- "... The graph isomorphism is to decide, given two graphs, whether an isomorphism exists. A related problem is to find an isomorphism and explicitly present it, whenever one exists. The equivalence of these two problem has been known for many years..."

SGI

- Booth, Kellogg S.; Colbourn, C. J. (1977), Problems polynomially equivalent to graph isomorphism.
- "... The graph isomorphism is to decide, given two graphs, whether an isomorphism exists. A related problem is to find an isomorphism and explicitly present it, whenever one exists. The equivalence of these two problem has been known for many years..."
- C.M. Hoffmann. Group-Theoretic Algorithms and Graph Isomorphism. LNCS 136, Springer-Verlag 1982.

SGI

- Booth, Kellogg S.; Colbourn, C. J. (1977), Problems polynomially equivalent to graph isomorphism.
- "... The graph isomorphism is to decide, given two graphs, whether an isomorphism exists. A related problem is to find an isomorphism and explicitly present it, whenever one exists. The equivalence of these two problem has been known for many years..."
- C.M. Hoffmann. Group-Theoretic Algorithms and Graph Isomorphism. LNCS 136, Springer-Verlag 1982.
- P1: Given two graphs G and H with n vertices each, decide whether they are isomorphic.
  P2: Given two graphs G and H, decide whether they are isomorphic, and if so, construct an isomorphism from G to H.

SGI

- Booth, Kellogg S.; Colbourn, C. J. (1977), Problems polynomially equivalent to graph isomorphism.
- "... The graph isomorphism is to decide, given two graphs, whether an isomorphism exists. A related problem is to find an isomorphism and explicitly present it, whenever one exists. The equivalence of these two problem has been known for many years..."
- C.M. Hoffmann. Group-Theoretic Algorithms and Graph Isomorphism. LNCS 136, Springer-Verlag 1982.
- P1: Given two graphs G and H with n vertices each, decide whether they are isomorphic.
  P2: Given two graphs G and H, decide whether they are isomorphic, and if so, construct an isomorphism from G to H.
- $P1 \equiv^{pol} P2$!

SGI

- Booth, Kellogg S.; Colbourn, C. J. (1977), Problems polynomially equivalent to graph isomorphism.
- "... The graph isomorphism is to decide, given two graphs, whether an isomorphism exists. A related problem is to find an isomorphism and explicitly present it, whenever one exists. The equivalence of these two problem has been known for many years..."
- C.M. Hoffmann. Group-Theoretic Algorithms and Graph Isomorphism. LNCS 136, Springer-Verlag 1982.
- P1: Given two graphs G and H with n vertices each, decide whether they are isomorphic.
  P2: Given two graphs G and H, decide whether they are isomorphic, and if so, construct an isomorphism from G to H.
- $P1 \equiv^{pol} P2$!

SGI

Рис.: A.N.Rybalov, stating SGI

SGI

Рис.: A.N.Rybalov, stating SGI

- The input set for searching graph isomorphism problem SGI is the set of all pairs of isomorphic graphs.

SGI

Рис.: A.N.Rybalov, stating SGI

- The input set for searching graph isomorphism problem SGI is the set of all pairs of isomorphic graphs.
- Given two isomorphic graphs G and H, one needs to construct an isomorphism between G and H.

SGI

Рис.: A.N.Rybalov, stating SGI

- The input set for searching graph isomorphism problem SGI is the set of all pairs of isomorphic graphs.
- Given two isomorphic graphs G and H, one needs to construct an isomorphism between G and H.

SGI and promise problems

- Promise problems (Even- Selman -Yacobi (1984))

SGI and promise problems

- Promise problems (Even- Selman -Yacobi (1984))
- For a polynomially bounded relation R, the search (promise) problem is given x that has a solution (i.e., there exists y such that $(x, y) \in R$) to find such a solution (i.e., find a y such that $(x, y) \in R$)).

SGI and promise problems

- Promise problems (Even- Selman -Yacobi (1984))
- For a polynomially bounded relation R, the search (promise) problem is given x that has a solution (i.e., there exists y such that $(x, y) \in R$) to find such a solution (i.e., find a y such that $(x, y) \in R$)).
- Hence, the promise is that x has a solution (i.e., y such that $(x, y) \in R$), and nothing is required in case x has no solution.

SGI and promise problems

- Promise problems (Even- Selman -Yacobi (1984))
- For a polynomially bounded relation R, the search (promise) problem is given x that has a solution (i.e., there exists y such that $(x, y) \in R$) to find such a solution (i.e., find a y such that $(x, y) \in R$)).
- Hence, the promise is that x has a solution (i.e., y such that $(x, y) \in R$), and nothing is required in case x has no solution.
- SGI: $R(G, H, \phi) \Leftrightarrow \phi : G \simeq H$. Given (G, H) such that $\exists \phi : G \simeq H$ to find ϕ.

SGI and promise problems

- Promise problems (Even- Selman -Yacobi (1984))
- For a polynomially bounded relation R, the search (promise) problem is given x that has a solution (i.e., there exists y such that $(x, y) \in R$) to find such a solution (i.e., find a y such that $(x, y) \in R$)).
- Hence, the promise is that x has a solution (i.e., y such that $(x, y) \in R$), and nothing is required in case x has no solution.
- SGI: $R(G, H, \phi) \Leftrightarrow \phi : G \simeq H$. Given (G, H) such that $\exists \phi : G \simeq H$ to find ϕ.

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI.$
  - Proof

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - Ascent.

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - **Ascent**.
  - $V_G = \{g_1, \ldots, g_n\}$.

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - **Ascent**.
  - $V_G = \{g_1, \ldots, g_n\}$.
  - Construct (in polynomial time) $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - **Ascent**.
  - $V_G = \{g_1, \ldots, g_n\}$.
  - Construct (in polynomial time) $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$
    - and $h_1, \ldots, h_n \in V_H$ such that:

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - **Ascent**.
  - $V_G = \{g_1, \ldots, g_n\}$.
  - Construct (in polynomial time) $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$
  - and $h_1, \ldots, h_n \in V_H$ such that:
  - 1) $G_i \simeq H_i$ for all i,

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - **Ascent**.
  - $V_G = \{g_1, \ldots, g_n\}$.
  - Construct (in polynomial time) $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$
  - and $h_1, \ldots, h_n \in V_H$ such that:
  - 1) $G_i \simeq H_i$ for all i,
  - 2) Every isomorphism between G_i and H_i takes G_{i-1} to H_{i-1} for all $i = 1, \ldots, n$,

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - **Ascent**.
  - $V_G = \{g_1, \ldots, g_n\}$.
  - Construct (in polynomial time) $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$
  - and $h_1, \ldots, h_n \in V_H$ such that:
  - 1) $G_i \simeq H_i$ for all i,
  - 2) Every isomorphism between G_i and H_i takes G_{i-1} to H_{i-1} for all $i = 1, \ldots, n$,
  - 3) Every isomorphism between G_i and H_i takes g_i to h_i for all $i = 1, \ldots, n$.

SGI is reducible to GI

- Theorem
  $SGI \leq^{pol} GI$.
  - Proof
  - **Ascent**.
  - $V_G = \{g_1, \ldots, g_n\}$.
  - Construct (in polynomial time) $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$
  - and $h_1, \ldots, h_n \in V_H$ such that:
  - 1) $G_i \simeq H_i$ for all i,
  - 2) Every isomorphism between G_i and H_i takes G_{i-1} to H_{i-1} for all $i = 1, \ldots, n$,
  - 3) Every isomorphism between G_i and H_i takes g_i to h_i for all $i = 1, \ldots, n$.

SGI

- Descent

SGI

- **Descent**
- The map $g_i \mapsto h_i$ is an isomorphism between G and H!

SGI

- **Descent**
- The map $g_i \mapsto h_i$ is an isomorphism between G and H!
- Indeed, by 1) $\exists \phi : G_n \to H_n$.

SGI

- **Descent**
- The map $g_i \mapsto h_i$ is an isomorphism between G and H!
- Indeed, by 1) $\exists \phi : G_n \to H_n$.
- By 2) ϕ takes g_n to h_n.

SGI

- **Descent**
- The map $g_i \mapsto h_i$ is an isomorphism between G and H!
- Indeed, by 1) $\exists \phi : G_n \to H_n$.
- By 2) ϕ takes g_n to h_n.
- $\phi|_{G_{n-1}}$ takes G_{n-1} to H_{n-1} by 2) and takes g_{n-1} to h_{n-1} by 3).

SGI

- **Descent**
- The map $g_i \mapsto h_i$ is an isomorphism between G and H!
- Indeed, by 1) $\exists \phi : G_n \to H_n$.
- By 2) ϕ takes g_n to h_n.
- $\phi|_{G_{n-1}}$ takes G_{n-1} to H_{n-1} by 2) and takes g_{n-1} to h_{n-1} by 3).
- Then $\phi|_{G_{n-1}}$takes G_{n-2} to H_{n-2} and we can continue this descent process until obtaining that ϕ takes every g_i to h_i. Hence the map $g_i \mapsto h_i$ is an isomorphism in question.

SGI

- **Descent**
- The map $g_i \mapsto h_i$ is an isomorphism between G and H!
- Indeed, by 1) $\exists \phi : G_n \to H_n$.
- By 2) ϕ takes g_n to h_n.
- $\phi|_{G_{n-1}}$ takes G_{n-1} to H_{n-1} by 2) and takes g_{n-1} to h_{n-1} by 3).
- Then $\phi|_{G_{n-1}}$takes G_{n-2} to H_{n-2} and we can continue this descent process until obtaining that ϕ takes every g_i to h_i. Hence the map $g_i \mapsto h_i$ is an isomorphism in question.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

- Since $G \simeq H$, $\exists h$ s.t. $G_1 \simeq H_1 = H_1(h)$.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

- Since $G \simeq H$, $\quad \exists h$ s.t. $G_1 \simeq H_1 = H_1(h)$.
- Find h by applying $\mathcal{A}$ to all n pairs $(G_1, H_1(h))$.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

- Since $G \simeq H$, $\quad \exists h$ s.t. $G_1 \simeq H_1 = H_1(h)$.
- Find h by applying $\mathcal{A}$ to all n pairs $(G_1, H_1(h))$.
- Set $h_1 = h$.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

- Since $G \simeq H$, $\quad \exists h$ s.t. $G_1 \simeq H_1 = H_1(h)$.
- Find h by applying $\mathcal{A}$ to all n pairs $(G_1, H_1(h))$.
- Set $h_1 = h$.
- g_1, h_1 are unique vertices of highest degree in G_1, H_1. Hence every isomorphism between G_i and H_i takes g_i to h_i.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

- Since $G \simeq H, \quad \exists h$ s.t. $G_1 \simeq H_1 = H_1(h)$.
- Find h by applying $\mathcal{A}$ to all n pairs $(G_1, H_1(h))$.
- Set $h_1 = h$.
- g_1, h_1 are unique vertices of highest degree in G_1, H_1. Hence every isomorphism between G_i and H_i takes g_i to h_i.
- Every isomorphism between G_1 and H_1 takes G_0 to H_0. Thus 1)-3) hold for $i = 1$.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

- Since $G \simeq H$, $\quad \exists h$ s.t. $G_1 \simeq H_1 = H_1(h)$.
- Find h by applying $\mathcal{A}$ to all n pairs $(G_1, H_1(h))$.
- Set $h_1 = h$.
- g_1, h_1 are unique vertices of highest degree in G_1, H_1. Hence every isomorphism between G_i and H_i takes g_i to h_i.
- Every isomorphism between G_1 and H_1 takes G_0 to H_0. Thus 1)-3) hold for $i = 1$.
- Continuing, find the desired $h_1, \ldots, h_n \in H$ and $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$.

SGI: Constructing the tower

- Let $\mathcal{A}$ be a polynomial algorithm, solving GI.
- Fix $k_1 \in K_{n+1}$ and consider

$$G_1 = G \cup_{g_1=k_1} K_{n+1}, \quad H_1(h) = H \cup_{h=k_1} K_{n+1}.$$

- Since $G \simeq H, \quad \exists h$ s.t. $G_1 \simeq H_1 = H_1(h)$.
- Find h by applying $\mathcal{A}$ to all n pairs $(G_1, H_1(h))$.
- Set $h_1 = h$.
- g_1, h_1 are unique vertices of highest degree in G_1, H_1. Hence every isomorphism between G_i and H_i takes g_i to h_i.
- Every isomorphism between G_1 and H_1 takes G_0 to H_0. Thus 1)-3) hold for $i = 1$.
- Continuing, find the desired $h_1, \ldots, h_n \in H$ and $G = G_0 < G_1 < \cdots < G_n$, $H = H_0 < H_1 < \cdots < H_n$.

Probabilisltic algorithm for SGI

- Theorem (A.N.Rybalov, 2015)

  If there exists a polynomial generic algorithm for SGI, then there exists a polynomial probabilistic algorithm computing SGI for all inputs.

Probabilisltic algorithm for SGI

- **Theorem (A.N.Rybalov, 2015)**

  If there exists a polynomial generic algorithm for SGI, then there exists a polynomial probabilistic algorithm computing SGI for all inputs.

- **Theorem (G.A.Noskov,2015)**

  There is a polynomial algorithm solving SGI on an asymptotically almost all inputs.

Probabilisltic algorithm for SGI

- **Theorem (A.N.Rybalov, 2015)**

  If there exists a polynomial generic algorithm for SGI, then there exists a polynomial probabilistic algorithm computing SGI for all inputs.

- **Theorem (G.A.Noskov,2015)**

  There is a polynomial algorithm solving SGI on an asymptotically almost all inputs.

SGI:Graph invariant τ

- The type $\tau(v) \in \mathbb{N}^*$ of a vertex v is the string of degrees of vertices of $N(v)$ in non-decreasing order.

SGI:Graph invariant τ

- The type $\tau(v) \in \mathbb{N}^*$ of a vertex v is the string of degrees of vertices of $N(v)$ in non-decreasing order.
- In detail, $\tau(v) = (d_1, ..., d_{d(v)})$, where $d_1 \leq d_2 \leq \ldots \leq d_{d(v)}$.

SGI:Graph invariant τ

- The type $\tau(v) \in \mathbb{N}^*$ of a vertex v is the string of degrees of vertices of $N(v)$ in non-decreasing order.
- In detail, $\tau(v) = (d_1, ..., d_{d(v)})$, where $d_1 \leq d_2 \leq \ldots \leq d_{d(v)}$.
- The type is preserved under graph isomorphisms: if $\phi : G \to H$ is a graph isomorphism then $\tau(\phi(v)) = \tau(v)$ for all $v \in V(G)$, i.e. $\tau \circ \phi = \tau$.

SGI:Graph invariant τ

- The type $\tau(v) \in \mathbb{N}^*$ of a vertex v is the string of degrees of vertices of $N(v)$ in non-decreasing order.
- In detail, $\tau(v) = (d_1, ..., d_{d(v)})$, where $d_1 \leq d_2 \leq \ldots \leq d_{d(v)}$.
- The type is preserved under graph isomorphisms: if $\phi : G \to H$ is a graph isomorphism then $\tau(\phi(v)) = \tau(v)$ for all $v \in V(G)$, i.e. $\tau \circ \phi = \tau$.
- The type τ_G of G is the Lex ordered sequence $(\tau(v_1), \ldots, \tau(v_n))$, $n = |V|$ of all types of all vertices.

SGI:Graph invariant τ

- The type $\tau(v) \in \mathbb{N}^*$ of a vertex v is the string of degrees of vertices of $N(v)$ in non-decreasing order.
- In detail, $\tau(v) = (d_1, ..., d_{d(v)})$, where $d_1 \leq d_2 \leq \ldots \leq d_{d(v)}$.
- The type is preserved under graph isomorphisms: if $\phi : G \to H$ is a graph isomorphism then $\tau(\phi(v)) = \tau(v)$ for all $v \in V(G)$, i.e. $\tau \circ \phi = \tau$.
- The type τ_G of G is the Lex ordered sequence $(\tau(v_1), \ldots, \tau(v_n))$, $n = |V|$ of all types of all vertices.
- The type function $G \mapsto \tau_G$ is a graph invariant , i.e. $\tau_G = \tau_H$ for isomorphic graphs G, H.

SGI:Graph invariant τ

- The type $\tau(v) \in \mathbb{N}^*$ of a vertex v is the string of degrees of vertices of $N(v)$ in non-decreasing order.
- In detail, $\tau(v) = (d_1, ..., d_{d(v)})$, where $d_1 \leq d_2 \leq \ldots \leq d_{d(v)}$.
- The type is preserved under graph isomorphisms: if $\phi : G \to H$ is a graph isomorphism then $\tau(\phi(v)) = \tau(v)$ for all $v \in V(G)$, i.e. $\tau \circ \phi = \tau$.
- The type τ_G of G is the Lex ordered sequence $(\tau(v_1), \ldots, \tau(v_n))$, $n = |V|$ of all types of all vertices.
- The type function $G \mapsto \tau_G$ is a graph invariant , i.e. $\tau_G = \tau_H$ for isomorphic graphs G, H.

Oblique graphs

- J. Schreyer, H.Walther, L.S. Melnikov, Vertex-oblique graphs, Discrete Math. 307 (2007).

Oblique graphs

- J. Schreyer, H.Walther, L.S. Melnikov, Vertex-oblique graphs, Discrete Math. 307 (2007).
- A graph G is said to be oblique if the map $\tau : V \to \mathbb{N}^*$ is injective or, in other words, there are no distinct vertices $u, v \in V(G)$ such that $\tau(v) = \tau(w)$. pause
- If a graph G is oblique then its type vector $\tau_G = (\tau(v_1), \ldots, \tau(v_n))$ is unique in the sense that $(\tau(v_1), \ldots, \tau(v_n)) = (\tau(w_1), \ldots, \tau(w_n))$ implies that $v_i = w_i$ for all i.

Oblique graphs

- J. Schreyer, H.Walther, L.S. Melnikov, Vertex-oblique graphs, Discrete Math. 307 (2007).
- A graph G is said to be oblique if the map $\tau : V \to \mathbb{N}^*$ is injective or, in other words, there are no distinct vertices $u, v \in V(G)$ such that $\tau(v) = \tau(w)$. pause
- If a graph G is oblique then its type vector $\tau_G = (\tau(v_1), \ldots, \tau(v_n))$ is unique in the sense that $(\tau(v_1), \ldots, \tau(v_n)) = (\tau(w_1), \ldots, \tau(w_n))$ implies that $v_i = w_i$ for all i.

- **Lemma (retrieving an isomorphism)**

  Let $\tau_G = (\tau(g_1), \ldots, \tau(g_n)), \tau_H = (\tau(h_1), \ldots, \tau(h_n))$ be the type-vectors of G, H respectively. If G is oblique and $G \simeq H$ then the map $g_i \mapsto h_i$ is the isomorphism of G onto H and there is no other isomorphism between G and H. In particular, every oblique graph has a trivial automorphism group.

Oblique graphs

- J. Schreyer, H.Walther, L.S. Melnikov, Vertex-oblique graphs, Discrete Math. 307 (2007).
- A graph G is said to be oblique if the map $\tau : V \to \mathbb{N}^*$ is injective or, in other words, there are no distinct vertices $u, v \in V(G)$ such that $\tau(v) = \tau(w)$. pause
- If a graph G is oblique then its type vector $\tau_G = (\tau(v_1), \ldots, \tau(v_n))$ is unique in the sense that $(\tau(v_1), \ldots, \tau(v_n)) = (\tau(w_1), \ldots, \tau(w_n))$ implies that $v_i = w_i$ for all i.

- **Lemma (retrieving an isomorphism)**

  Let $\tau_G = (\tau(g_1), \ldots, \tau(g_n)), \tau_H = (\tau(h_1), \ldots, \tau(h_n))$ be the type-vectors of G, H respectively. If G is oblique and $G \simeq H$ then the map $g_i \mapsto h_i$ is the isomorphism of G onto H and there is no other isomorphism between G and H. In particular, every oblique graph has a trivial automorphism group.

SGI: Partial polynomial algorithm $\mathcal{A}$

- Input set: $\mathcal{I} = \cup\, \mathcal{I}_n$, where
  $\mathcal{I}_n = \{(G, H) : G, H \in \mathcal{G}_n,\ G \simeq H\}$.

SGI: Partial polynomial algorithm $\mathcal{A}$

- Input set: $\mathcal{I} = \cup\, \mathcal{I}_n$, where
  $\mathcal{I}_n = \{(G, H) : G, H \in \mathcal{G}_n,\ G \simeq H\}$.
- The problem: given $(G, H) \in \mathcal{I}_n$ to construct $\pi \in S_n$ such that $\pi G = H$.

SGI: Partial polynomial algorithm $\mathcal{A}$

- Input set: $\mathcal{I} = \cup\, \mathcal{I}_n$, where $\mathcal{I}_n = \{(G, H) : G, H \in \mathcal{G}_n,\ G \simeq H\}$.
- The problem: given $(G, H) \in \mathcal{I}_n$ to construct $\pi \in S_n$ such that $\pi G = H$.
- Given isomorphic graphs $G, H \in \mathcal{G}_n$, the algorithm first computes their type-vectors

$$\tau_G = (\tau(g_1), \ldots, \tau(g_n)), \tau_H = (\tau(h_1), \ldots, \tau(h_n)).$$

SGI: Partial polynomial algorithm $\mathcal{A}$

- Input set: $\mathcal{I} = \cup\, \mathcal{I}_n$, where $\mathcal{I}_n = \{(G, H) : G, H \in \mathcal{G}_n,\ G \simeq H\}$.
- The problem: given $(G, H) \in \mathcal{I}_n$ to construct $\pi \in S_n$ such that $\pi G = H$.
- Given isomorphic graphs $G, H \in \mathcal{G}_n$, the algorithm first computes their type-vectors

$$\tau_G = (\tau(g_1), \ldots, \tau(g_n)), \tau_H = (\tau(h_1), \ldots, \tau(h_n)).$$

- If the coordinates of at least one vector are not distinct, the algorithm answers "?"(don't know).

SGI: Partial polynomial algorithm $\mathcal{A}$

- Input set: $\mathcal{I} = \cup\, \mathcal{I}_n$, where $\mathcal{I}_n = \{(G, H) : G, H \in \mathcal{G}_n,\ G \simeq H\}$.
- The problem: given $(G, H) \in \mathcal{I}_n$ to construct $\pi \in S_n$ such that $\pi G = H$.
- Given isomorphic graphs $G, H \in \mathcal{G}_n$, the algorithm first computes their type-vectors

$$\tau_G = (\tau(g_1), \ldots, \tau(g_n)), \tau_H = (\tau(h_1), \ldots, \tau(h_n)).$$

- If the coordinates of at least one vector are not distinct, the algorithm answers "?"(don't know).
- Otherwise G, H are both oblique and $\mathcal{A}$ checks whether the map $\phi : g_i \mapsto h_i$ is an isomorphism or not.

SGI: Partial polynomial algorithm $\mathcal{A}$

- Input set: $\mathcal{I} = \cup\, \mathcal{I}_n$, where $\mathcal{I}_n = \{(G, H) : G, H \in \mathcal{G}_n,\ G \simeq H\}$.
- The problem: given $(G, H) \in \mathcal{I}_n$ to construct $\pi \in S_n$ such that $\pi G = H$.
- Given isomorphic graphs $G, H \in \mathcal{G}_n$, the algorithm first computes their type-vectors

$$\tau_G = (\tau(g_1), \ldots, \tau(g_n)), \tau_H = (\tau(h_1), \ldots, \tau(h_n)).$$

- If the coordinates of at least one vector are not distinct, the algorithm answers "?"(don't know).
- Otherwise G, H are both oblique and $\mathcal{A}$ checks whether the map $\phi : g_i \mapsto h_i$ is an isomorphism or not.
- If this map turns out to be an isomorphism, then the algorithm gives ϕ as the answer to the problem.

SGI: Partial polynomial algorithm $\mathcal{A}$

- Input set: $\mathcal{I} = \cup\, \mathcal{I}_n$, where $\mathcal{I}_n = \{(G, H) : G, H \in \mathcal{G}_n,\ G \simeq H\}$.
- The problem: given $(G, H) \in \mathcal{I}_n$ to construct $\pi \in S_n$ such that $\pi G = H$.
- Given isomorphic graphs $G, H \in \mathcal{G}_n$, the algorithm first computes their type-vectors

$$\tau_G = (\tau(g_1), \ldots, \tau(g_n)), \tau_H = (\tau(h_1), \ldots, \tau(h_n)).$$

- If the coordinates of at least one vector are not distinct, the algorithm answers "?"(don't know).
- Otherwise G, H are both oblique and $\mathcal{A}$ checks whether the map $\phi : g_i \mapsto h_i$ is an isomorphism or not.
- If this map turns out to be an isomorphism, then the algorithm gives ϕ as the answer to the problem.

SGI: Failure probability

- $\mathcal{A}$ gives the right answer for the set of inputs $\mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)$ and fails on the complement set $\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)$, where $\mathcal{S}_n = \mathcal{G}_n - \mathcal{G}_n^o$.

SGI: Failure probability

- $\mathcal{A}$ gives the right answer for the set of inputs $\mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)$ and fails on the complement set $\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)$, where $\mathcal{S}_n = \mathcal{G}_n - \mathcal{G}_n^o$.

- **Theorem**

  The failure probability of the algorithm $\mathcal{A}$ tends to zero as n tends to infinity. In other words, the algorithm solves SGI with a high probability.

SGI: Failure probability

- $\mathcal{A}$ gives the right answer for the set of inputs $\mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)$ and fails on the complement set $\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)$, where $\mathcal{S}_n = \mathcal{G}_n - \mathcal{G}_n^o$.

- **Theorem**

  The failure probability of the algorithm $\mathcal{A}$ tends to zero as n tends to infinity. In other words, the algorithm solves SGI with a high probability.

  - J.Schreyer (2007): The property of a graph $G \in \mathcal{G}_n$ to be oblique holds asymptotically almost surely, i.e.

$$\lim_{n\to\infty} \frac{|\mathcal{G}_n^o|}{|\mathcal{G}_n|} = 1,$$

or, equivalently,

$$\lim_{n\to\infty} \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|} = 1.$$

SGI: Failure probability

- $\mathcal{A}$ gives the right answer for the set of inputs $\mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)$ and fails on the complement set $\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)$, where $\mathcal{S}_n = \mathcal{G}_n - \mathcal{G}_n^o$.

- Theorem

The failure probability of the algorithm $\mathcal{A}$ tends to zero as n tends to infinity. In other words, the algorithm solves SGI with a high probability.

  - J.Schreyer (2007): The property of a graph $G \in \mathcal{G}_n$ to be oblique holds asymptotically almost surely, i.e.

$$\lim_{n\to\infty} \frac{|\mathcal{G}_n^o|}{|\mathcal{G}_n|} = 1,$$

or, equivalently,

$$\lim_{n\to\infty} \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|} = 1.$$

SGI: Failure probability

- $\mathcal{A}$ gives the right answer for the set of inputs $\mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)$ and fails on the complement set $\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)$, where $\mathcal{S}_n = \mathcal{G}_n - \mathcal{G}_n^o$.

- Theorem

The failure probability of the algorithm $\mathcal{A}$ tends to zero as n tends to infinity. In other words, the algorithm solves SGI with a high probability.

  - J.Schreyer (2007): The property of a graph $G \in \mathcal{G}_n$ to be oblique holds asymptotically almost surely, i.e.

$$\lim_{n\to\infty} \frac{|\mathcal{G}_n^o|}{|\mathcal{G}_n|} = 1,$$

or, equivalently,

$$\lim_{n\to\infty} \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|} = 1.$$

SGI: Failure probability

- Consider $pr_1 : \mathcal{I}_n \to \mathcal{G}_n$.

SGI: Failure probability

- Consider $pr_1 : \mathcal{I}_n \to \mathcal{G}_n$.
- What are the preimages of pr_1?

SGI: Failure probability

- Consider $pr_1 : \mathcal{I}_n \to \mathcal{G}_n$.
- What are the preimages of pr_1?
- $\left|pr_1^{-1} G\right| \leq n!$ for every $G \in \mathcal{G}_n$.

SGI: Failure probability

- Consider $pr_1 : \mathcal{I}_n \to \mathcal{G}_n$.
- What are the preimages of pr_1?
- $\left|pr_1^{-1} G\right| \leq n!$ for every $G \in \mathcal{G}_n$.
- In case G is oblique we can say more:

$$\left|pr_1^{-1} G\right| = n!$$

SGI: Failure probability

- Consider $pr_1 : \mathcal{I}_n \to \mathcal{G}_n$.
- What are the preimages of pr_1?
- $\left|pr_1^{-1} G\right| \leq n!$ for every $G \in \mathcal{G}_n$.
- In case G is oblique we can say more:

$$\left|pr_1^{-1} G\right| = n!$$

- Indeed, $Aut\,(G) = 1$ and thus the orbit $S_n G$ consists of $n!$ graphs and the set $(G, S_n G)$ is the preimage of G under pr_1.

SGI: Failure probability

- Consider $pr_1 : \mathcal{I}_n \to \mathcal{G}_n$.
- What are the preimages of pr_1?
- $\left|pr_1^{-1} G\right| \leq n!$ for every $G \in \mathcal{G}_n$.
- In case G is oblique we can say more:

$$\left|pr_1^{-1} G\right| = n!$$

- Indeed, $Aut\,(G) = 1$ and thus the orbit $S_n G$ consists of $n!$ graphs and the set $(G, S_n G)$ is the preimage of G under pr_1.
- Conclude: $pr_1 : \mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o) \to \mathcal{G}_n^o$ has the fibers of cardinality $n!$, hence $|\mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)| = n!\,|\mathcal{G}_n^o|$ and so

$$|\mathcal{I}_n| \geq n!\,|\mathcal{G}_n^o|\,.$$

SGI: Failure probability

- Consider $pr_1 : \mathcal{I}_n \to \mathcal{G}_n$.
- What are the preimages of pr_1?
- $\left|pr_1^{-1} G\right| \leq n!$ for every $G \in \mathcal{G}_n$.
- In case G is oblique we can say more:

$$\left|pr_1^{-1} G\right| = n!$$

- Indeed, $Aut\,(G) = 1$ and thus the orbit $S_n G$ consists of $n!$ graphs and the set $(G, S_n G)$ is the preimage of G under pr_1.
- Conclude: $pr_1 : \mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o) \to \mathcal{G}_n^o$ has the fibers of cardinality $n!$, hence $|\mathcal{I}_n \cap (\mathcal{G}_n^o \times \mathcal{G}_n^o)| = n!\,|\mathcal{G}_n^o|$ and so

$$|\mathcal{I}_n| \geq n!\,|\mathcal{G}_n^o|\,.$$

SGI: Failure probability

- Considering the fibers of pr_1 over the set $\mathcal{S}_n$ we conclude that

$$|\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)| \leq n! \, |\mathcal{S}_n| \, .$$

SGI: Failure probability

- Considering the fibers of pr_1 over the set $\mathcal{S}_n$ we conclude that

$$|\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)| \leq n! \, |\mathcal{S}_n| \, .$$

- $$\frac{|\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)|}{|\mathcal{I}_n|} \leq \frac{n! \, |\mathcal{S}_n|}{n! \, |\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n| - |\mathcal{S}_n|}$$

SGI: Failure probability

- Considering the fibers of pr_1 over the set $\mathcal{S}_n$ we conclude that

$$|\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)| \leq n! \, |\mathcal{S}_n| \, .$$

- $$\frac{|\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)|}{|\mathcal{I}_n|} \leq \frac{n! \, |\mathcal{S}_n|}{n! \, |\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n| - |\mathcal{S}_n|}$$

- $$\frac{|\mathcal{S}_n|}{|\mathcal{G}_n|} \left(\frac{1}{1 - \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|}} \right) \to 0, \; n \to \infty.$$

SGI: Failure probability

- Considering the fibers of pr_1 over the set $\mathcal{S}_n$ we conclude that

$$|\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)| \leq n! \, |\mathcal{S}_n| \, .$$

- $$\frac{|\mathcal{I}_n \cap (\mathcal{S}_n \times \mathcal{S}_n)|}{|\mathcal{I}_n|} \leq \frac{n! \, |\mathcal{S}_n|}{n! \, |\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n^o|} = \frac{|\mathcal{S}_n|}{|\mathcal{G}_n| - |\mathcal{S}_n|}$$

- $$\frac{|\mathcal{S}_n|}{|\mathcal{G}_n|} \left(\frac{1}{1 - \frac{|\mathcal{S}_n|}{|\mathcal{G}_n|}} \right) \to 0, \;\; n \to \infty.$$

Disease

Read, Ronald C.; Corneil, Derek G. (1977), *The graph isomorphism disease, Journal of Graph Theory 1 (4): 339-363,*

Disease

Read, Ronald C.; Corneil, Derek G. (1977), *The graph isomorphism disease, Journal of Graph Theory 1 (4): 339-363,*

Disease

Read, Ronald C.; Corneil, Derek G. (1977), *The graph isomorphism disease, Journal of Graph Theory 1 (4): 339-363,*

Disease

Read, Ronald C.; Corneil, Derek G. (1977), *The graph isomorphism disease, Journal of Graph Theory 1 (4): 339-363,*

Signs that a problem is a Mathematical Disease (R.Lipton)

- 1. A problem must be easy to state to be a MD. Hodge-Conjecture (10^6 USD) will never be a disease.

Signs that a problem is a Mathematical Disease (R.Lipton)

- 1. A problem must be easy to state to be a MD.
  Hodge-Conjecture (10^6 USD) will never be a disease.
- 2. A problem must seem to be accessible, even to an amateur.
  Your reaction should be: that is open?

Signs that a problem is a Mathematical Disease (R.Lipton)

- 1. A problem must be easy to state to be a MD. Hodge-Conjecture (10^6 USD) will never be a disease.
- 2. A problem must seem to be accessible, even to an amateur. Your reaction should be: that is open?
- 3. The problem must seem to be easy. A problem must also have been repeatedly "solved"to be a true MD. A good MD usually has been "proved"many times – often by the same person.

Signs that a problem is a Mathematical Disease (R.Lipton)

- 1. A problem must be easy to state to be a MD. Hodge-Conjecture (10^6 USD) will never be a disease.
- 2. A problem must seem to be accessible, even to an amateur. Your reaction should be: that is open?
- 3. The problem must seem to be easy. A problem must also have been repeatedly "solved"to be a true MD. A good MD usually has been "proved"many times – often by the same person.
- If you see a paper in arXiv.org with many "updates"that is a good sign that the problem is a MD.

Signs that a problem is a Mathematical Disease (R.Lipton)

- 1. A problem must be easy to state to be a MD. Hodge-Conjecture (10^6 USD) will never be a disease.
- 2. A problem must seem to be accessible, even to an amateur. Your reaction should be: that is open?
- 3. The problem must seem to be easy. A problem must also have been repeatedly "solved"to be a true MD. A good MD usually has been "proved"many times – often by the same person.
- If you see a paper in arXiv.org with many "updates"that is a good sign that the problem is a MD.

GI as a gift

- Does P=NP? S.Smale: I sometimes consider this problem as a gift to mathematics from computer science.

GI as a gift

- Does P=NP? S.Smale: I sometimes consider this problem as a gift to mathematics from computer science.
- Diophant($\mathbb{Z}/2$): Given (as input) k polynomials in n variables with coefficients in $\mathbb{Z}/2$. Decide if there is a common zero $x \in (Z/2)^n$?

GI as a gift

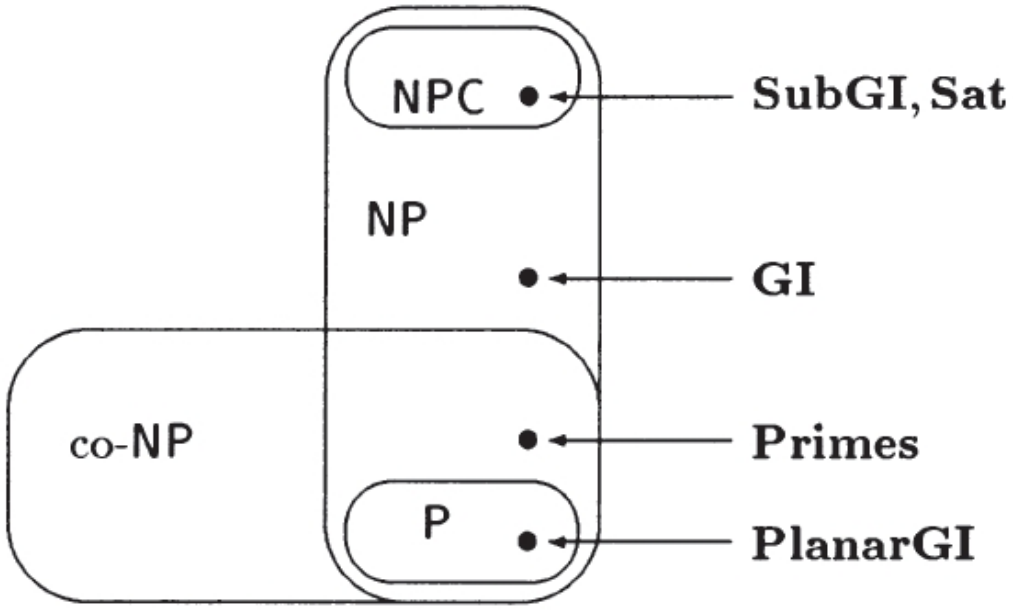
- Does P=NP? S.Smale: I sometimes consider this problem as a gift to mathematics from computer science.
- Diophant($\mathbb{Z}/2$): Given (as input) k polynomials in n variables with coefficients in $\mathbb{Z}/2$. Decide if there is a common zero $x \in (Z/2)^n$?
- Conjecture. There is no polynomial time algorithm over $Z/2$ deciding this problem. This is a reformulation of the classic conjecture $P \neq NP$.

GI as a gift

- Does P=NP? S.Smale: I sometimes consider this problem as a gift to mathematics from computer science.
- Diophant($\mathbb{Z}/2$): Given (as input) k polynomials in n variables with coefficients in $\mathbb{Z}/2$. Decide if there is a common zero $x \in (Z/2)^n$?
- Conjecture. There is no polynomial time algorithm over $Z/2$ deciding this problem. This is a reformulation of the classic conjecture $P \neq NP$.
- Questions. What is the reduction relation between GI and $Diophant(\mathbb{Z}/2)$? Specifically, can GI be polynomially reduced to $Diophant(\mathbb{Z}/2)$? Is $SeachDiophant(\mathbb{Z}/2)$ polynomially reducible to $Diophant(\mathbb{Z}/2)$?

GI as a gift

- Does P=NP? S.Smale: I sometimes consider this problem as a gift to mathematics from computer science.
- Diophant($\mathbb{Z}/2$): Given (as input) k polynomials in n variables with coefficients in $\mathbb{Z}/2$. Decide if there is a common zero $x \in (Z/2)^n$?
- Conjecture. There is no polynomial time algorithm over $Z/2$ deciding this problem. This is a reformulation of the classic conjecture $P \neq NP$.
- Questions. What is the reduction relation between GI and $Diophant(\mathbb{Z}/2)$? Specifically, can GI be polynomially reduced to $Diophant(\mathbb{Z}/2)$? Is $SeachDiophant(\mathbb{Z}/2)$ polynomially reducible to $Diophant(\mathbb{Z}/2)$?


NPC
SubGI, Sat
NP
GI
co-NP
Primes
P
PlanarGI

Algorithm LABEL (Babai, Erdos and Selkow (1980))

Algorithm LABEL (Babai, Erdos and Selkow (1980))

- Canonical labelling a graph G.

Algorithm LABEL (Babai, Erdos and Selkow (1980))

- Canonical labelling a graph G.
- Step 0: Input graph G and parameter $L = 3 \log_2 n$.

Algorithm LABEL (Babai, Erdos and Selkow (1980))

- Canonical labelling a graph G.
- Step 0: Input graph G and parameter $L = 3\log_2 n$.
- Step 1: Re-label the vertices of G so that they satisfy

$$d_G(v_1) \geq d_G(v_2) \geq \cdots \geq d_G(v_n)$$

:

Algorithm LABEL (Babai, Erdos and Selkow (1980))

- Canonical labelling a graph G.
- Step 0: Input graph G and parameter $L = 3\log_2 n$.
- Step 1: Re-label the vertices of G so that they satisfy

$$d_G(v_1) \geq d_G(v_2) \geq \cdots \geq d_G(v_n)$$

:

- If there exists $i < L$ such that $d_G(v_i) = d_G(v_{i+1})$, then FAIL.

Algorithm LABEL (Babai, Erdos and Selkow (1980))

- Canonical labelling a graph G.
- Step 0: Input graph G and parameter $L = 3\log_2 n$.
- Step 1: Re-label the vertices of G so that they satisfy

$$d_G(v_1) \geq d_G(v_2) \geq \cdots \geq d_G(v_n)$$

:

- If there exists $i < L$ such that $d_G(v_i) = d_G(v_{i+1})$, then FAIL.
- Step 2: For $i > L$ let

$$X_i = \{j \in \{1, 2, \ldots, L\} : \{v_i, v_j\} \in E_G.$$

Re-label vertices $v_{L+1}, v_{L+2}, ..., v_n$ so that these sets satisfy

$$X_{L+1} \succ X_{L+2} \succ \cdots \succ X_n,$$

where $\succ$ denotes lexicographic order.
If there exists $i < n$ such that $X_i = X_{i+1}$ then FAIL.

Algorithm LABEL (Babai, Erdos and Selkow (1980))

- Canonical labelling a graph G.
- Step 0: Input graph G and parameter $L = 3\log_2 n$.
- Step 1: Re-label the vertices of G so that they satisfy

$$d_G(v_1) \geq d_G(v_2) \geq \cdots \geq d_G(v_n)$$

:

- If there exists $i < L$ such that $d_G(v_i) = d_G(v_{i+1})$, then FAIL.
- Step 2: For $i > L$ let

$$X_i = \{j \in \{1, 2, \ldots, L\} : \{v_i, v_j\} \in E_G.$$

Re-label vertices $v_{L+1}, v_{L+2}, ..., v_n$ so that these sets satisfy

$$X_{L+1} \succ X_{L+2} \succ \cdots \succ X_n,$$

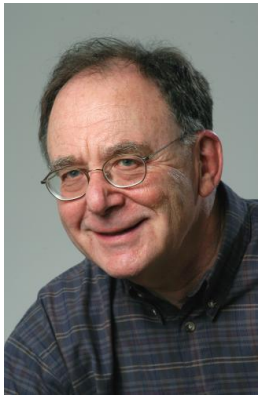
where $\succ$ denotes lexicographic order.
If there exists $i < n$ such that $X_i = X_{i+1}$ then FAIL.

R.Lipton and R.Karp

R.Lipton and R.Karp

- 
- R. Lipton, The beacon set approach to graph isomorphism, Research Report 135, Yale University, 1978.

R.Lipton and R.Karp

- 
- R. Lipton, The beacon set approach to graph isomorphism, Research Report 135, Yale University, 1978.
- 

R.Lipton and R.Karp

- 
- R. Lipton, The beacon set approach to graph isomorphism, Research Report 135, Yale University, 1978.
- 
- Richard M. Karp, Probabilistic analysis of a canonical numbering algorithm for graphs (1979).

R.Lipton and R.Karp

- 
- R. Lipton, The beacon set approach to graph isomorphism, Research Report 135, Yale University, 1978.
- 
- Richard M. Karp, Probabilistic analysis of a canonical numbering algorithm for graphs (1979).

Comparison of 3 algorithms

Algorithm	Execution Time	Frequency of Failure
Erdös-Babai	$O(n^2)$	$O(n^{-1/7})$
Lipton	$O(n^6 \log n)$	c^{-n}, $c > 1$
Karp	$O(n^2 \log n)$	$O(n^{3/2}2^{-n/2})$

Edge graph isomorphism test (E-test).

- The input set $\mathcal{I} = \cup_n(\mathcal{G}_n \times \mathcal{G}_n)$.

Edge graph isomorphism test (E-test).

- The input set $\mathcal{I} = \cup_n(\mathcal{G}_n \times \mathcal{G}_n)$.
- E-test: Given G_1, G_2
  1. Compare $|EG_1|, |EG_2|$,
  2. If $|EG_1| \neq |EG_2|$ the test says that G_1, G_2 are non-isomorphic.
  3. If $|EG_1| = |EG_2|$ the test says "I don't know".

Edge graph isomorphism test (E-test).

- The input set $\mathcal{I} = \cup_n(\mathcal{G}_n \times \mathcal{G}_n)$.
- E-test: Given G_1, G_2
  1. Compare $|EG_1|, |EG_2|$,
  2. If $|EG_1| \neq |EG_2|$ the test says that G_1, G_2 are non-isomorphic.
  3. If $|EG_1| = |EG_2|$ the test says "I don't know".
- The failure set of E-test is the set of pairs

$$F_n = \{(G_1, G_2) : G_1, G_2 \in \mathcal{G}(n), |EG_1| = |EG_2|\}.$$

Edge graph isomorphism test (E-test).

- The input set $\mathcal{I} = \cup_n (\mathcal{G}_n \times \mathcal{G}_n)$.
- E-test: Given G_1, G_2
  1. Compare $|EG_1|, |EG_2|$,
  2. If $|EG_1| \neq |EG_2|$ the test says that G_1, G_2 are non-isomorphic.
  3. If $|EG_1| = |EG_2|$ the test says "I don't know".
- The failure set of E-test is the set of pairs

$$F_n = \{(G_1, G_2) : G_1, G_2 \in \mathcal{G}(n), |EG_1| = |EG_2|\}.$$

- To measure the size of the failure set one needs an ensemble of distributions on $\mathcal{G}(n)$. For instance the binomial distribution P_n with parameters $N = \frac{n(n-1)}{2}, p$.

Edge graph isomorphism test (E-test).

- The input set $\mathcal{I} = \cup_n(\mathcal{G}_n \times \mathcal{G}_n)$.
- E-test: Given G_1, G_2
  1. Compare $|EG_1|, |EG_2|$,
  2. If $|EG_1| \neq |EG_2|$ the test says that G_1, G_2 are non-isomorphic.
  3. If $|EG_1| = |EG_2|$ the test says "I don't know".
- The failure set of E-test is the set of pairs

$$F_n = \{(G_1, G_2) : G_1, G_2 \in \mathcal{G}(n), |EG_1| = |EG_2|\}.$$

- To measure the size of the failure set one needs an ensemble of distributions on $\mathcal{G}(n)$. For instance the binomial distribution P_n with parameters $N = \frac{n(n-1)}{2}, p$.
- $P_n(G) = p^k q^{N-k}$, where $k = |E|, q = 1 - p, 0 < p < 1$.

Edge graph isomorphism test (E-test).

- The input set $\mathcal{I} = \cup_n(\mathcal{G}_n \times \mathcal{G}_n)$.
- E-test: Given G_1, G_2
  1. Compare $|EG_1|, |EG_2|$,
  2. If $|EG_1| \neq |EG_2|$ the test says that G_1, G_2 are non-isomorphic.
  3. If $|EG_1| = |EG_2|$ the test says "I don't know".
- The failure set of E-test is the set of pairs

$$F_n = \{(G_1, G_2) : G_1, G_2 \in \mathcal{G}(n), |EG_1| = |EG_2|\}.$$

- To measure the size of the failure set one needs an ensemble of distributions on $\mathcal{G}(n)$. For instance the binomial distribution P_n with parameters $N = \frac{n(n-1)}{2}, p$.
- $P_n(G) = p^k q^{N-k}$, where $k = |E|, q = 1 - p, 0 < p < 1$.

Failure probability

- Theorem (G.A.Noskov-A.N.Rybalov,2014)

  Let F_n be the failure set of the E-test on the set of inputs $\mathcal{G}(n,p) \times \mathcal{G}(n,p)$, $n \in \mathbb{N}$, $0 < p \leq \frac{1}{2}$.

Failure probability

- **Theorem (G.A.Noskov-A.N.Rybalov,2014)**

  Let F_n be the failure set of the E-test on the set of inputs $\mathcal{G}(n,p) \times \mathcal{G}(n,p)$, $n \in \mathbb{N}$, $0 < p \leq \frac{1}{2}$.

  1. *If $\lim_{n\to\infty} np = 0$, then $F = \cup F_n$ happens a.a.s., i.e. the failure probability $P(F_n)$ tends to 1 as n tends to infinity,*

Failure probability

- **Theorem (G.A.Noskov-A.N.Rybalov,2014)**

  Let F_n be the failure set of the E-test on the set of inputs $\mathcal{G}(n,p) \times \mathcal{G}(n,p)$, $n \in \mathbb{N}$, $0 < p \leq \frac{1}{2}$.

  1. *If $\lim_{n\to\infty} np = 0$, then $F = \cup F_n$ happens a.a.s., i.e. the failure probability $P(F_n)$ tends to 1 as n tends to infinity,*
  2. *If $\lim_{n\to\infty} np = \lambda$, $\lambda > 0$, then $P(F_n) = \Pi(\lambda)_2^2$, where $\Pi(\lambda)$ is the Poisson distribution vector and $0 < \Pi(\lambda)_2^2 < \min\left\{e^{\lambda^2-2\lambda}, 1\right\}$,*

Failure probability

- **Theorem (G.A.Noskov-A.N.Rybalov,2014)**

  Let F_n be the failure set of the E-test on the set of inputs $\mathcal{G}(n,p) \times \mathcal{G}(n,p)$, $n \in \mathbb{N}$, $0 < p \leq \frac{1}{2}$.

  1. *If $\lim_{n\to\infty} np = 0$, then $F = \cup F_n$ happens a.a.s., i.e. the failure probability $P(F_n)$ tends to 1 as n tends to infinity,*
  2. *If $\lim_{n\to\infty} np = \lambda$, $\lambda > 0$, then $P(F_n) = \Pi(\lambda)_2^2$, where $\Pi(\lambda)$ is the Poisson distribution vector and $0 < \Pi(\lambda)_2^2 < \min\left\{e^{\lambda^2-2\lambda}, 1\right\}$,*
  3. *If $\lim_{n\to\infty} np = \infty$, then F is negligible (i.e. $P(F_n) \to 0$ as $n \to \infty$) and, moreover, $P(F_n) \leq \frac{10}{\sqrt{Npq}}$ for $N = \binom{n}{2} \geq 10$.*

Failure probability

- **Theorem (G.A.Noskov-A.N.Rybalov,2014)**

  Let F_n be the failure set of the E-test on the set of inputs $\mathcal{G}(n,p) \times \mathcal{G}(n,p)$, $n \in \mathbb{N}$, $0 < p \leq \frac{1}{2}$.

  1. *If $\lim_{n\to\infty} np = 0$, then $F = \cup F_n$ happens a.a.s., i.e. the failure probability $P(F_n)$ tends to 1 as n tends to infinity,*
  2. *If $\lim_{n\to\infty} np = \lambda$, $\lambda > 0$, then $P(F_n) = \Pi(\lambda)_2^2$, where $\Pi(\lambda)$ is the Poisson distribution vector and $0 < \Pi(\lambda)_2^2 < \min\left\{e^{\lambda^2-2\lambda}, 1\right\}$,*
  3. *If $\lim_{n\to\infty} np = \infty$, then F is negligible (i.e. $P(F_n) \to 0$ as $n \to \infty$) and, moreover, $P(F_n) \leq \frac{10}{\sqrt{Npq}}$ for $N = \binom{n}{2} \geq 10$.*
  4. *If, in case 3, p is constant, then $P(F_n)$ is asymptotic to $\frac{1}{n\sqrt{2\pi pq}}$.*

Failure probability

- **Theorem (G.A.Noskov-A.N.Rybalov,2014)**

  Let F_n be the failure set of the E-test on the set of inputs $\mathcal{G}(n,p) \times \mathcal{G}(n,p)$, $n \in \mathbb{N}$, $0 < p \leq \frac{1}{2}$.

  1. *If $\lim_{n\to\infty} np = 0$, then $F = \cup F_n$ happens a.a.s., i.e. the failure probability $P(F_n)$ tends to 1 as n tends to infinity,*
  2. *If $\lim_{n\to\infty} np = \lambda$, $\lambda > 0$, then $P(F_n) = \Pi(\lambda)_2^2$, where $\Pi(\lambda)$ is the Poisson distribution vector and $0 < \Pi(\lambda)_2^2 < \min\left\{e^{\lambda^2-2\lambda}, 1\right\}$,*
  3. *If $\lim_{n\to\infty} np = \infty$, then F is negligible (i.e. $P(F_n) \to 0$ as $n \to \infty$) and, moreover, $P(F_n) \leq \frac{10}{\sqrt{Npq}}$ for $N = \binom{n}{2} \geq 10$.*
  4. *If, in case 3, p is constant, then $P(F_n)$ is asymptotic to $\frac{1}{n\sqrt{2\pi pq}}$.*

Proof: Case $np(n) \to \lambda > 0$

- By the Poisson theorem for every natural k

$$b(k; n, p) \to \pi_k, n \to \infty,$$

Proof: Case $np(n) \to \lambda > 0$

- By the Poisson theorem for every natural k

$$b(k; n, p) \to \pi_k, n \to \infty,$$

- where

$$\pi_k = e^{-\lambda} \frac{\lambda^k}{k!}, k = 0, 1, 2, \ldots.$$

Proof: Case $np(n) \to \lambda > 0$

- By the Poisson theorem for every natural k

$$b(k; n, p) \to \pi_k, n \to \infty,$$

- where

$$\pi_k = e^{-\lambda} \frac{\lambda^k}{k!}, k = 0, 1, 2, \ldots.$$

- That is, $B(n, p) \to \Pi(\lambda)$ coordinatewise, where

$$\Pi(\lambda) = (\pi_0(\lambda), ..., \pi_k(\lambda), ...),$$

is the Poisson distribution vector.

Proof: Case $np(n) \to \lambda > 0$

- By the Poisson theorem for every natural k

$$b(k; n, p) \to \pi_k, n \to \infty,$$

- where

$$\pi_k = e^{-\lambda} \frac{\lambda^k}{k!}, k = 0, 1, 2, \ldots.$$

- That is, $B(n, p) \to \Pi(\lambda)$ coordinatewise, where

$$\Pi(\lambda) = (\pi_0(\lambda), ..., \pi_k(\lambda), ...),$$

is the Poisson distribution vector.

- Using the result of Yu.V.Prohorov (1953) we deduce

$$\lim_{n \to \infty} |B(n, p)|_2 = |\Pi(\lambda)|_2.$$

Proof: Case $np(n) \to \lambda > 0$

- By the Poisson theorem for every natural k

$$b(k; n, p) \to \pi_k, n \to \infty,$$

- where

$$\pi_k = e^{-\lambda} \frac{\lambda^k}{k!}, k = 0, 1, 2, \ldots.$$

- That is, $B(n, p) \to \Pi(\lambda)$ coordinatewise, where

$$\Pi(\lambda) = (\pi_0(\lambda), ..., \pi_k(\lambda), ...),$$

  is the Poisson distribution vector.
- Using the result of Yu.V.Prohorov (1953) we deduce

$$\lim_{n \to \infty} |B(n, p)|_2 = |\Pi(\lambda)|_2 .$$

Case $p = const$

Case $p = const$

- $$P_n(F_n) = \sum_{i=0}^{N} \binom{N}{i}^2 p^{2i} q^{2N-2i} =$$

$$q^{2N} S_N\left(\left(\frac{p}{q}\right)^2\right),$$

where $S_m(t) = \sum_{i=0}^{m} \binom{m}{i}^2 t^i$.

Case $p = const$

- $$P_n(F_n) = \sum_{i=0}^{N} \binom{N}{i}^2 p^{2i} q^{2N-2i} =$$

$$q^{2N} S_N\left(\left(\frac{p}{q}\right)^2\right),$$

where $S_m(t) = \sum_{i=0}^{m} \binom{m}{i}^2 t^i$.

Case $p = const$

- In its turn the quadratic binomial sum can be expressed in terms of Legendre polynomial. Precisely, setting $t = \left(\frac{p}{q}\right)^2$ we obtain

Case $p = const$

- In its turn the quadratic binomial sum can be expressed in terms of Legendre polynomial. Precisely, setting $t = \left(\frac{p}{q}\right)^2$ we obtain
- $$P_n(F_n) = (1-2p)^N L_N\left(\frac{1+\left(\frac{p^2}{q^2}\right)}{1-\left(\frac{p^2}{q^2}\right)}\right),$$

  where

Case $p = const$

- In its turn the quadratic binomial sum can be expressed in terms of Legendre polynomial. Precisely, setting $t = \left(\frac{p}{q}\right)^2$ we obtain
- $$P_n(F_n) = (1-2p)^N L_N\left(\frac{1+\left(\frac{p^2}{q^2}\right)}{1-\left(\frac{p^2}{q^2}\right)}\right),$$

  where
- $$L_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left(x^2-1\right)^n, \ (n \geq 1).$$

Case $p = const$

- In its turn the quadratic binomial sum can be expressed in terms of Legendre polynomial. Precisely, setting $t = \left(\frac{p}{q}\right)^2$ we obtain
- $$P_n(F_n) = (1-2p)^N L_N\left(\frac{1+\left(\frac{p^2}{q^2}\right)}{1-\left(\frac{p^2}{q^2}\right)}\right),$$

  where
- $$L_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left(x^2-1\right)^n, \ \ (n \geq 1).$$

Case $p = const$

- To determine the asymptotic of $L_N\left(\frac{1+\left(\frac{p}{q}\right)^2}{1-\left(\frac{p}{q}\right)^2}\right)$ we use the Laplace-Heine formula

$$L_N(x) \sim \frac{1}{\sqrt{2\pi N}\sqrt[4]{x^2-1}}\left(x+\sqrt{x^2-1}\right)^{N+\frac{1}{2}}, \quad (|x|>1)$$

Case $p = const$

- To determine the asymptotic of $L_N\left(\frac{1+\left(\frac{p}{q}\right)^2}{1-\left(\frac{p}{q}\right)^2}\right)$ we use the Laplace-Heine formula

$$L_N(x) \sim \frac{1}{\sqrt{2\pi N}\sqrt[4]{x^2-1}}\left(x+\sqrt{x^2-1}\right)^{N+\frac{1}{2}}, \quad (|x|>1)$$

Questions

- What is the asymptotic of the failure probability of the **E**-test in case of planar graphs, graphs of bounded degree, some other interesting classes of graphs?

Questions

- What is the asymptotic of the failure probability of the **E**-test in case of planar graphs, graphs of bounded degree, some other interesting classes of graphs?
- The model $\mathcal{G}(n, m)$ consists of graphs on the set $[n]$ with m edges. The edge test fails here from the very beginning. The right modification would be a test, based on counting the number of 2-walks. What is its failure probability? Note, that the number of 2-walks equals $\sum_{i=1}^{n} \binom{d_i}{2}$, where d_i is the degree of the vertex i.

Questions

- What is the asymptotic of the failure probability of the **E**-test in case of planar graphs, graphs of bounded degree, some other interesting classes of graphs?
- The model $\mathcal{G}(n, m)$ consists of graphs on the set $[n]$ with m edges. The edge test fails here from the very beginning. The right modification would be a test, based on counting the number of 2-walks. What is its failure probability? Note, that the number of 2-walks equals $\sum_{i=1}^{n} \binom{d_i}{2}$, where d_i is the degree of the vertex i.
- What is the asymptotic failure probability of tests based on counting of k-walks with fixed $k \geq 2$? The same with counting all walks?

Questions

- What is the asymptotic of the failure probability of the **E**-test in case of planar graphs, graphs of bounded degree, some other interesting classes of graphs?
- The model $\mathcal{G}(n, m)$ consists of graphs on the set $[n]$ with m edges. The edge test fails here from the very beginning. The right modification would be a test, based on counting the number of 2-walks. What is its failure probability? Note, that the number of 2-walks equals $\sum_{i=1}^{n} \binom{d_i}{2}$, where d_i is the degree of the vertex i.
- What is the asymptotic failure probability of tests based on counting of k-walks with fixed $k \geq 2$? The same with counting all walks?

Questions

- Is there a "trivial test"for the Maximal Clique problem (to which the GI reduces polynomially).

Questions

- Is there a "trivial test"for the Maximal Clique problem (to which the GI reduces polynomially).
- What is the generic case complexity of Hidden Subgroup Problem?

Questions

- Is there a "trivial test"for the Maximal Clique problem (to which the GI reduces polynomially).
- What is the generic case complexity of Hidden Subgroup Problem?
- It is not true that the generic case complexity is invariant under polynomial equivalence. Thus it makes sense to ask whether the following problems admit generically polynomial algorithms: Search Isomorphism, Graph Automorphism, Order of the Automorphism Group, Number of Graph Isomorphisms, Double Coset Membership, Croup Factorization, Group Intersection, Setwise Stabilizer.

Questions

- Is there a "trivial test"for the Maximal Clique problem (to which the GI reduces polynomially).
- What is the generic case complexity of Hidden Subgroup Problem?
- It is not true that the generic case complexity is invariant under polynomial equivalence. Thus it makes sense to ask whether the following problems admit generically polynomial algorithms: Search Isomorphism, Graph Automorphism, Order of the Automorphism Group, Number of Graph Isomorphisms, Double Coset Membership, Croup Factorization, Group Intersection, Setwise Stabilizer.
- There are several interesting zeta functions on graphs (Riemann, Ruelle, Ihara, Selberg and others). What is the complexity of the isomorphism tests based on these functions?

Questions

- Is there a "trivial test"for the Maximal Clique problem (to which the GI reduces polynomially).
- What is the generic case complexity of Hidden Subgroup Problem?
- It is not true that the generic case complexity is invariant under polynomial equivalence. Thus it makes sense to ask whether the following problems admit generically polynomial algorithms: Search Isomorphism, Graph Automorphism, Order of the Automorphism Group, Number of Graph Isomorphisms, Double Coset Membership, Croup Factorization, Group Intersection, Setwise Stabilizer.
- There are several interesting zeta functions on graphs (Riemann, Ruelle, Ihara, Selberg and others). What is the complexity of the isomorphism tests based on these functions?

Conclusion

- Есть табак, да нечем нюхать!

Conclusion

- Есть табак, да нечем нюхать!
  - Respect GI!
    - Ill GI!

Conclusion

- Есть табак, да нечем нюхать!
  - Respect GI!
    - Ill GI!

Conclusion

- Есть табак, да нечем нюхать!
  - Respect GI!
    - Ill GI!
    - Infect GI!

Conclusion

- Есть табак, да нечем нюхать!
  - Respect GI!
    - Ill GI!
  - Infect GI!
- Happy 60th Birthday, Alexei!

Conclusion

- Есть табак, да нечем нюхать!
  - Respect GI!
    - Ill GI!
  - Infect GI!
- Happy 60th Birthday, Alexei!