Linear time recognition of circle graphs

Christophe Paul

CNRS - University of Montpellier, France

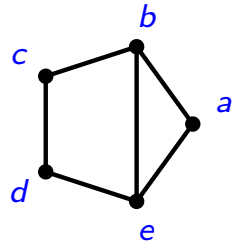
STACS 2026, Grenoble – March 10-13, 2026.

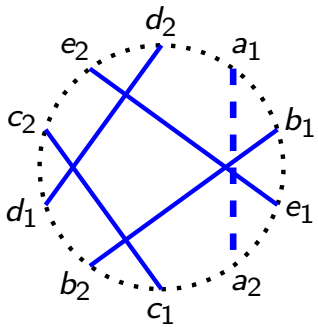
Joint work with: **Ignaz Rutter** (University of Passau, Germany)

Based on a previous paper with: Derek Corneil (University of Toronto), Emeric Gioan (CNRS, University of Montpellier), Marc Tedder (University of Toronto)

Circle graphs

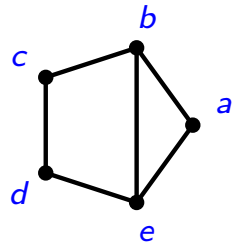
A circle graph is the intersection graph of a set of chords in a circle.

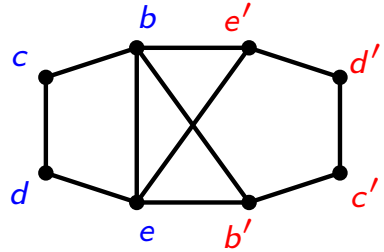


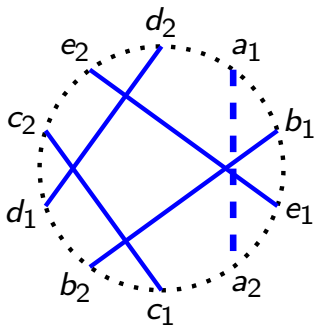


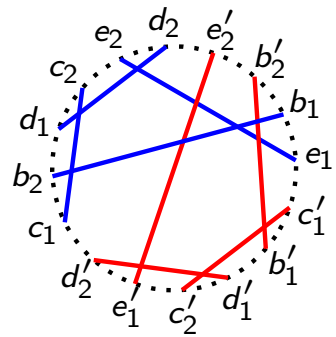
Circle graphs

A circle graph is the intersection graph of a set of chords in a circle.





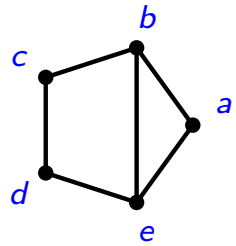


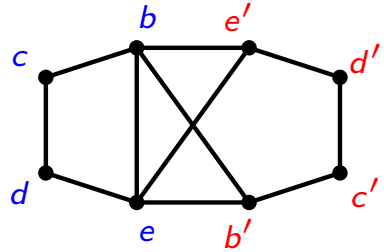


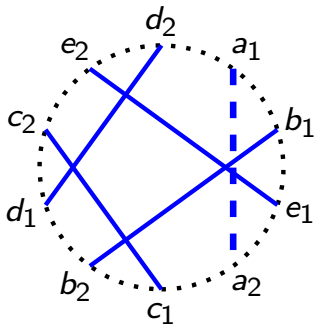
A split in a graph is a (non-trivial) bipartition (X, Y) of the vertices such that $x \in X$ is adjacent to $y \in Y$ iff $x \in N(Y)$ and $y \in N(X)$.

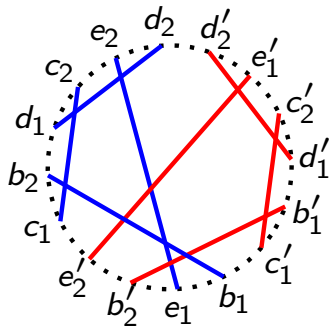
Circle graphs

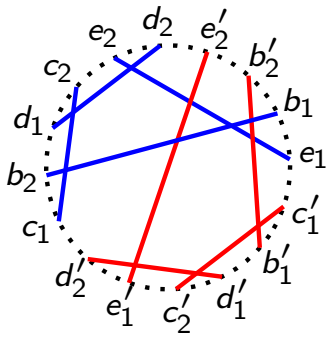
A circle graph is the intersection graph of a set of chords in a circle.







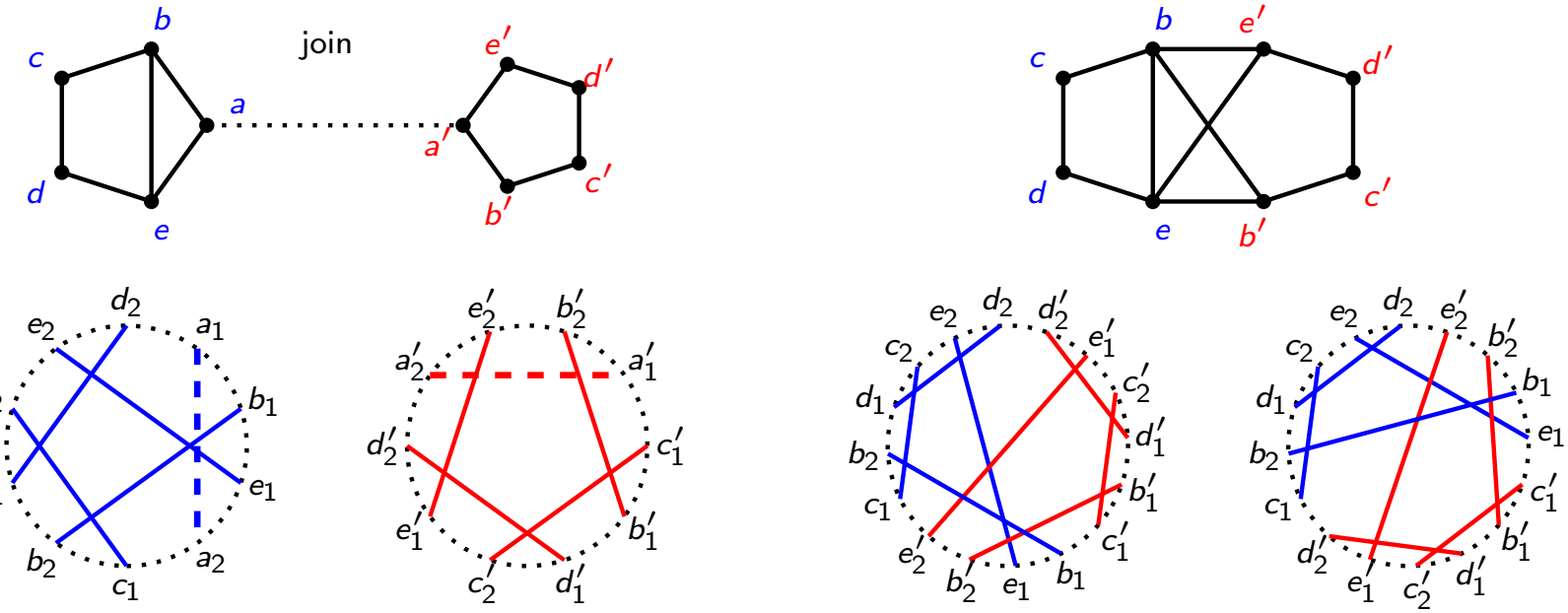




A split in a graph is a (non-trivial) bipartition (X, Y) of the vertices such that $x \in X$ is adjacent to $y \in Y$ iff $x \in N(Y)$ and $y \in N(X)$.

Circle graphs

A circle graph is the intersection graph of a set of chords in a circle.

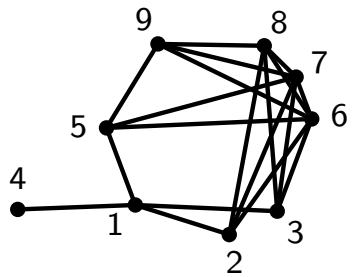


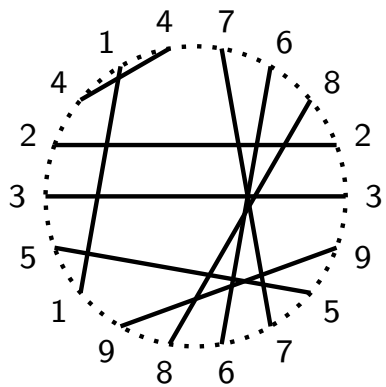
A split in a graph is a (non-trivial) bipartition (X, Y) of the vertices such that $x \in X$ is adjacent to $y \in Y$ iff $x \in N(Y)$ and $y \in N(X)$.

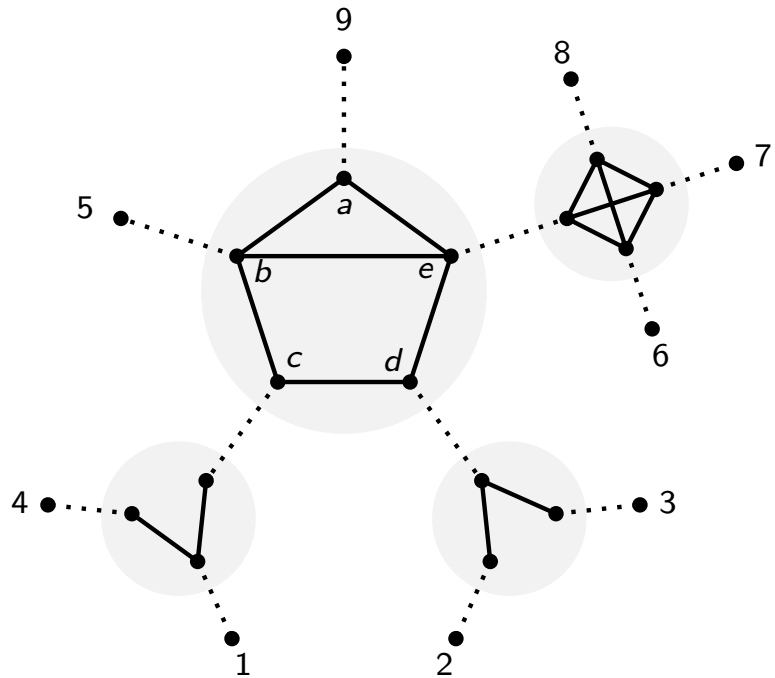
Circle graphs and split decomposition

Theorem [Cunningham, Edmonds, 1980]

Every graph uniquely split-decomposes into primes, cliques and stars.





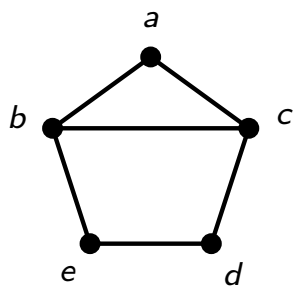


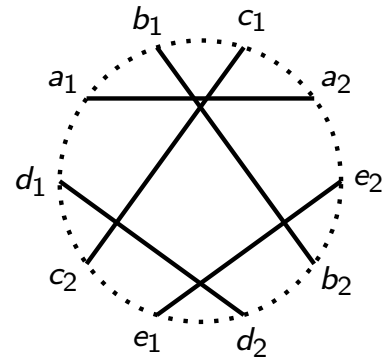
Theorem [Naji 1982, Bouchet 1985]

A graph is a circle graph iff its prime components are circle graphs.

Prime circle graphs

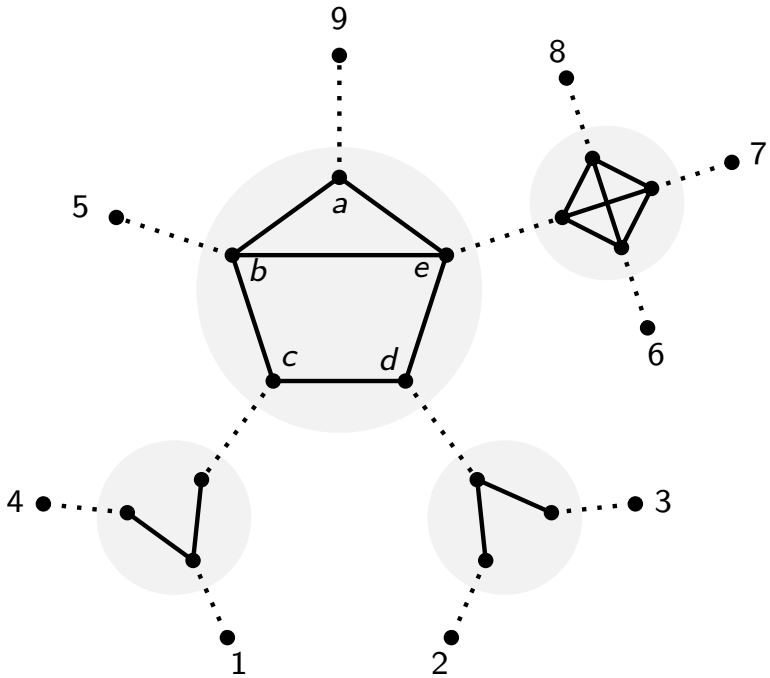
Theorem [Bouchet 1985] A circle graph is prime if and only if it has a unique (up to reversal) chord diagram.

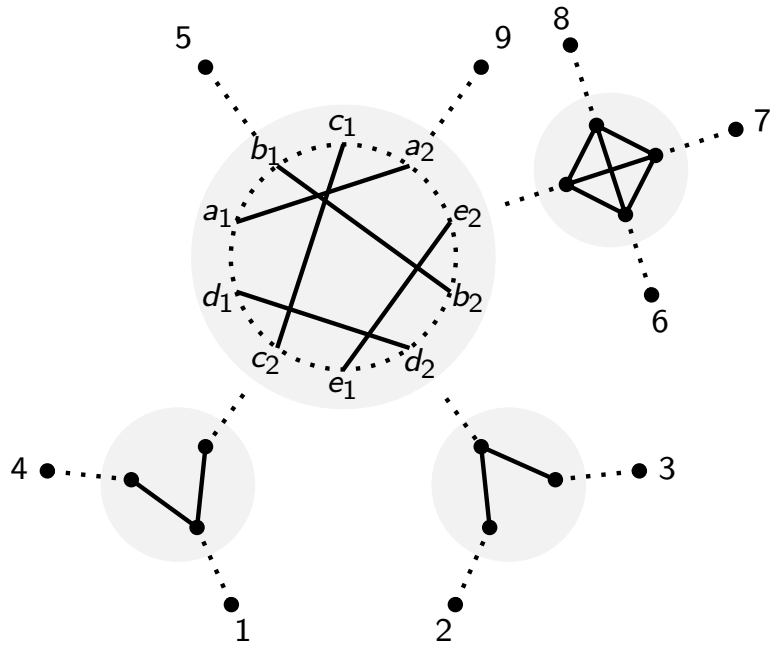




Prime circle graphs

Theorem [Bouchet 1985] A circle graph is prime if and only if it has a unique (up to reversal) chord diagram.





$\rightsquigarrow$ A circle graph G, and its set of chord diagrams, can be compactly represented by the split diagram tree, denoted **SDT**(G).

Existing algorithms

- $O(n^7)$ [Naji, 1985] $\rightarrow$ system of linear equations
- $O(n^5)$ [Bouchet, 1987] $\rightarrow$ detection vertex minor obstructions

Existing algorithms

- $O(n^7)$ [Naji, 1985] $\rightarrow$ system of linear equations
- $O(n^5)$ [Bouchet, 1987] $\rightarrow$ detection vertex minor obstructions
- $O(n^3)$ [Gabor, Hsu, Suppovit, 1989] $\rightarrow$ split decomposition
- $O(n^2)$ [Spinrad, 1992] $\rightarrow$ incremental split decomposition

Existing algorithms

- $O(n^7)$ [Naji, 1985] $\rightarrow$ system of linear equations
- $O(n^5)$ [Bouchet, 1987] $\rightarrow$ detection vertex minor obstructions
- $O(n^3)$ [Gabor, Hsu, Suppovit, 1989] $\rightarrow$ split decomposition
- $O(n^2)$ [Spinrad, 1992] $\rightarrow$ incremental split decomposition
- $O((n+m) \cdot \alpha(n,m))$ [Corneil, Gioan, Paul, Tedder, 2014]
  $\rightarrow$ LexBFS incremental split decomposition

Existing algorithms

- $O(n^7)$ [Naji, 1985] $\rightarrow$ system of linear equations
- $O(n^5)$ [Bouchet, 1987] $\rightarrow$ detection vertex minor obstructions
- $O(n^3)$ [Gabor, Hsu, Suppovit, 1989] $\rightarrow$ split decomposition
- $O(n^2)$ [Spinrad, 1992] $\rightarrow$ incremental split decomposition
- $O((n+m) \cdot \alpha(n,m))$ [Corneil, Gioan, Paul, Tedder, 2014]

  $\rightarrow$ LexBFS incremental split decomposition

  $\rightarrow$ Based on union-find data-structure [Tarjan 1975]

Existing algorithms

- $O(n^7)$ [Naji, 1985] $\rightarrow$ system of linear equations
- $O(n^5)$ [Bouchet, 1987] $\rightarrow$ detection vertex minor obstructions
- $O(n^3)$ [Gabor, Hsu, Suppovit, 1989] $\rightarrow$ split decomposition
- $O(n^2)$ [Spinrad, 1992] $\rightarrow$ incremental split decomposition
- $O((n+m) \cdot \alpha(n,m))$ [Corneil, Gioan, Paul, Tedder, 2014]

  $\rightarrow$ LexBFS incremental split decomposition

  $\rightarrow$ Based on union-find data-structure [Tarjan 1975]

Theorem [P., Rutter] Circle graphs can be recognized in linear time.

$\rightsquigarrow$ by using PC-trees instead of union-find data-structure !

Corneil, Gioan, P. & Tedder's algorithm

1. Compute a LexBFS ordering of the input graph G.
2. Maintain the split diagram tree of G by incrementally adding the vertices of G according to the LexBFS ordering.

Corneil, Gioan, P. & Tedder's algorithm

1. Compute a LexBFS ordering of the input graph G.
2. Maintain the split diagram tree of G by incrementally adding the vertices of G according to the LexBFS ordering.

$\rightsquigarrow$ The recognition problem reduces to the following problem:

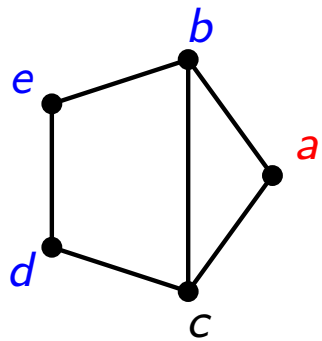
Let $G' = G + x$ be a graph such that

- G is a circle graph,
- x is the last vertex of a LexBFS of G' and
- we are given the split diagram tree $\mathbf{SDT}(G)$ of G.

In amortized $O(d(x))$-time, compute

- $\mathbf{SDT}(G + x)$ if $G + x$ is a circle graph, or
- return that $G + x$ is not circle.

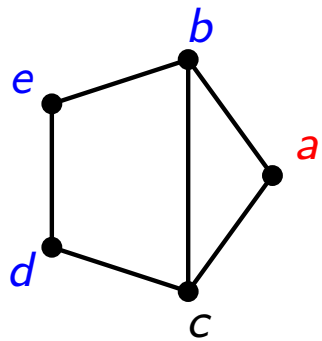
LexBFS lemma (1)

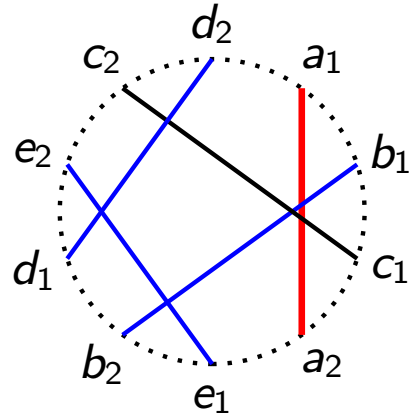


$\rightsquigarrow \langle e, d, b, c, a \rangle$ is a LexBFS ordering of the house graph

LexBFS lemma (1)

Lemma [Corneil, Gioan, P., Tedder, 2014]
If x is the last vertex of a LexBFS ordering of a circle graph G, then G has a chord diagram C in which x is extreme
(the neighbourhood of x is consecutive in C).

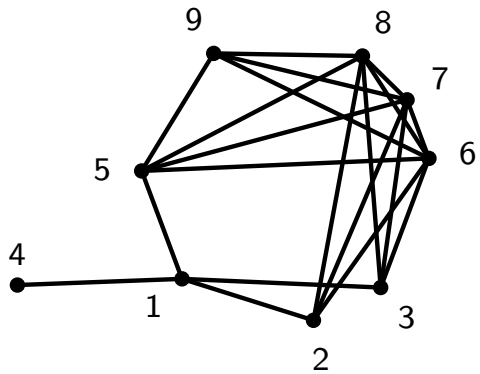


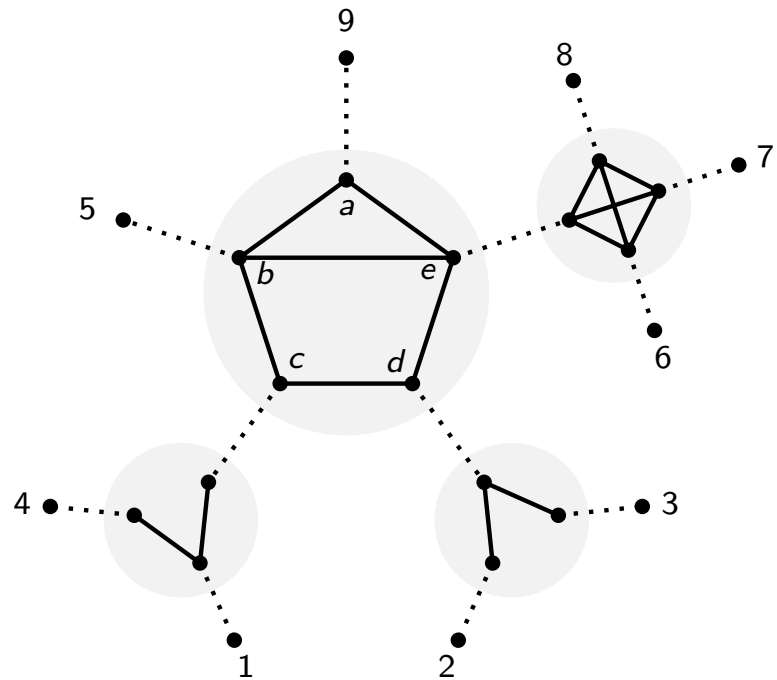


$\rightsquigarrow \langle e, d, b, c, a \rangle$ is a LexBFS ordering of the house graph

LexBFS lemma (2)

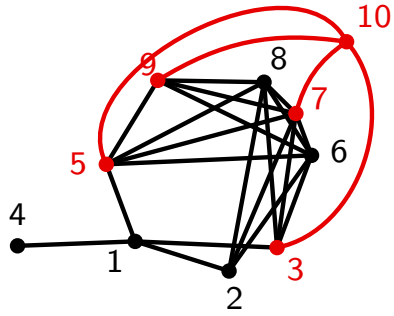
Lemma [Corneil, Gioan, P., Tedder, 2014]
If σ is a LexBFS ordering of G and C is a split component of G, then σ_C is a LexBFS ordering of C.

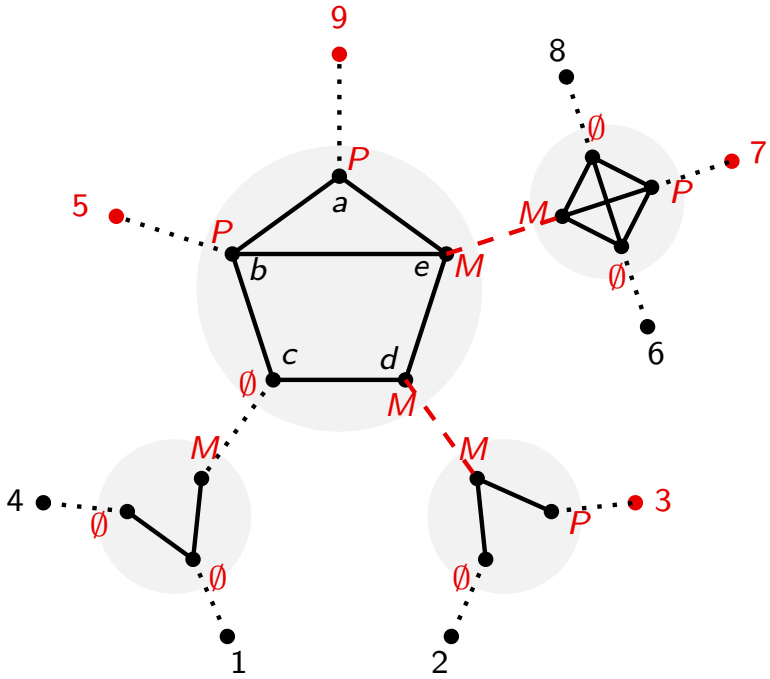




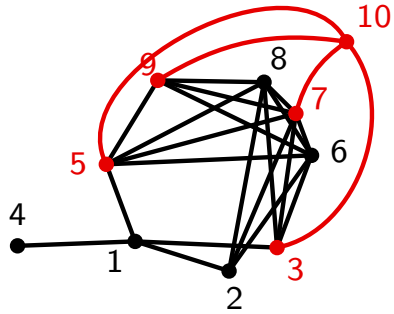
$\rightsquigarrow \langle 4, 1, 2, 5, 3, 6, 7, 8, 9 \rangle$ is a LexBFS ordering of the graph on the left
$\langle e, d, b, c, a \rangle$ is a LexBFS ordering of the house graph.

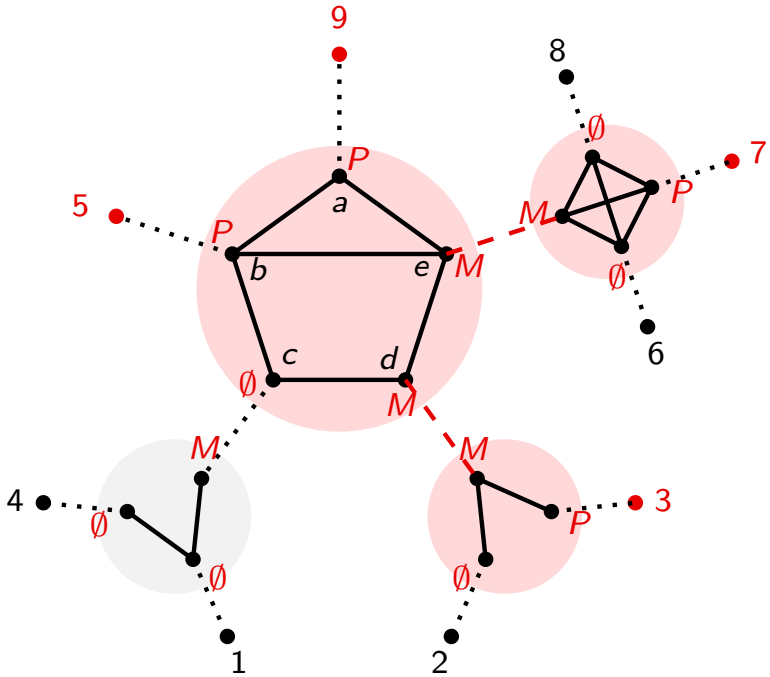
Updating the split-tree (1)



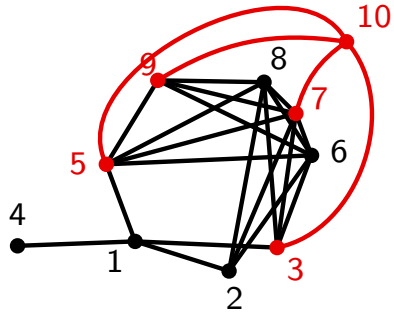


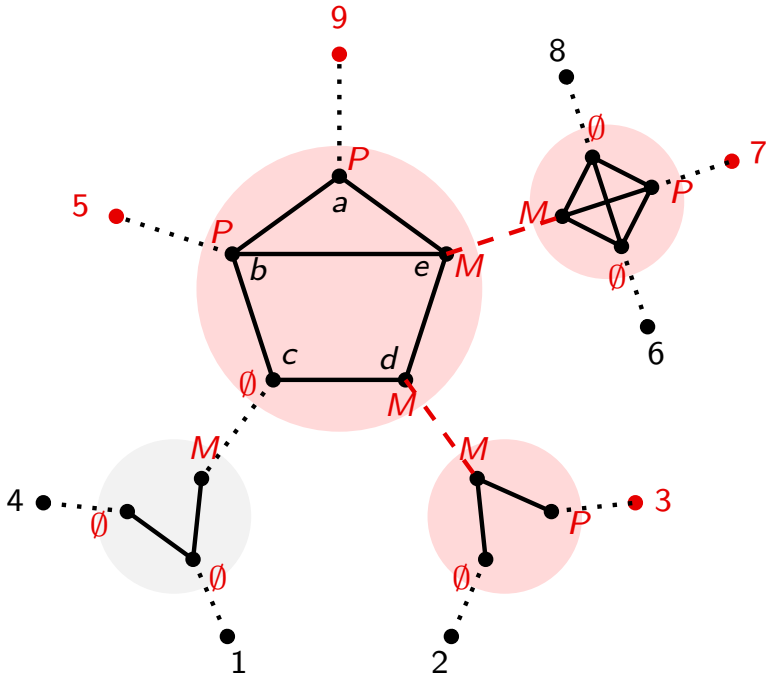
Updating the split-tree (1)

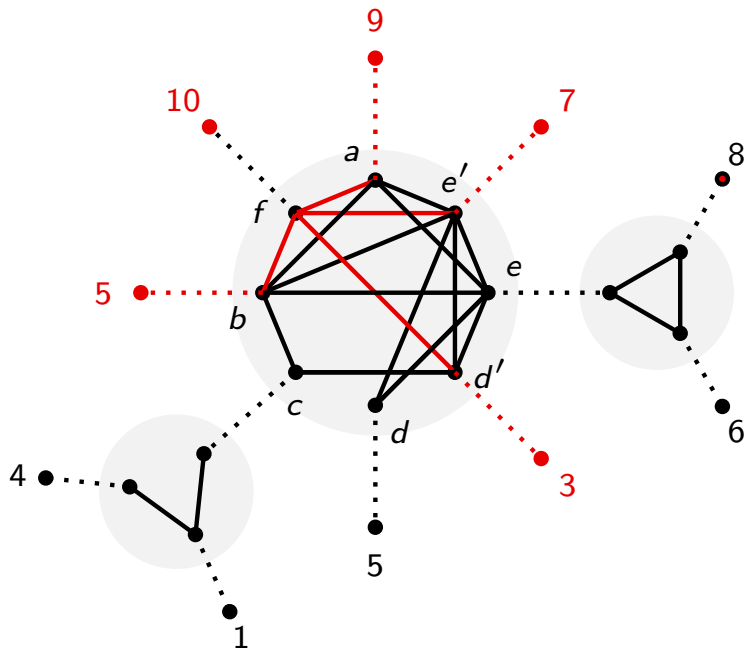




Updating the split-tree (1)







Updating the split-tree (2)

1. **PC**(G) *contains a clique node u whose marker vertices are all perfect; or T contains a star node u whose marker vertices are all empty except its center, which is perfect; or T contains a unique hybrid node u, which is prime.*

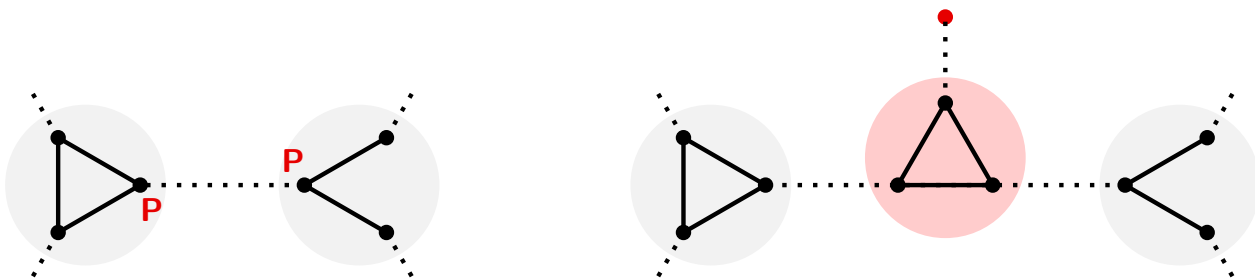
2. **PC**(G) *contains a tree-edge e whose extremities are both perfect, and this edge is unique; or T contains a tree-edge with one perfect and one empty extremity, and this edge is unique.*

3. **PC**(G) *contains a hybrid node u, and this node is degenerate.*

Updating the split-tree (2)

1. **PC**(G) *contains a clique node* u *whose marker vertices are all perfect; or* T *contains a star node* u *whose marker vertices are all empty except its center, which is perfect; or* T *contains a unique hybrid node* u*, which is prime.*

2. **PC**(G) *contains a tree-edge* e *whose extremities are both perfect, and this edge is unique; or* T *contains a tree-edge with one perfect and one empty extremity, and this edge is unique.*



3. **PC**(G) *contains a hybrid node* u*, and this node is degenerate.*

Updating the split-tree (2)

1. **PC**(G) *contains a clique node u whose marker vertices are all perfect; or T contains a star node u whose marker vertices are all empty except its center, which is perfect; or T contains a unique hybrid node u, which is prime.*

2. **PC**(G) *contains a tree-edge e whose extremities are both perfect, and this edge is unique; or T contains a tree-edge with one perfect and one empty extremity, and this edge is unique.*

3. **PC**(G) *contains a hybrid node u, and this node is degenerate.*

4. **PC**(G) *contains a (unique) fully mixed subtree M.*

   4.1 cleaning step
   4.2 contraction / join step
   4.3 insertion step

Updating the split-tree (2)

1. **PC**(G) *contains a clique node u whose marker vertices are all perfect; or T contains a star node u whose marker vertices are all empty except its center, which is perfect; or T contains a unique hybrid node u, which is prime.*

   $\rightsquigarrow$ $O(1)$-time / $O(d(x))$-time

2. **PC**(G) *contains a tree-edge e whose extremities are both perfect, and this edge is unique; or T contains a tree-edge with one perfect and one empty extremity, and this edge is unique.*

   $\rightsquigarrow$ $O(1)$-time

3. **PC**(G) *contains a hybrid node u, and this node is degenerate.*

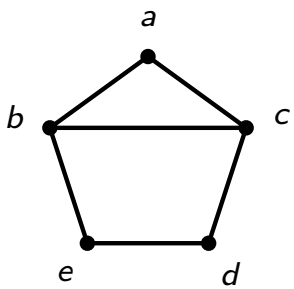
   $\rightsquigarrow$ $O(1)$-time

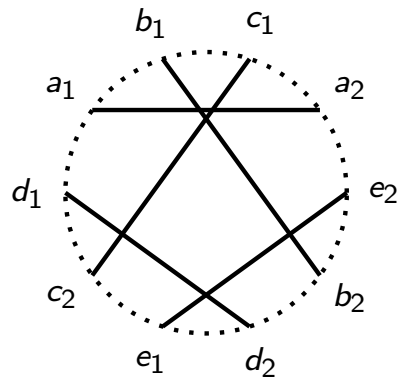
4. **PC**(G) *contains a (unique) fully mixed subtree M.*

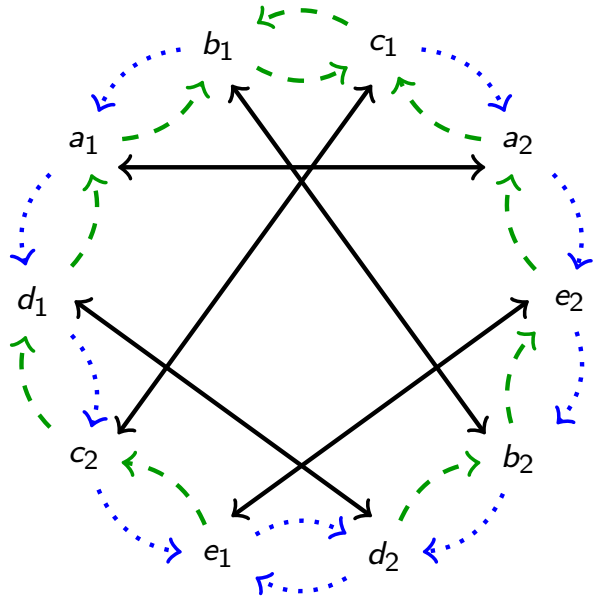
   $\rightsquigarrow$ amortized $O(d(x))$-time

   4.1 cleaning step
   4.2 contraction / join step
   4.3 insertion step

The data structure (1) – Consistent Symmetric Cycles



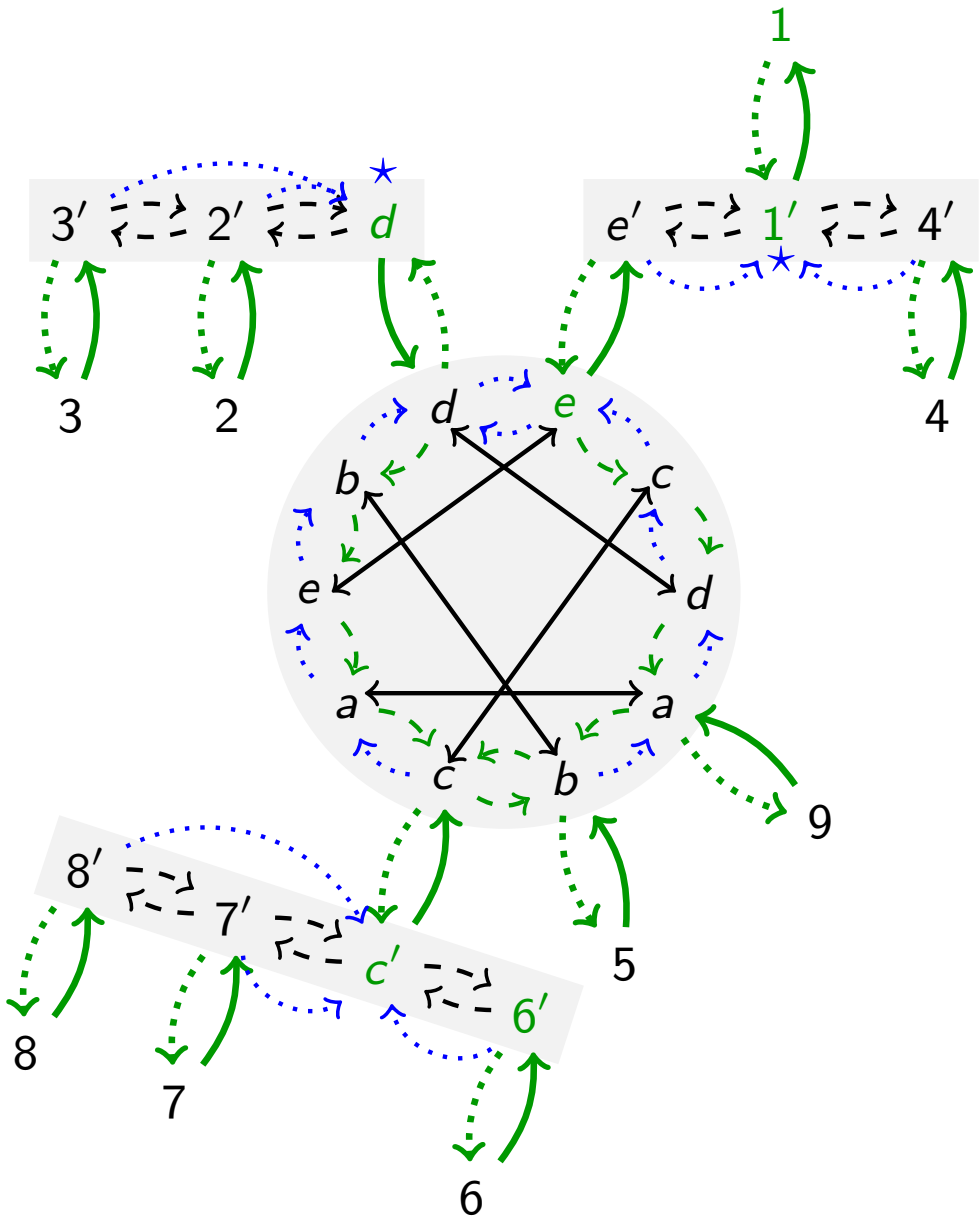




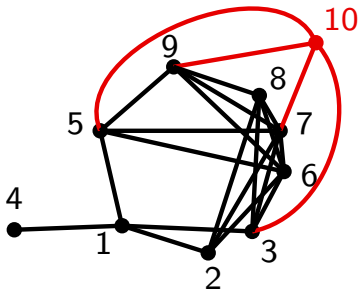
Lemma [Corneil, Gioan, P., Tedder, 2014] CSC's allow to perform

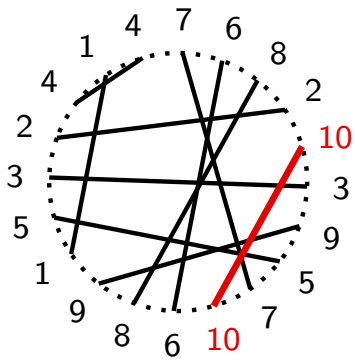
1. join operation in $O(1)$-time,
2. chord insertion in $O(1)$-time,
3. consecutivity test of a set S of chords in $O(|S|)$-time.

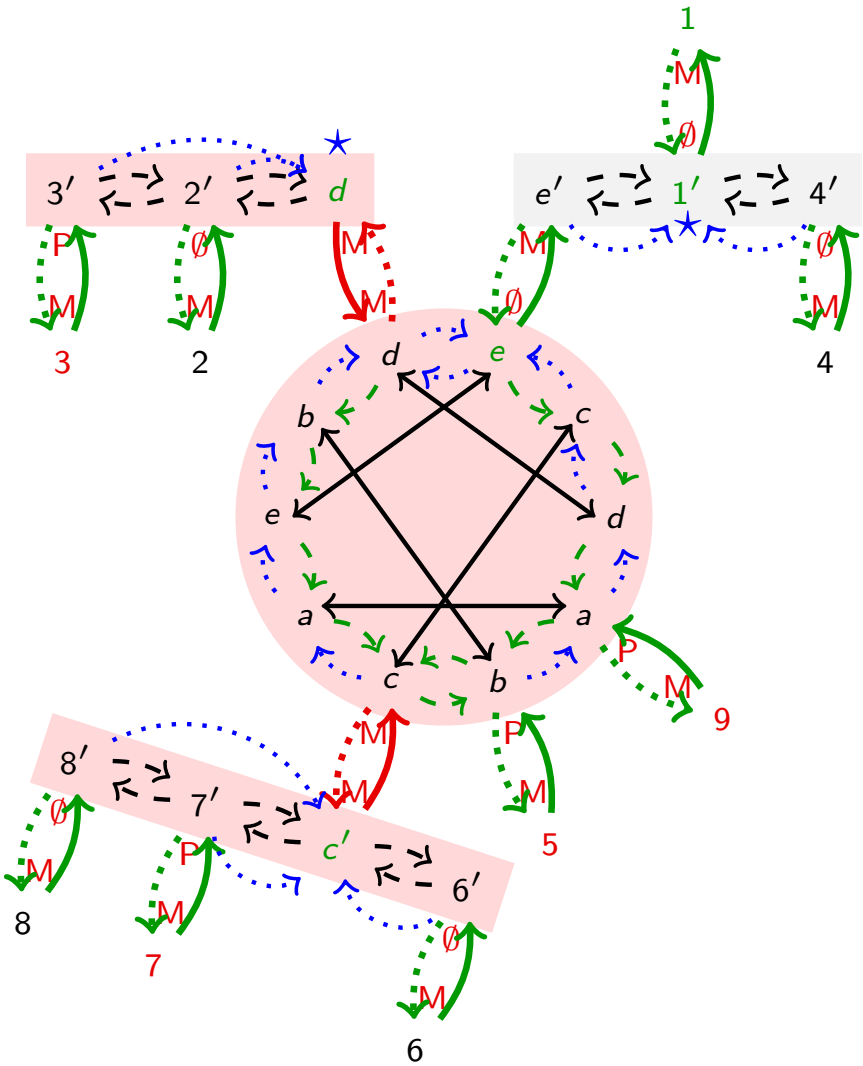
The data structure (2) – The split PC-tree



Updating the split PC-tree (1)

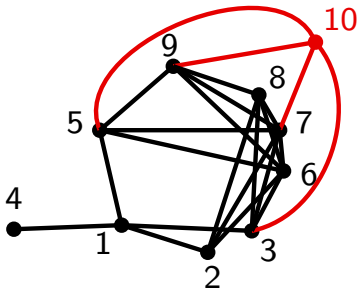


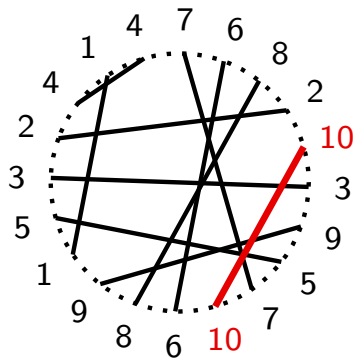


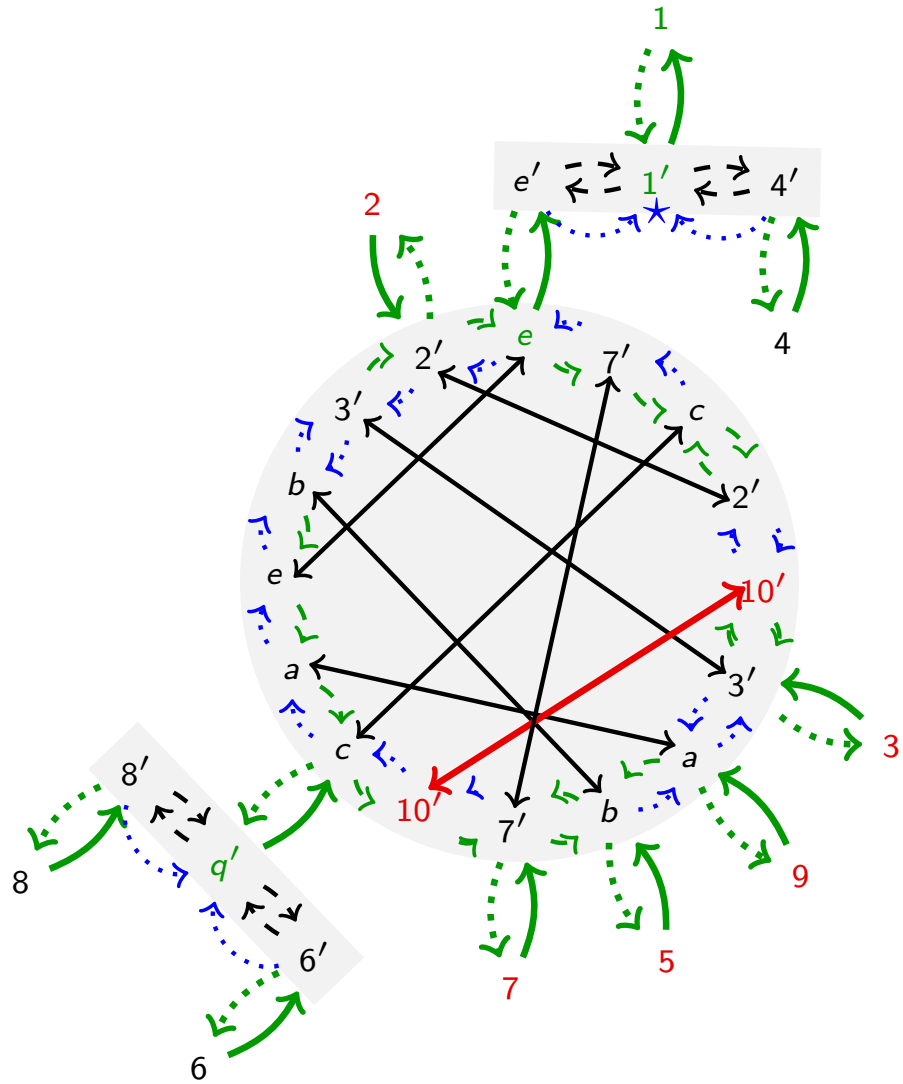


1. Compute $T(N(10))$ and cleaning it
2. contract $T(N(10))$ by performing joins

Updating the split PC-tree (2)







2. Contract $T(N(10))$ by performing joins
3. Test whether $N(10)$ is consecutive and if so insert a new chord.

Some observations and intuition

Observation

In a prime node the perfect marker vertices forms an interval and the mixed are the bookends of that interval

$\rightsquigarrow$ As a consequence, the fully-mixed subtree is a caterpillar

Some observations and intuition

Observation

In a prime node the perfect marker vertices forms an interval and the mixed are the bookends of that interval

$\rightsquigarrow$ As a consequence, the fully-mixed subtree is a caterpillar

Lemma [Corneil, Gioan, P., Tedder, 2014] Let $\mathbf{ST}(G) = (T, \mathcal{F})$ be the split-tree of a graph G. Then

$$\forall \text{ vertex } x \text{ of } G,\ |T(N_G(x))| \leq 2 \cdot |N_G(x)|.$$

Amortized time complexity analysis

Let $G' = G + x$ be a graph such that

- x is a good vertex of G',
- G is a circle graph and the split-tree $\mathbf{ST}(G) = (T, \mathcal{F})$ is represented by the split PC-tree $\mathbf{PC}(G)$

Lemma [P., Rutter, 2025]
In time $O(|T(N_{G'}(x))|)$, we can either compute $\mathbf{PC}(G')$ or conclude that G' is not a circle graph.

1. First, compute $T(N_{G'}(x))$ or conclude that G' is not a circle graph.
2. Then, contract $T(N_{G'}(x))$ and insert x to obtain $\mathbf{PC}(G')$ or conclude that G' is not a circle graph

Amortized time complexity analysis

For an inner node u of $\mathbf{ST}(G) = (T, \mathcal{F})$, we define the potential function

$$\Phi_u = \begin{cases} 1 & \text{if } u \text{ is non-degenerate,} \\ \deg_T(u) - 2 & \text{if } u \text{ is degenerate,} \end{cases}$$

and we set $\Phi(\mathbf{ST}(G)) = \sum_{u \in \mathbf{ST}(G)} \Phi_u$.

Amortized time complexity analysis

For an inner node u of $\mathbf{ST}(G) = (T, \mathcal{F})$, we define the potential function

$$\Phi_u = \begin{cases} 1 & \text{if } u \text{ is non-degenerate,} \\ \deg_T(u) - 2 & \text{if } u \text{ is degenerate,} \end{cases}$$

and we set $\Phi(\mathbf{ST}(G)) = \sum_{u \in \mathbf{ST}(G)} \Phi_u$.

The amortized cost to insert vertex x is

$$a(x) = O(|T(N_{G+x}(x))|) + \Phi(\mathbf{ST}((G + x)) - \Phi(\mathbf{ST}((G)).$$

Amortized time complexity analysis

For an inner node u of $\mathbf{ST}(G) = (T, \mathcal{F})$, we define the potential function

$$\Phi_u = \begin{cases} 1 & \text{if } u \text{ is non-degenerate,} \\ \deg_T(u) - 2 & \text{if } u \text{ is degenerate,} \end{cases}$$

and we set $\Phi(\mathbf{ST}(G)) = \sum_{u \in \mathbf{ST}(G)} \Phi_u$.

The amortized cost to insert vertex x is

$$a(x) = O(|T(N_{G+x}(x))|) + \Phi(\mathbf{ST}((G + x)) - \Phi(\mathbf{ST}((G)).$$

Lemma [P., Rutter, 2025]
Let $G' = G + x$ be a graph such that x is a good vertex of G and G is a circle graph. If $G' = G + x$ is a circle graph with x a good vertex, then

$$a(x) = O(|N_{G'}(x)|).$$

Amortized time complexity analysis

Theorem [P., Rutter, 2025]
The running time of the Corneil, Gioan, P., Tedder's circle graph recognition algorithm with the split PC-tree data-structure is linear.

Open problem
Could the $\alpha(n, m)$ factor be also removed from the general split decomposition algorithm of [Corneil, Gioan, P., Tedder, Algorithmica 2014] ?

Thank you !