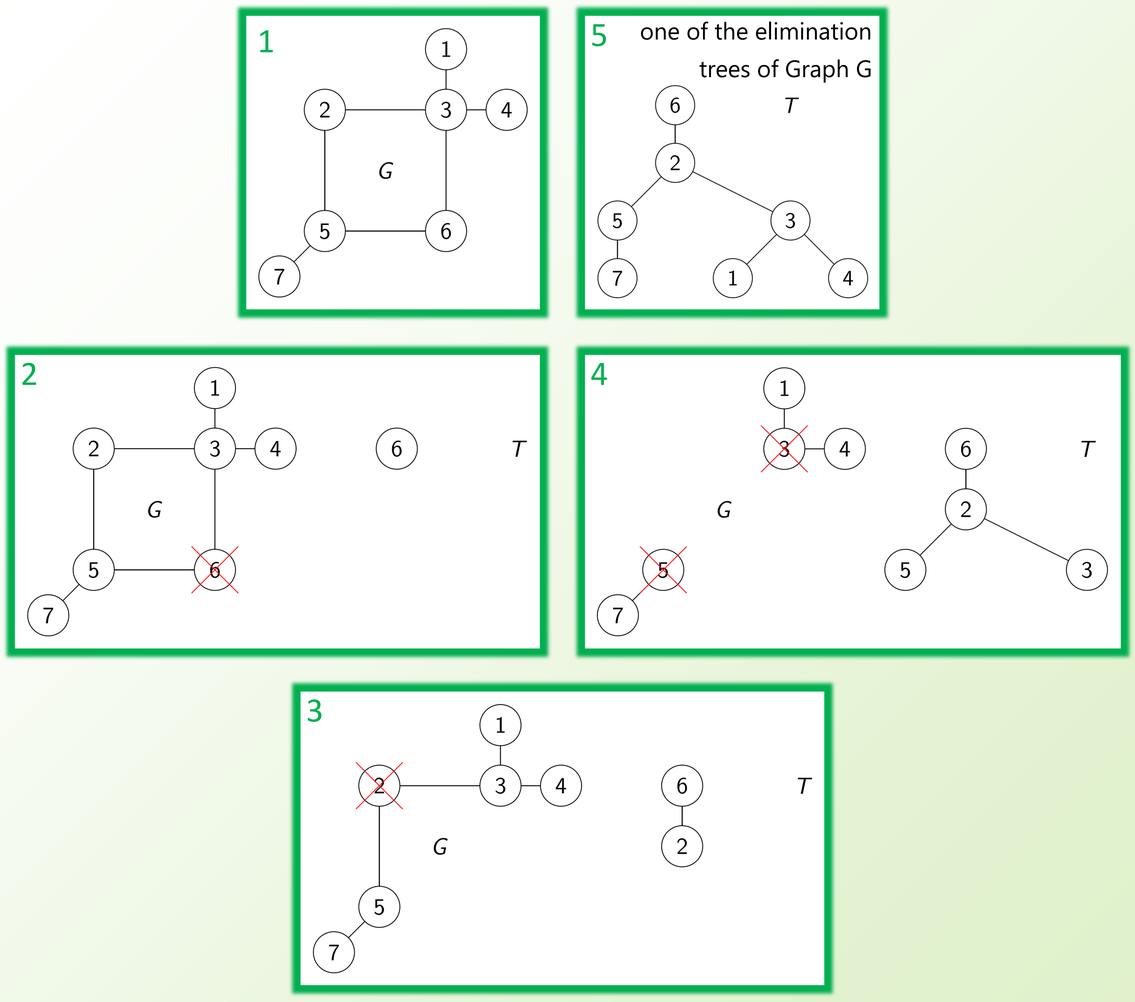
Efficient generation of elimination trees

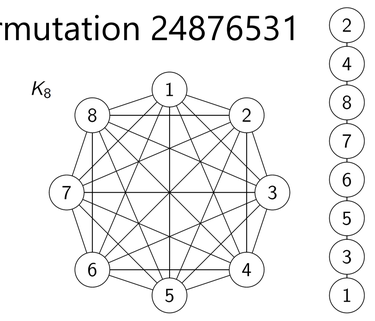
Definition: Elimination tree (recursively)

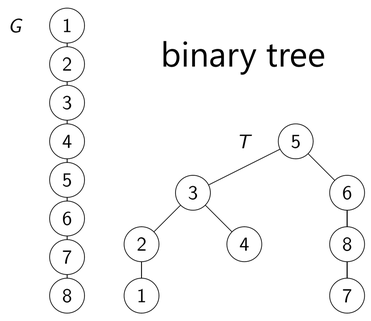
- G = (V, E) connected graph
- elimination tree T for G is a rooted tree with vertex set V
- obtained by choosing a root x ∈ V in T and recursing on the connected components of G – x to produce the subtrees of x



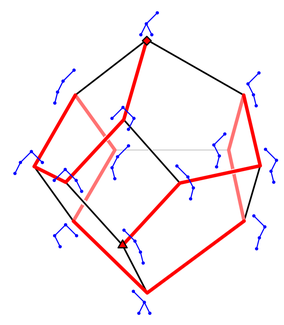
Motivation

- generating all elimination trees for a graph generalizes algorithms for generating permutations, binary trees, ...



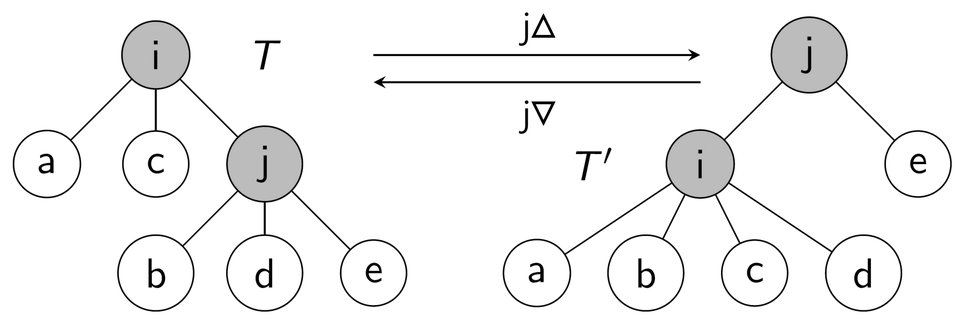


- We look at the flip graph for any given graph G under tree rotations.
- Algorithm, which generates all elimination trees for a graph G using rotations, finds a hamilton path on the flip graph of G.



Rotation jΔ

- Make j parent of i (and i child of j).
- A subtree S of j in T remains a subtree of j, unless the vertices of S belong to the same connected component of G – j as i, in which case S becomes a subtree of i.

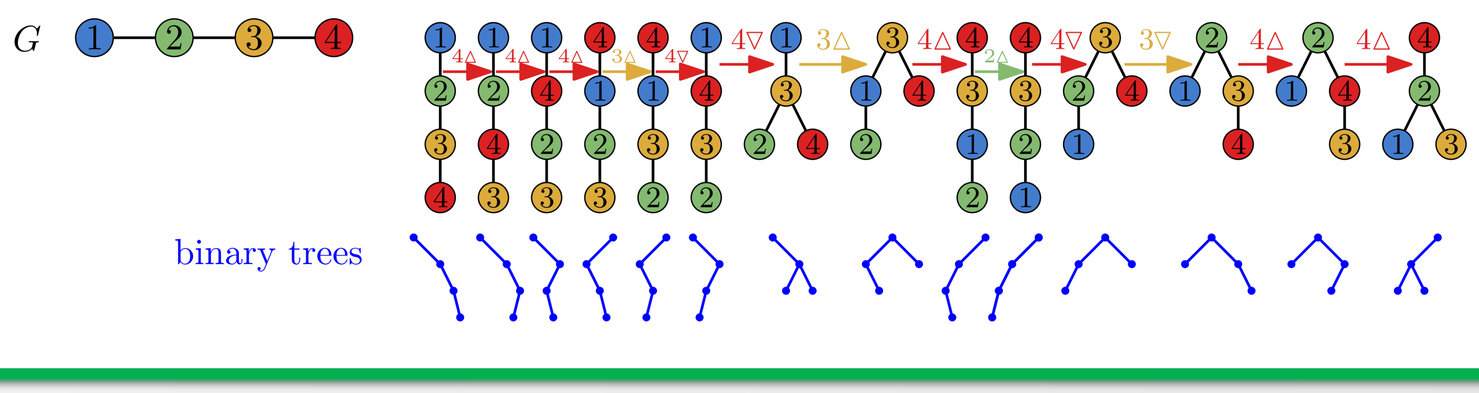


T and T′ are elimination trees from graph G in definition

The algorithm

generates all elimination trees for a chordal graph G = ([n], E)

1. Visit one initial elimination tree T_0.
2. Generate an unvisited elimination tree from G by performing an rotation xΔ or x∇ for the largest possible vertex x in the most recently visited elimination tree. If no such rotation exists, or the rotation is ambiguous, then terminate. Otherwise, visit this elimination forest and repeat 2.

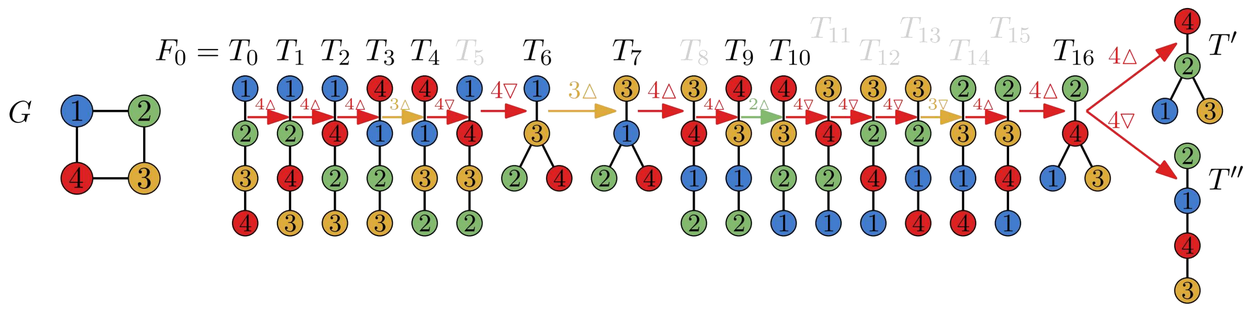


Efficient generation

- The algorithm requires storing all previously visited elimination trees, in order to decide upon the next rotation, which should generate an unvisited elimination tree.
- In the paper is shown, that we can get rid of this defect and make the algorithm memoryless.
- For a chordal graph G = ([n], E) with m = |E| edges, the efficient algorithm visits each elimination tree for G in time $\mathcal{O}(m+n)$.
- For trees, this can be improved to $\mathcal{O}(1)$.

When does the algorithm work?

- graph G is chordal ⇔ the algorithm finds all elimination trees
- If G is not chordal, the algorithm terminates because of an ambiguous rotation or because there is no possible rotation.



The algorithm terminates, because the rotation on T_{16} is ambiguous.

References

- Jean Cardinal, Arturo Merino, and Torsten Mütze. "Efficient generation of elimination trees and graph associahedra". In: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). SIAM. 2022, pp. 2128–2140.
- poster by Julian Sampels