Utrecht University

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

25 April, 2025

Tutte's theorem as an educational formalization project

Pim Otte
PhD Candidate

Outline

Introduction

Motivation

Tutte’s theorem

Educational formalization project

Conclusion

Outline

About me

- Pim Otte
- PhD candidate at Utrecht University & Technical University of Eindhoven
- Topic “Type Theory for Education”
- Webpage: `https://pim.otte.dev` (Preprint: [Ott25])

Topic

- Formalization of Tutte’s theorem in Lean 4, almost merged to mathlib
- Template for educational formalization projects

Outline

Motivation

- Why Tutte’s theorem?

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Template for educational formalization projects

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Template for educational formalization projects
  - Asymmetry in number of teachers vs students

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Template for educational formalization projects
  - Asymmetry in number of teachers vs students
  - Bridging the gap between tutorials and "blueprint projects"

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Template for educational formalization projects
  - Asymmetry in number of teachers vs students
  - Bridging the gap between tutorials and "blueprint projects"
- Shout-outs:
  - Mathlib reviewers; Yaël Dillies, Bhavik Mehta, Kyle Miller

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Template for educational formalization projects
  - Asymmetry in number of teachers vs students
  - Bridging the gap between tutorials and "blueprint projects"
- Shout-outs:
  - Mathlib reviewers; Yaël Dillies, Bhavik Mehta, Kyle Miller
  - Related work: Formalization of graph algorithms in Isabelle/HOL by Abdulaziz [Abd24].

Outline

Introduction

Motivation

Tutte’s theorem

Educational formalization project

Conclusion

SimpleGraph/Subgraph

```
structure SimpleGraph (V : Type u) where
  Adj : V → V → Prop
  symm : Symmetric Adj := by aesop_graph
  loopless : Irreflexive Adj := by aesop_graph

structure Subgraph {V : Type u} (G : SimpleGraph V) where
  verts : Set V
  Adj : V → V → Prop
  adj_sub : ∀ {v w : V}, Adj v w -> G.Adj v w
  edge_vert : ∀ {v w : V}, Adj v w -> v ∈ verts
  symm : Symmetric Adj := by aesop_graph
```

Perfect matching

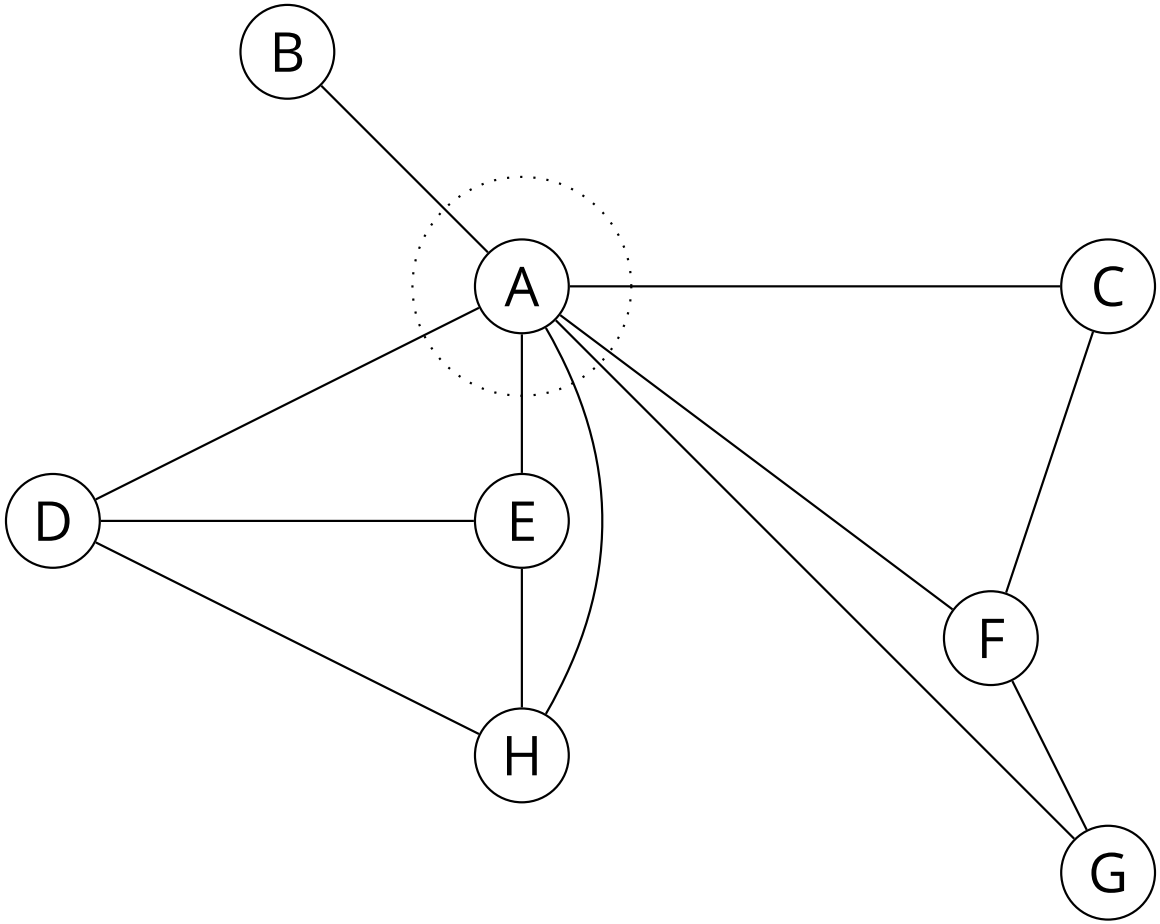
```
def IsMatching (M : Subgraph G) : Prop :=
  ∀ {|v|}, v ∈ M.verts → ∃! w, M.Adj v w

def IsSpanning (G' : Subgraph G) : Prop := ∀ v : V, v ∈
    G'.verts

def IsPerfectMatching (M : G.Subgraph) : Prop :=
  M.IsMatching ∧ M.IsSpanning
```

Tutte’s theorem

Theorem (Tutte, 1947).

A graph G has a perfect matching if and only if for any subset $U \subseteq V$ the graph $G - U$ has at most $|U|$ components of odd size.

Tutte Violator



Tutte violator

Definition (Tutte violator).

A subset $|U|$ such that $G - U$ has more than $|U|$ components of odd size.

```
def IsTutteViolator (G: SimpleGraph V) (u : Set V) : Prop :=
  u.ncard < ((⊤ : G.Subgraph).deleteVerts
    u).coe.oddComponents.ncard
```

Formal proof of Tutte's theorem

```
theorem tutte [Fintype V] :
  (∃ (M : Subgraph G) , M.IsPerfectMatching) ↔
  (∀ (u : Set V), ¬ G.IsTutteViolator u) := by
classical
refine ⟨by rintro ⟨M, hM⟩; apply not_IsTutteViolator hM, ?_⟩
contrapose!
intro h
by_cases hvOdd : Odd (Fintype.card V)
· exact ⟨∅, isTutteViolator_empty hvOdd⟩
· exact exists_TutteViolator h (Nat.not_odd_iff_even.mp hvOdd)
```

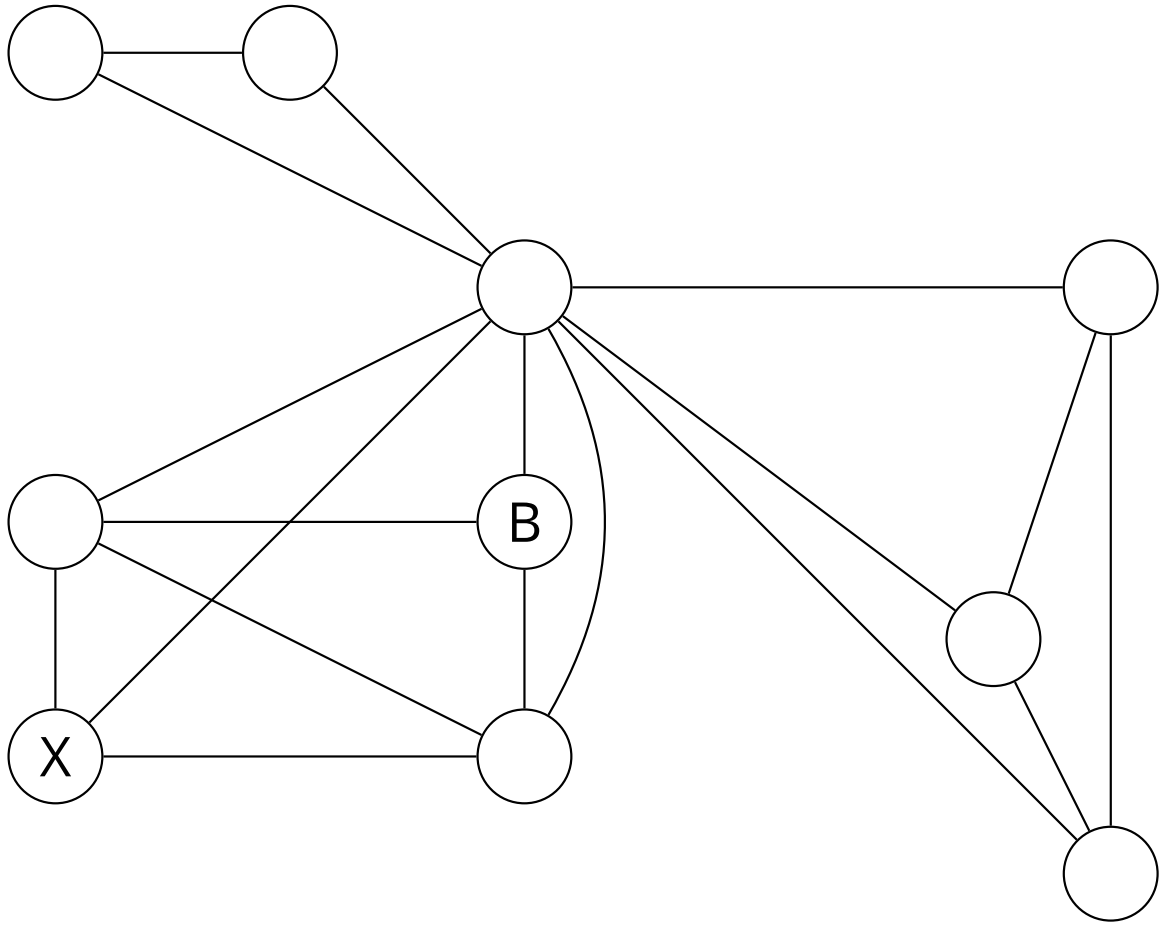
Edge-maximal counterexample

- A Tutte violator in a graph also works in a smaller graph

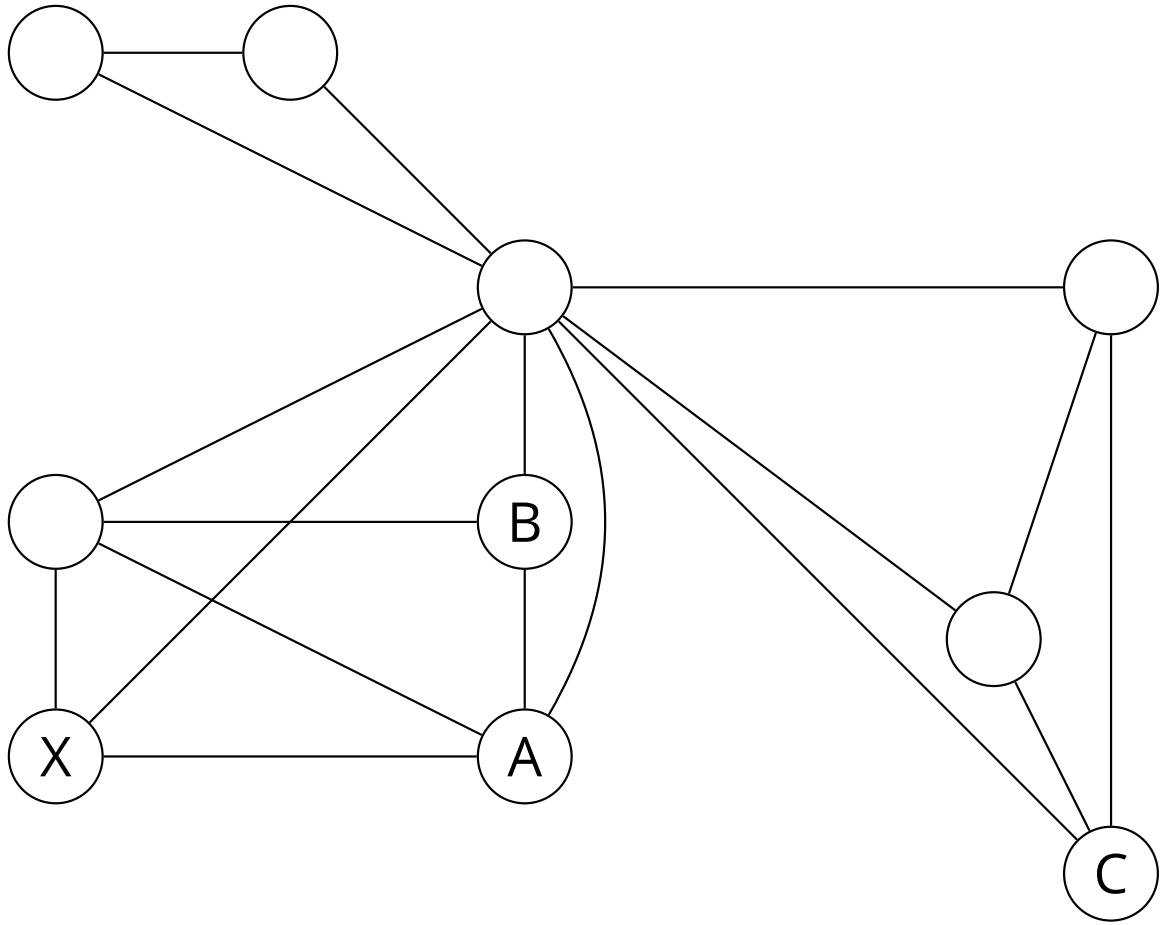
Edge-maximal counterexample

- A Tutte violator in a graph also works in a smaller graph
- Thus, we can work with a graph in which adding any edge yields a perfect matching

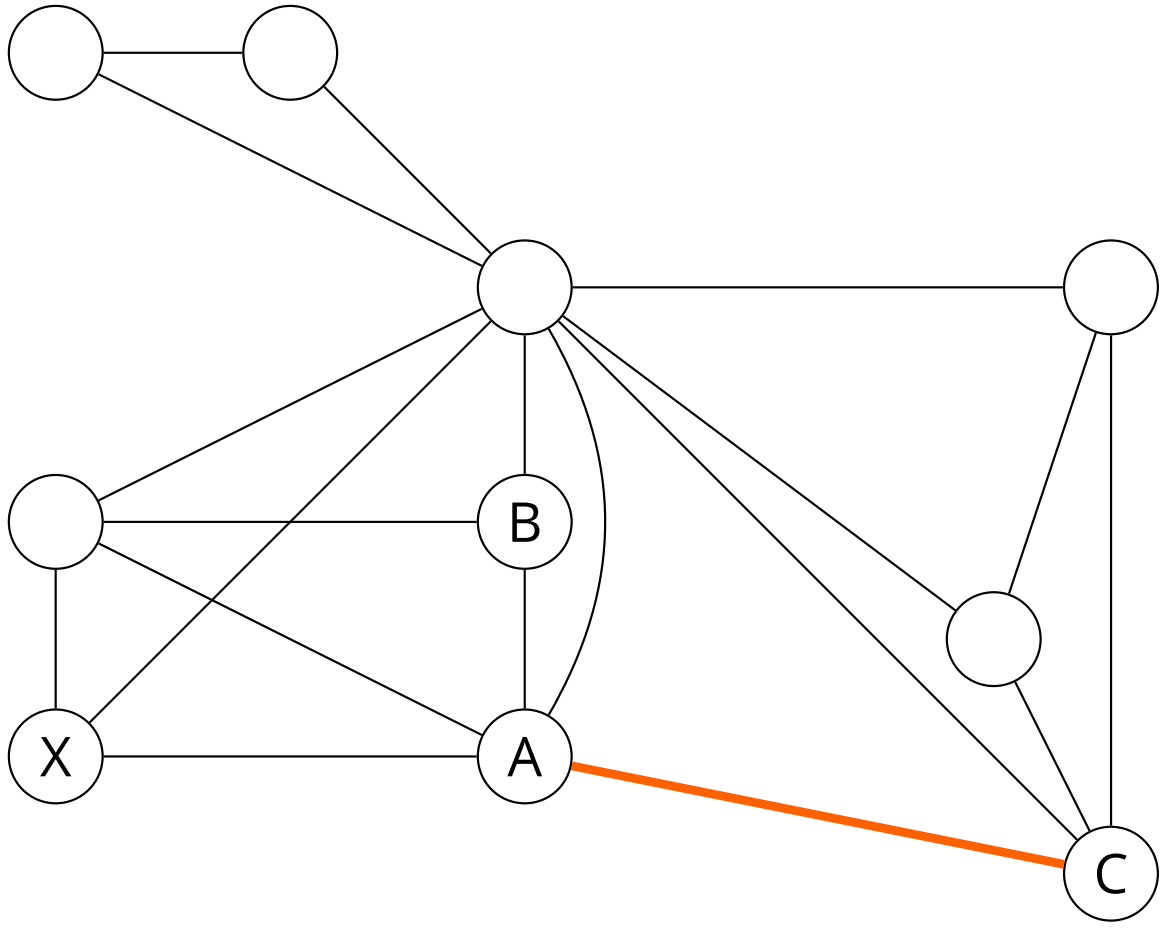
Alternating Matchings



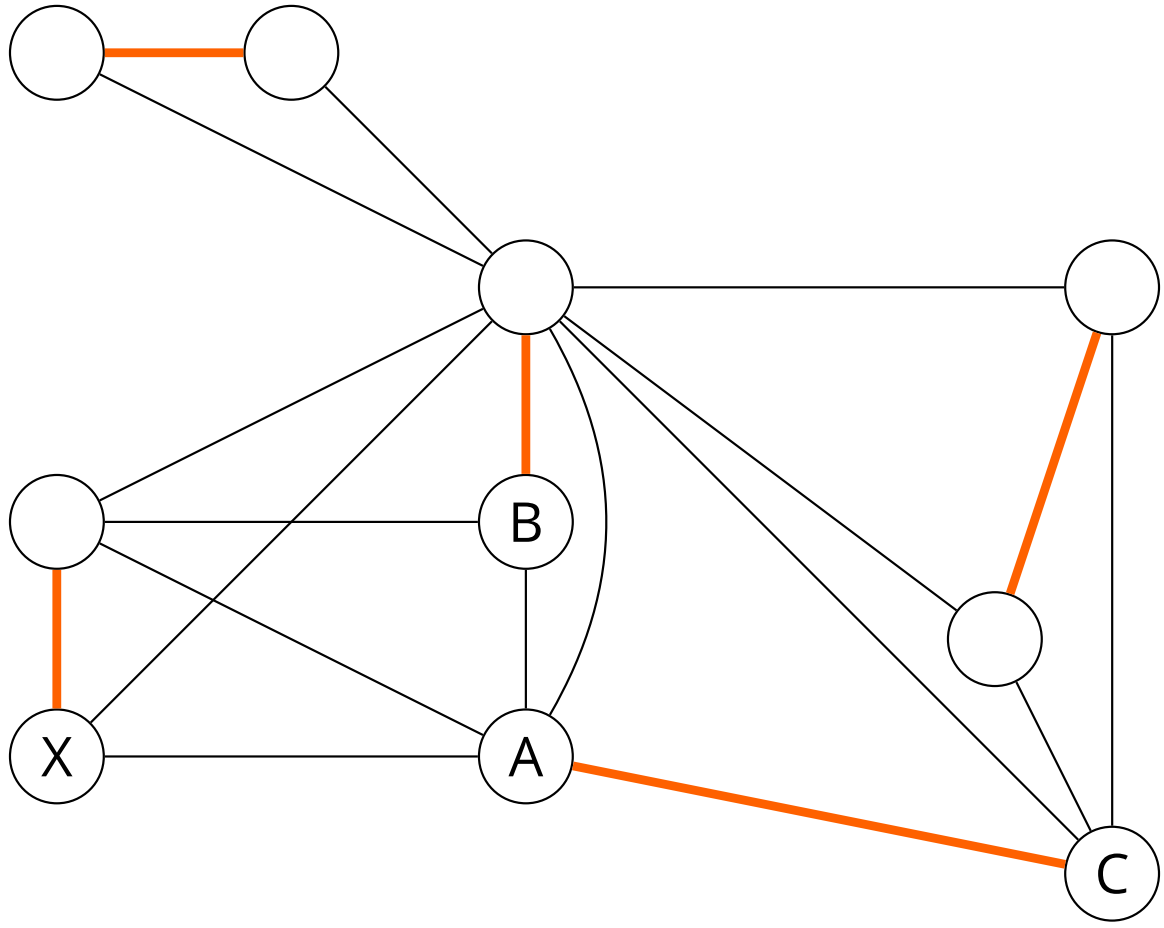
Alternating Matchings



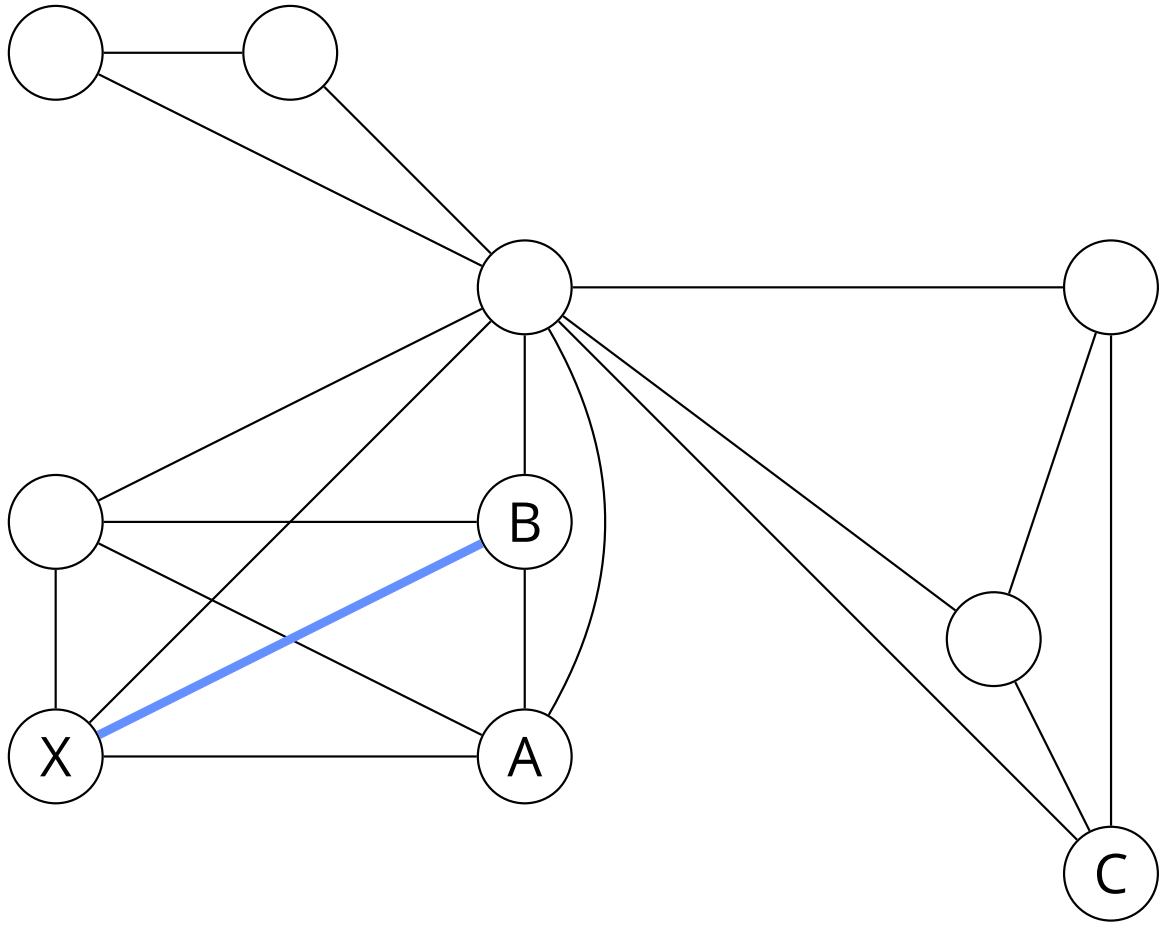
Alternating Matchings



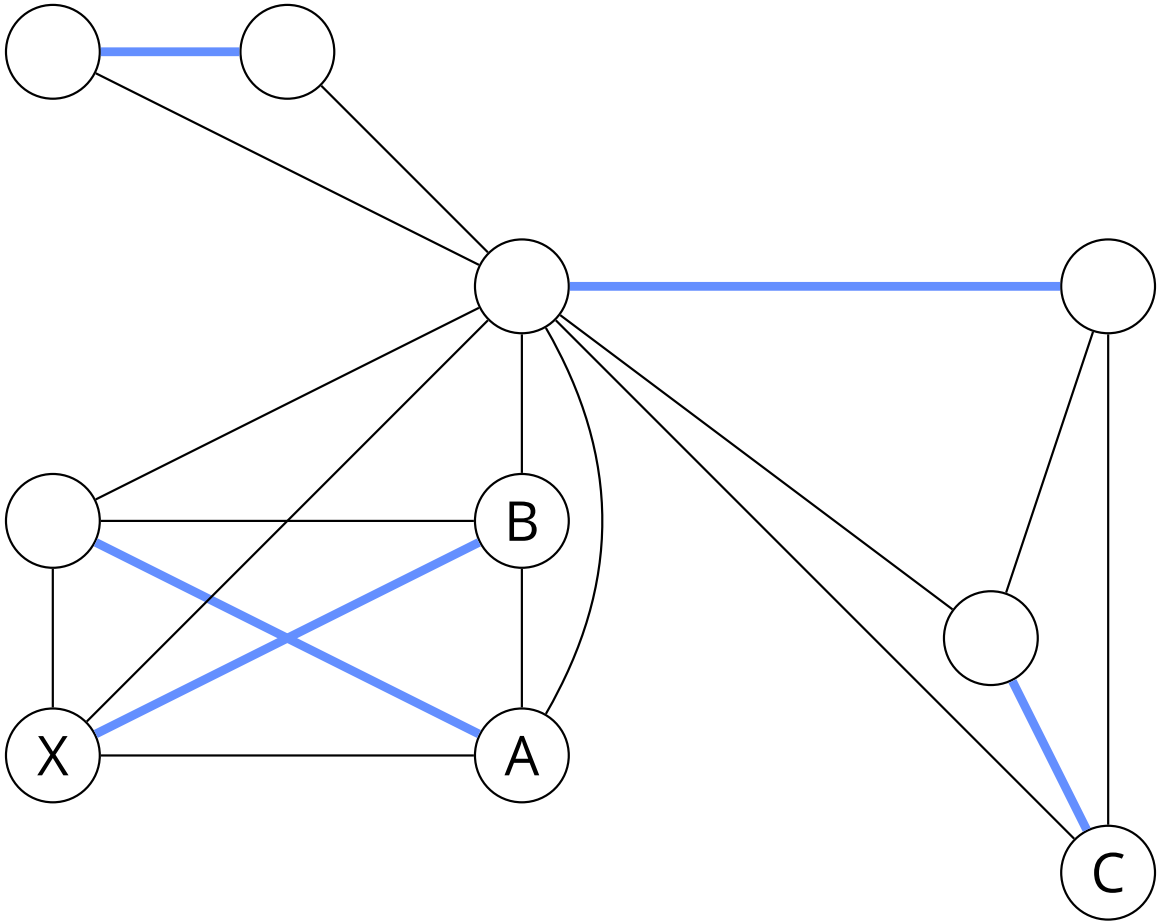
Alternating Matchings



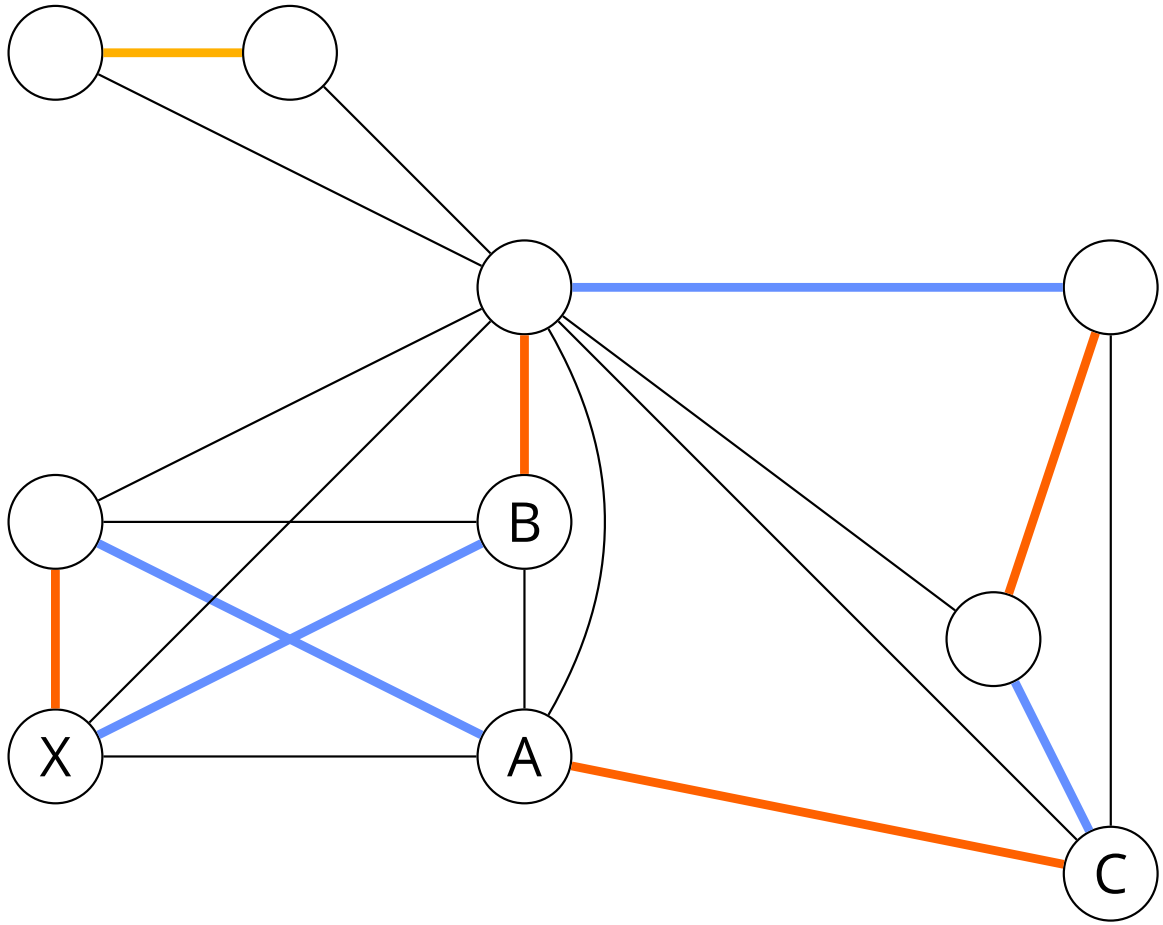
Alternating Matchings



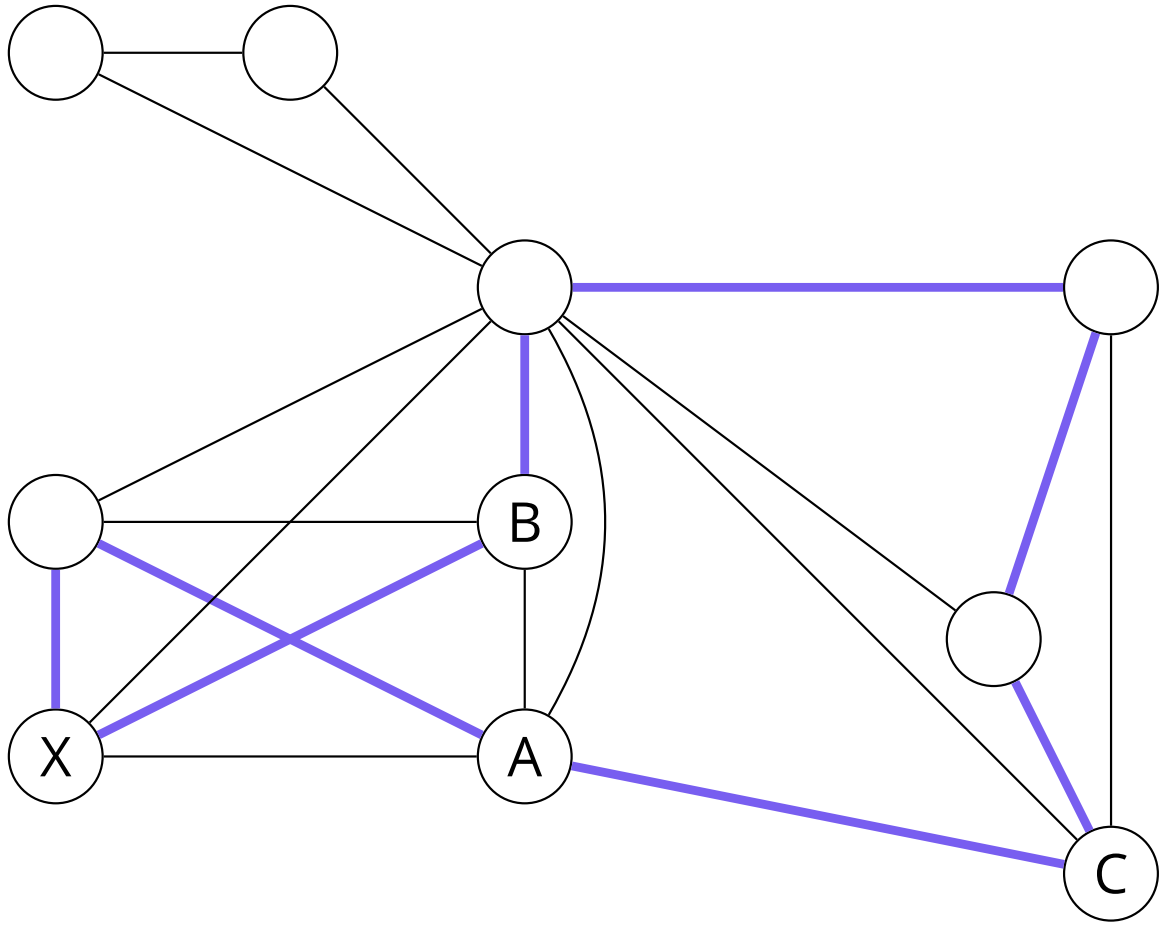
Alternating Matchings



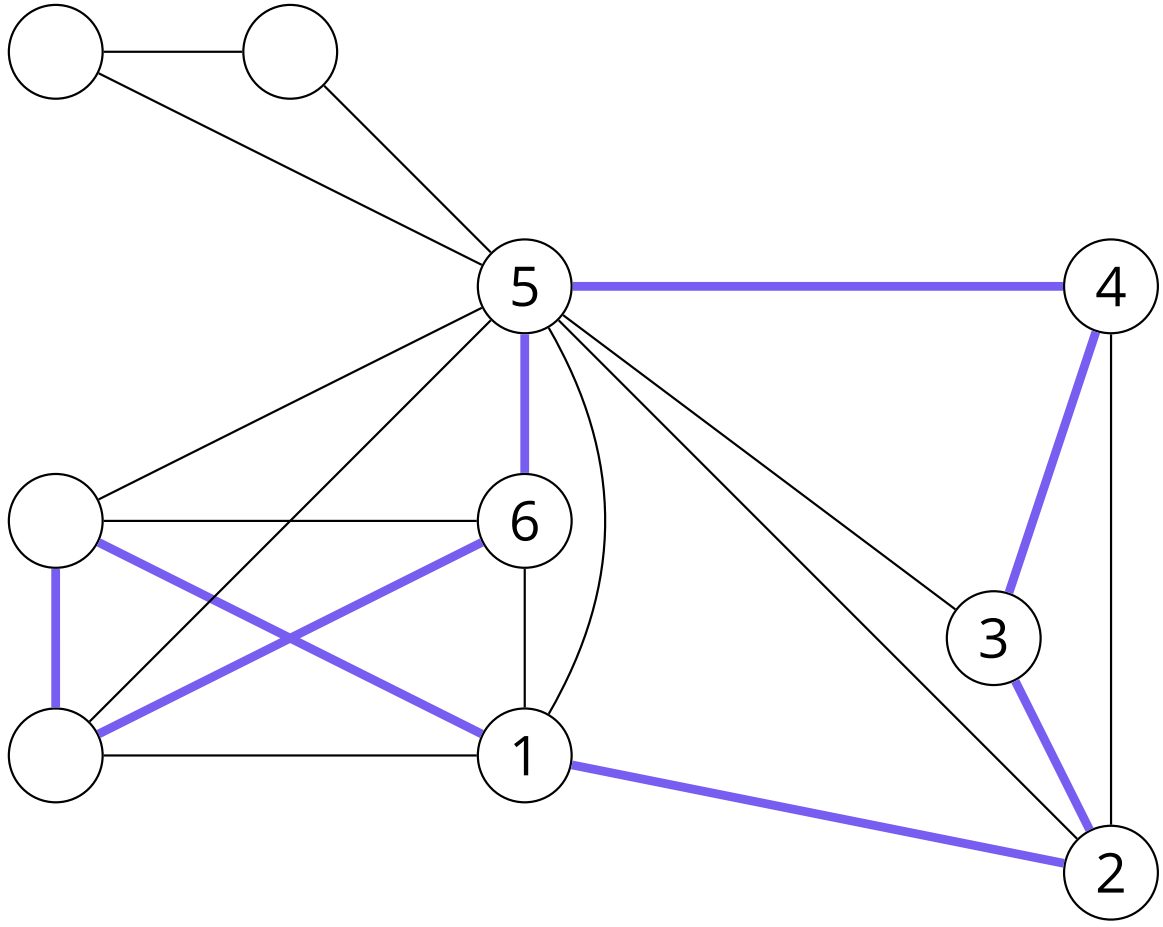
Alternating Matchings



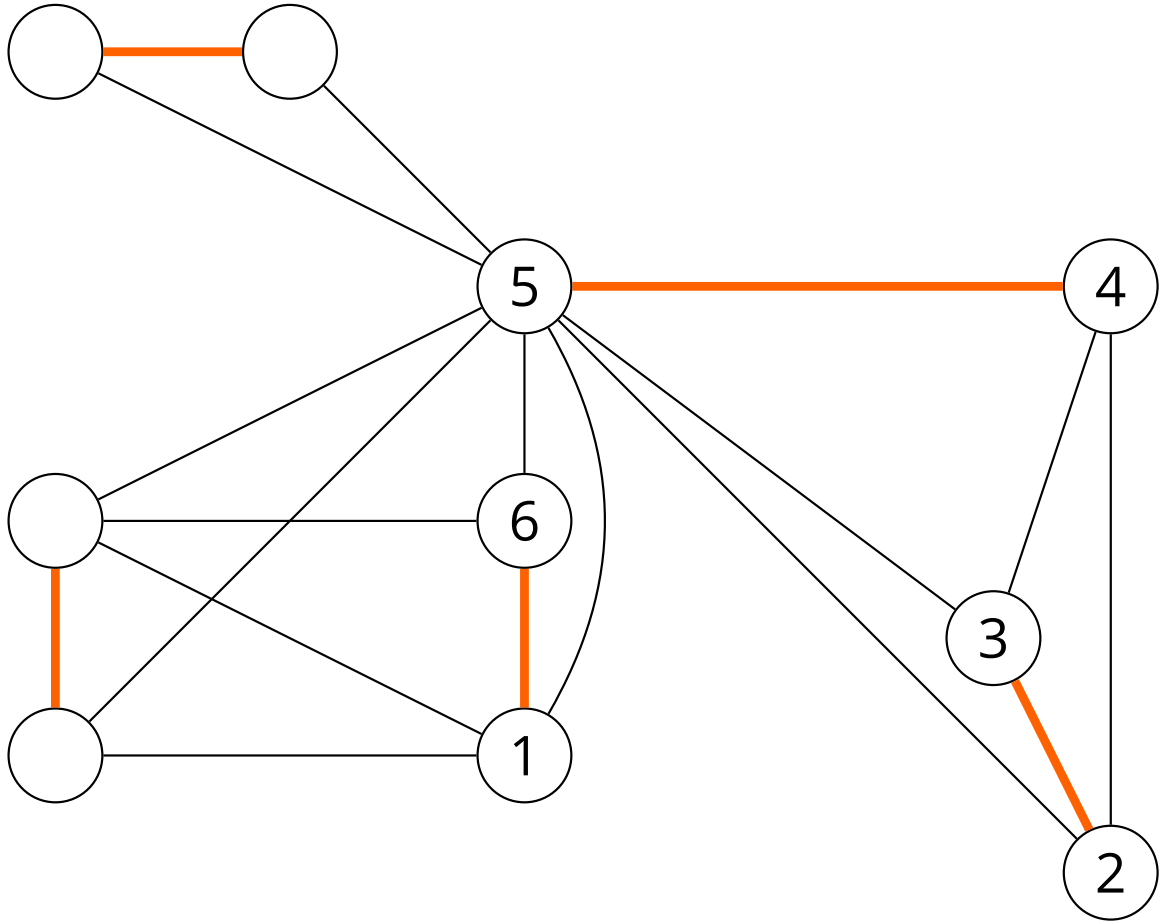
Alternating Matchings



Alternating Matchings



Alternating Matchings



Alternating Matchings: Use in proof

- Construct cycles as symmetric difference of perfect matchings

Alternating Matchings: Use in proof

- Construct cycles as symmetric difference of perfect matchings
- Show that these cycles are alternating

Alternating Matchings: Use in proof

- Construct cycles as symmetric difference of perfect matchings
- Show that these cycles are alternating
- Show that symmetric difference with alternating graph preserves perfect matching

Alternating Matchings: Use in proof

- Construct cycles as symmetric difference of perfect matchings
- Show that these cycles are alternating
- Show that symmetric difference with alternating graph preserves perfect matching
- Construct specific alternating cycle needed.

Symmetric difference: SimpleGraph vs Subgraph

- for `H H' : Subgraph G` it holds that
  `(H △ H').verts = (H \ H').verts ∪ (H' \ H).verts`

Symmetric difference: SimpleGraph vs Subgraph

- for `H H' : Subgraph G` it holds that
  `(H △ H').verts = (H \ H').verts ∪ (H' \ H).verts`
- So, for two perfect matchings `(H △ H').verts = ∅`

Symmetric difference: SimpleGraph vs Subgraph

- for `H H' : Subgraph G` it holds that
  `(H △ H').verts = (H \ H').verts ∪ (H' \ H).verts`
- So, for two perfect matchings `(H △ H').verts = ∅`
- This problem doesn't occur for the symmetric difference on SimpleGraphs

Symmetric difference: SimpleGraph vs Subgraph

- for `H H' : Subgraph G` it holds that
  `(H △ H').verts = (H \ H').verts ∪ (H' \ H).verts`
- So, for two perfect matchings `(H △ H').verts = ∅`
- This problem doesn't occur for the symmetric difference on SimpleGraphs
- `(G △ G').Adj v w ↔ (G.Adj v w ∧ ¬G'.Adj v w) ∨ (¬G.Adj v w ∧ G'.Adj v w)`

IsCycles and IsAlternating

```
def IsCycles (G : SimpleGraph V) :=
  ∀ {⦃v⦄}, (G.neighborSet v).Nonempty →
          (G.neighborSet v).ncard = 2

def IsAlternating (G G' : SimpleGraph V) :=
  ∀ {⦃v w w': V⦄}, w ≠ w' → G.Adj v w → G.Adj v w' →
    (G'.Adj v w ↔ ¬ G'.Adj v w')
```

Symmetric difference of matchings is alternating

```
lemma Subgraph.IsPerfectMatching.isAlternating_symmDiff_left
    {M' : Subgraph G'} (hM : M.IsPerfectMatching)
    (hM' : M'.IsPerfectMatching) :
    (M.spanningCoe △ M'.spanningCoe).IsAlternating
    M.spanningCoe := by
  intro v w w' hww' hvw hvw'
  obtain ⟨v1, hm1, hv1⟩ := hM.1 (hM.2 v)
  obtain ⟨v2, hm2, hv2⟩ := hM'.1 (hM'.2 v)
  simp only [symmDiff_def] at *
  aesop
```

Lessons from formalization

- Work in the “correct” ambient type

Lessons from formalization

- Work in the “correct” ambient type
- Develop the right abstraction

Lessons from formalization

- Work in the “correct” ambient type
- Develop the right abstraction
- Do not mix classical and constructive approaches

Outline

Introduction

Motivation

Tutte’s theorem

Educational formalization project

Conclusion

Context

- Reason for formalizing Tutte’s theorem

Context

- Reason for formalizing Tutte’s theorem
- Gap between available “teachers” and “students”

Context

- Reason for formalizing Tutte’s theorem
- Gap between available “teachers” and “students”
- Gap between tutorials and “blueprint projects”

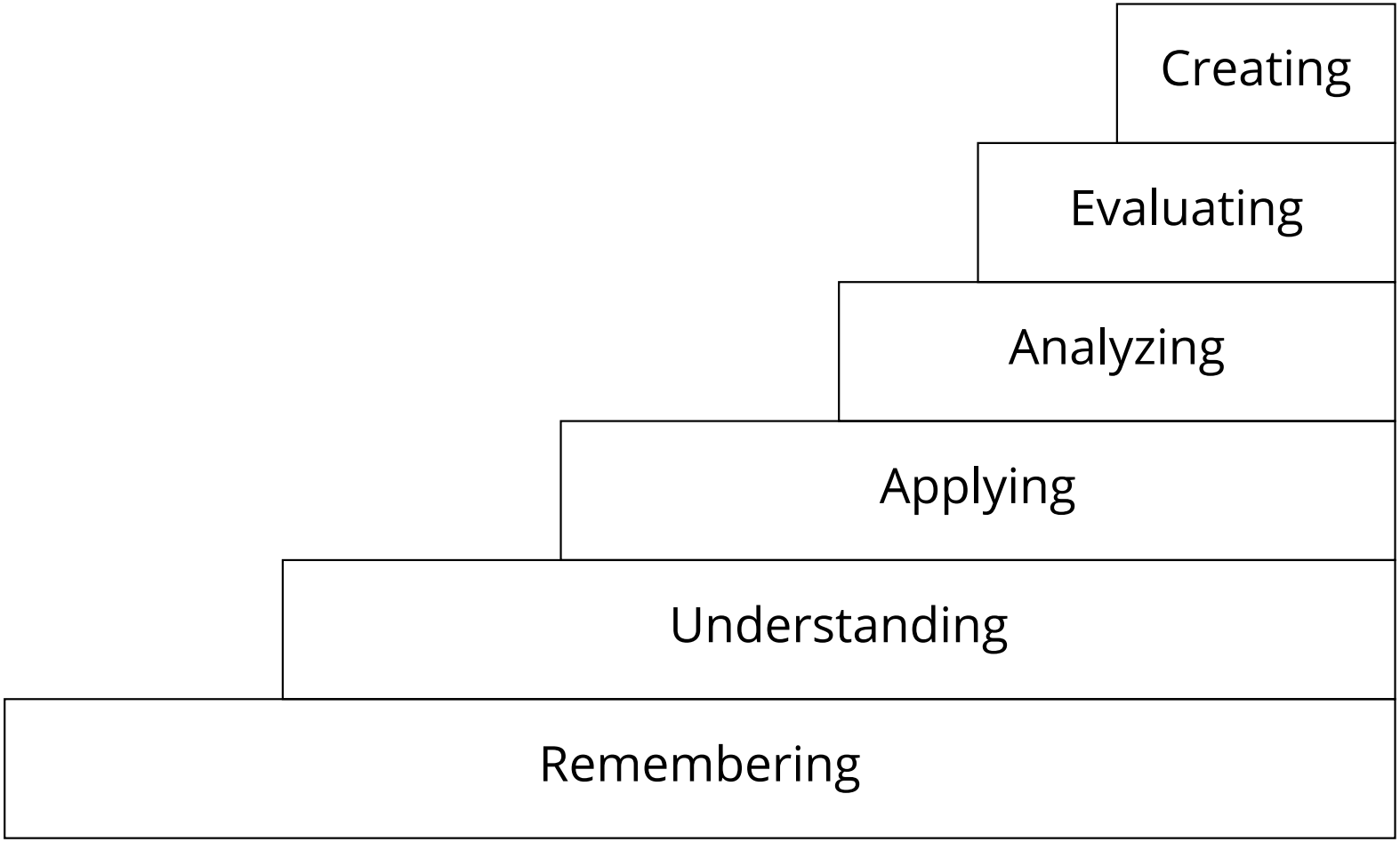
Context

- Reason for formalizing Tutte’s theorem
- Gap between available “teachers” and “students”
- Gap between tutorials and “blueprint projects”
- Goal: Enable educational formalization projects with minimal teacher input.

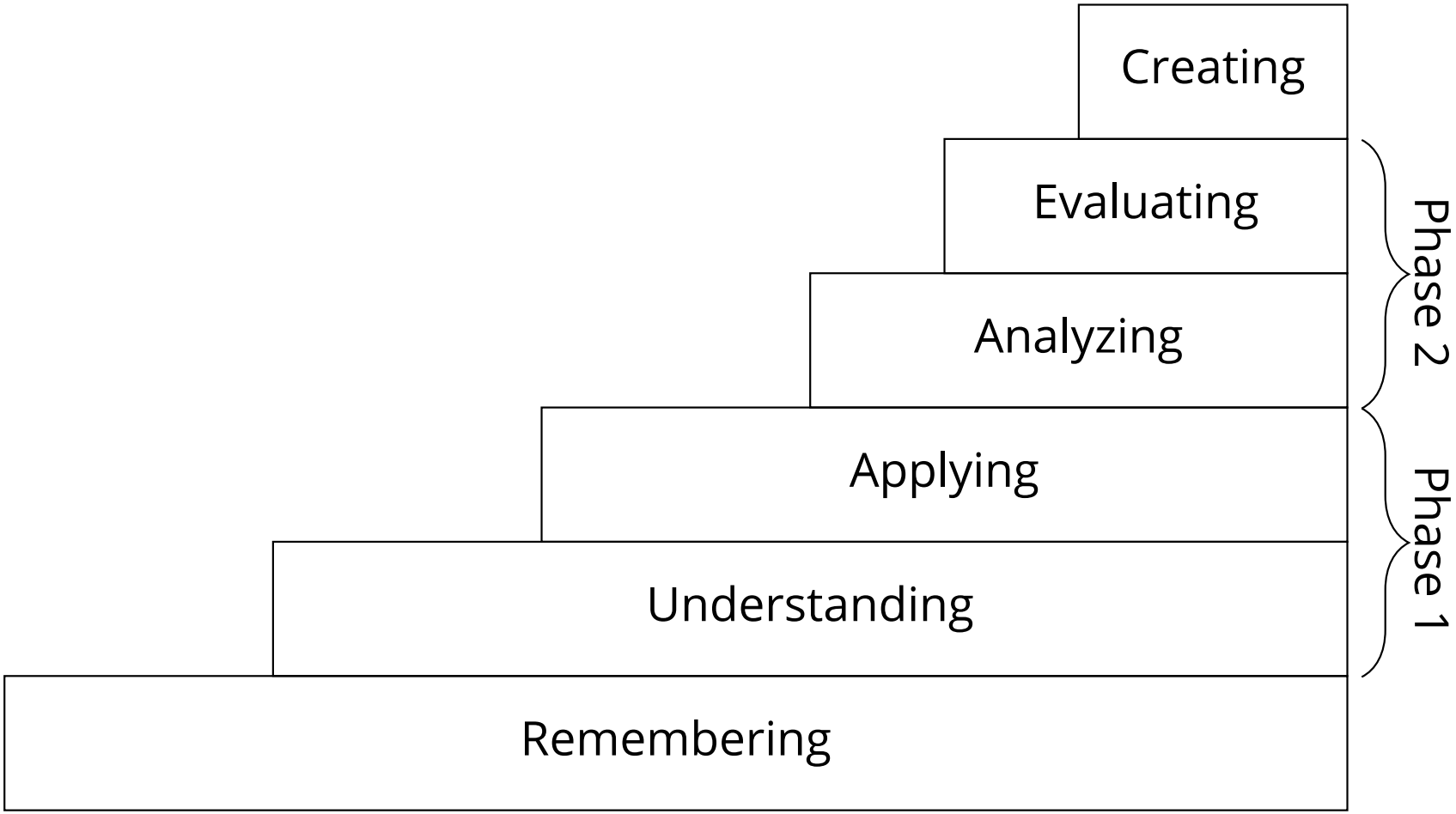
The template

	Phase 1	Phase 2
Deliverable	Initial formalization	Polished formalization
Learning goals	Working with an ITP Proving goals Formulating intermediate goals	Refactoring formal proofs Architecting formal proofs
Teacher role	Provide goal statement Recommend resources	Review formalization
Student role	Focus on learning	Attention for details Attention for structure

Intermezzo: Bloom's taxonomy



Intermezzo: Bloom's taxonomy



Tutte's theorem as an educational formalization project

	Phase 1	Phase 2
Deliverable	5000 lines of spaghetti	36 Pull Requests
Learning goals	Typeclass Synthesis have-tactic structure	Classical approach Golfing
Teacher role Lean Community	Provided goal statement Recommended resources	Reviewed formalization
Student role Pim Otte	Focus on getting a compiling proof	Refactored to mathlib-level

Proposed conditions

- Student: In the Goldilocks zone of Bloom’s taxonomy

Proposed conditions

- Student: In the Goldilocks zone of Bloom’s taxonomy
- Teacher:
  - Select suitable goal

Proposed conditions

- Student: In the Goldilocks zone of Bloom’s taxonomy
- Teacher:
  - Select suitable goal
  - Provide resources

Proposed conditions

- Student: In the Goldilocks zone of Bloom’s taxonomy
- Teacher:
  - Select suitable goal
  - Provide resources
  - Review in the second phase

Proposed conditions

- Student: In the Goldilocks zone of Bloom's taxonomy
- Teacher:
  - Select suitable goal
  - Provide resources
  - Review in the second phase
  - Communicate template to students

Outline

Introduction

Motivation

Tutte’s theorem

Educational formalization project

Conclusion

Conclusion

- Formalized Tutte’s theorem

Conclusion

- Formalized Tutte’s theorem
- Proposed a template for educational formalization projects

Conclusion

- Formalized Tutte's theorem
- Proposed a template for educational formalization projects
- Opportunities: Formalizing graph algorithms, testing the template in traditional setting.

References

- Mohammad Abdulaziz, *A formal correctness proof of edmonds' blossom shrinking algorithm*, 2024.
- Pim Otte, *Tutte's theorem as an educational formalization project*, 2025.

Utrecht University

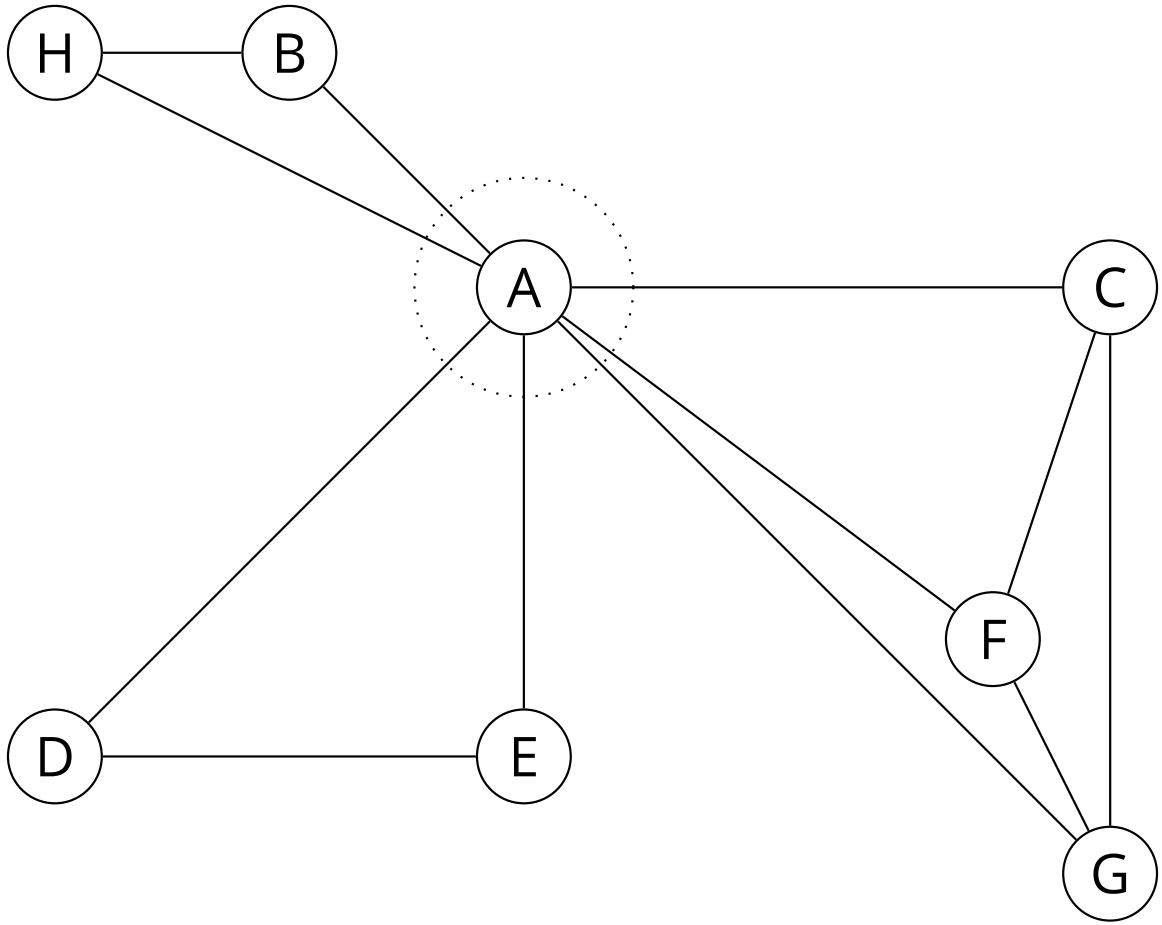
Sharing science,
shaping tomorrow

Thanks to Yaël Dillies, Bhavik Mehta, Kyle Miller and other mathlib reviewers.

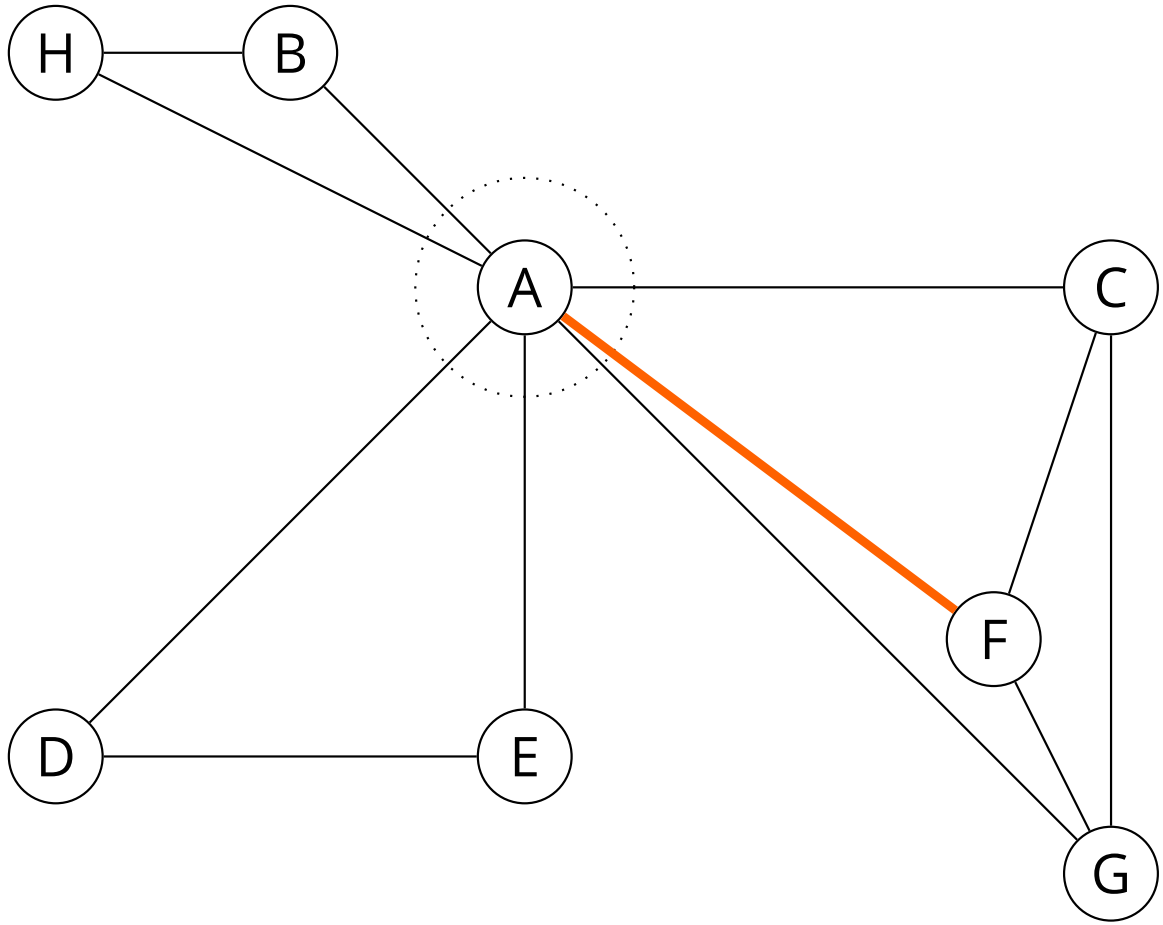
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

pim.otte.dev

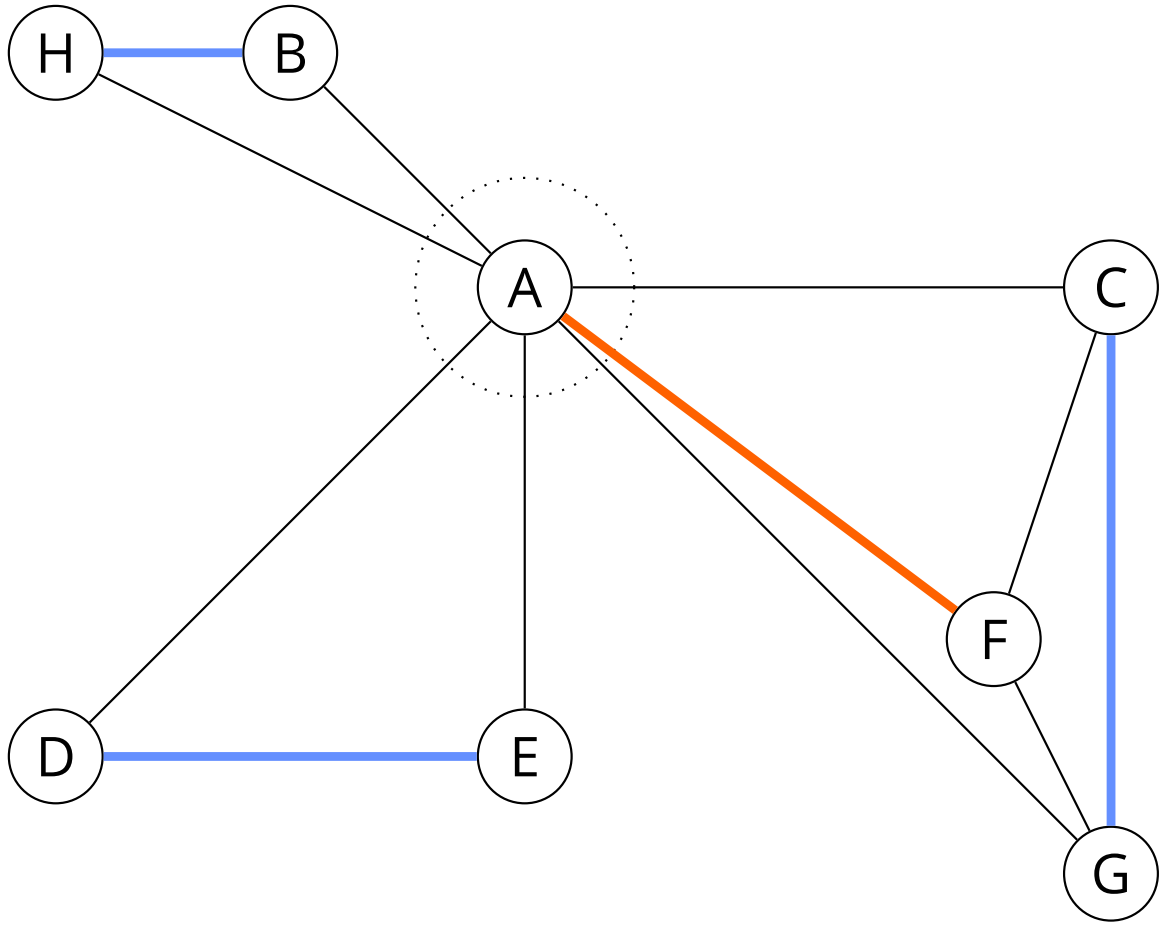
Oops! All cliques



Oops! All cliques



Oops! All cliques



Formalization: Oops! All cliques

```
theorem Subgraph.IsPerfectMatching.exists_of_isClique_supp (hveven :
    Even (Fintype.card V)) (h : ¬G.IsTutteViolator G.universalVerts) (h' : ∀
    (K : G.deleteUniversalVerts.coe.ConnectedComponent),
    G.deleteUniversalVerts.coe.IsClique K.supp) : ∃ (M : Subgraph G),
    M.IsPerfectMatching := by
classical
obtain ⟨M, ⟨hM, hsub⟩⟩ :=
    IsMatching.exists_verts_compl_subset_universalVerts h h'
obtain ⟨M', hM'⟩ := ((G.isClique_universalVerts.subset
    hsub).even_iff_exists_isMatching
(Set.toFinite _)).mp (by simpa [Set.even_ncard_compl_iff hveven,
    -Set.toFinset_card,← Set.ncard_eq_toFinset_card'] using
    hM.even_card)
use M ⊔ M'
have hspan : (M ⊔ M').IsSpanning := by
rw [Subgraph.isSpanning_iff, Subgraph.verts_sup, hM'.1]
exact M.verts.union_compl_self
exact ⟨hM.sup hM'.2 (by
simp only [hM.support_eq_verts, hM'.2.support_eq_verts, hM'.1,
    Subgraph.verts_sup]
exact (Set.disjoint_compl_left_iff_subset.mpr fun {|a|} a ↦ a).symm), hspan⟩
```

Utrecht University

Sharing science,
shaping tomorrow

Thanks to Yaël Dillies, Bhavik Mehta, Kyle Miller and other mathlib reviewers.

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

pim.otte.dev