Week 4: Graph Theory and Structural Induction

CSC 236:Introduction to the Theory of Computation

Summer 2024

Instructor: Lily

Announcement

- Peer review (5 points)
  - [1 point] Complete all your assigned reviews
  - [1 point] For each review accuracy of marking (for now: if you give mark a, and the TA gives mark t
    - $|a - t| \leq 1$: full marks
    - $|a - t| \in (1, 2]$: -0.75
    - $|a - t| \in (2, 3]$: -0.5
    - $|a - t| \in (3,4]$: -0.25
    - $|a - t| > 4$: no marks
- A2 Q1 (a) now unmarked, A2 Q1 (d) question modified, A2 Q2 (a)-(c) hints modified, Q2 (d) removed.

Trees

- Root
- Binary tree
- Height

Recursively Defined Sets

- $\mathbb{N}$
- Sequence of balanced brackets
- Binary trees

Structural Induction

Prove: every non-empty binary tree has one more node than edge.

Recursively define set $S \subseteq \mathbb{N} \times \mathbb{N}$.

- $(0,0) \in S$
- If $(a, b) \in S$, then both $(a+1, b+1) \in S$ and $(a+3, b) \in S$

Define $S' = \{(x, y) \in \mathbb{N} \times \mathbb{N} : (x \geq y) \land (3 | x - y)\}$. Prove that $S = S'$.

Now You Try!

1. Give a recursive definition over the alphabet $\{+, -, (,)\} \cup \mathbb{N}$ of *well-formed* expressions involving addition and subtraction on the natural numbers.

2. A *ternary tree* can have *at most* three children. Prove using structural induction, that for every $n \geq 1$, every non-empty ternary tree of height n has at most $(3^n - 1)/2$ nodes.

Q1. Give a recursive definition over the alphabet $\{+, -, (,)\} \cup \mathbb{N}$ of *well-formed* expressions involving addition and subtraction on the natural numbers.

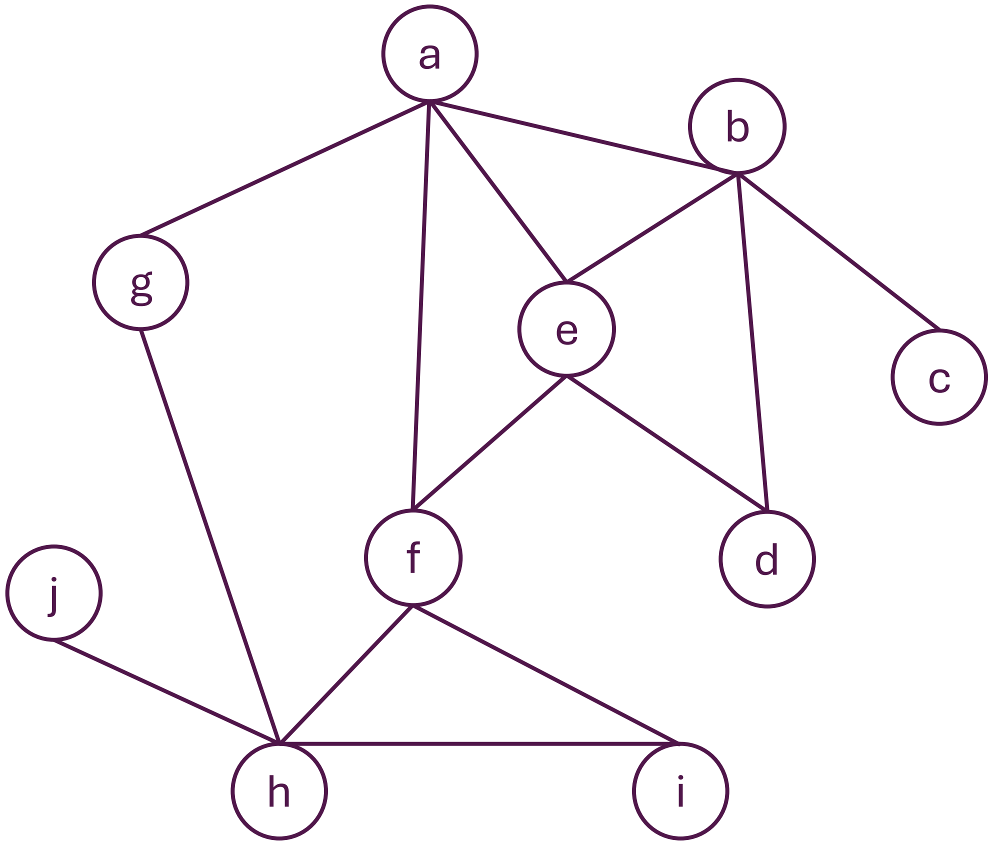
Q2. A *ternary tree* can have *at most* three children. Prove using structural induction, that for every $n \geq 1$, every non-empty ternary tree of height n has at most $(3^n - 1)/2$ nodes.

Minimum Spanning Tree (MST)

- Weighted graph: $G = (V, E)$ and weight function $w: E \to \mathbb{R}$
- Spanning tree: subgraph $T = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$ which is a tree
- Weight of subgraph T:

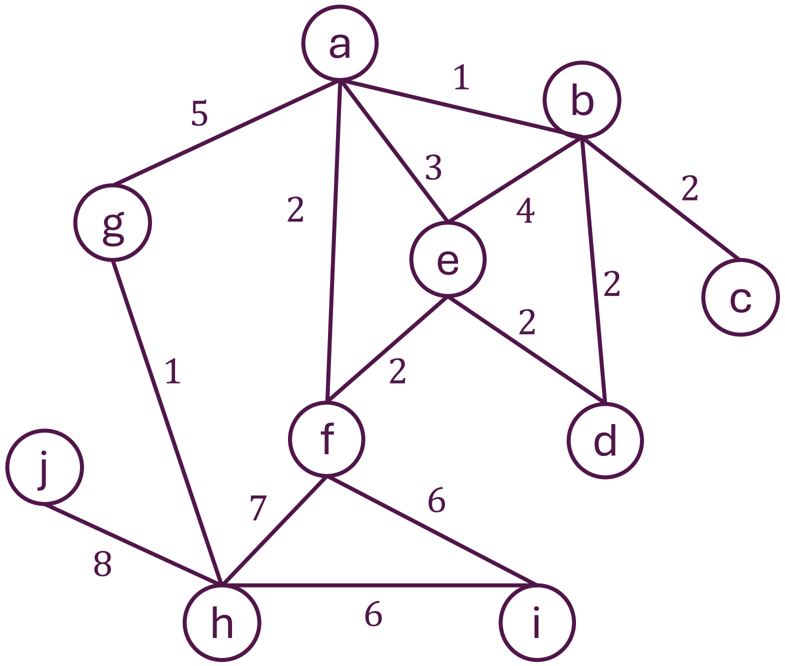
$w(T) = \sum_{e \in E'} w(e)$

- MST: for connected weighted graph G, spanning tree T with minimum weight



Prim's Algorithm

```
def mst_prim(V, E, w)-> list[edges]:
  # Pre: G = (V,E) connected
  # Post: output MST
 T = []
 visited = {a}
 while visited != V:
   (u,v) = min weight edge
   T = T.append((u,v))
   visited.add(v)
 return T
```



Program Correctness (Iterative)

- Preconditions: properties of the input
- Postconditions: properties of the output

Program Correctness. Let f be a function with a set of preconditions and post conditions. Then f is *correct* (with respect to the pre- and postconditions) if for every input $I\ to\ f$, if I satisfies the preconditions, then $f(I)$ terminates and all the postconditions hold after termination.

Correctness of Prim's Algorithm

```
def mst_prim(V, E, w)-> list[edges]:
  # Pre: G = (V,E) connect
  # Post: output MST
  T = []
  visited = {a}
  while visited != V:
    (u,v) = min weight edge
    T = T.append((u,v))
    visited.add(v)
  return T
```

Asymptotic Analysis

```
def mst_prim(V, E, w)-> list[edges]:
  # Pre: G = (V,E) connect
  # Post: output MST
  T = []
  visited = {a}
  while visited != V:
    (u,v) = min weight edge
    T = T.append((u,v))
    visited.add(v)
  return T
```

Recap

- Graph terminology: trees
- Structural induction
  - Recursive definition
- Introduction to proof-of-correctness
- More thorough asymptotic analysis recap

Next time… many more examples of proofs-of-correctness