Graph Theory - Supplementary

COMP9020 Tutorial

JIAPENG WANG

24T3 Week10, H18B & F15A

1 Tutorial Outline

1.1 Definition

Ask yourself these questions below to test your understanding.

- What is the definition of *graph*?$\rightarrow$ undirected, directed
  - How to represent a graph?
- What is a *walk*?
  - What are a *trail* and a *path*? What are a *circuit* and a *cycle*?
  - What is the relationship between these concepts?
- What is *connectedness*?$\rightarrow$ Connected Component
  - What is *acyclic graph*? $\rightarrow$ tree, forest
  - What is *complete graph*? *Complete bipartite graph*?$\rightarrow$ K_n, $K_{m,n}$
  - What are the properties of these concepts?
- What is *isomorphism*? What are the necessary conditions for it?
  - What is *Vertex Degree*?$\rightarrow$ regular graph, degree sequence
- What is the *Chromatic Number* $\chi(G)$ and the *Clique Number* $\kappa(G)$?
  - What are the properties of these concepts?
- What is the *Planar Graph*?
  - How to determine whether a graph is planar or not?
- How to explore a graph?
  - What is *Eulerian* and *Hamiltonian traversals*?
  - How to determine whether a graph has *Euler trail/circuit* or not?
  - How to determine whether a graph has *Hamiltonian path/cycle*?

1.2 Brainstorming

The following questions are open and have no standard answers.

- Why is the concept of *bijection* so important in mathematics?
- Why are graphs so important in computer science?
- What are the applications of graph theory?
- ......

2 Details about Binary Trees

In this section, I will make some problems for you to think about.

Some of these problems are quite challenging, *far* exceeding the difficulty level of the final exam. However, they are highly beneficial for deepening our understanding of some concepts covered this term.

Hints will be provided at the end, but not complete solutions. If you have the time and would like to discuss them with me, feel free to reach out anytime.

2.1 Exercise

Please read the definition of a *binary tree* and complete the following exercises.

Definition:

In computer science, a binary tree is a tree data structure in which each node has at most two children, referred to as the left child and the right child. We assume that empty trees (trees with no nodes) are trivial binary trees.

Expression/ Recursive Definition:

A recursive definition using set theory states that

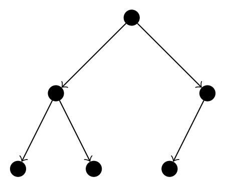
(B) An empty tree, τ, is a binary tree.

(R) An ordered pair $(T_{\text{left}}, T_{\text{right}})$ where T_{left} and T_{right} are binary trees.

Related Concepts:

- A **leaf** in a binary tree is a node that has no successors (i.e. it is of the form (τ, τ)).
- An **internal node** is a node which is not a leaf.
- A **fully-internal node** in a binary tree is a node that has exactly two successors (i.e. it is of the form (T_1, T_2) where $T_1, T_2 \neq \tau$).

Ex1. Please represent the following binary tree in the expression provided above, and answer the number of *leaves*, *internal nodes*, and *fully-internal nodes*.



Ex2. Does there exist a binary tree with an *Euler trail*?

Ex3.1. Define the following functions recursively: [1]

- `count(T)`: the number of nodes in a binary tree T.
- `leaves(T)`: the number of leaves in a binary tree T.
- `f_internal(T)`: the number of fully-internal nodes in a binary tree T.

Ex3.2. If T is a binary tree, let P(T) be the proposition that

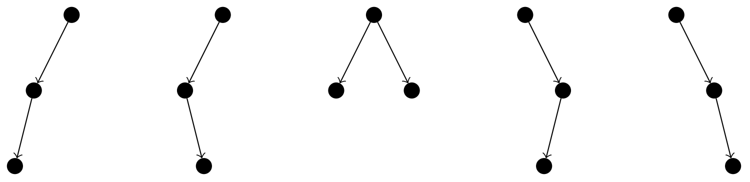
$$\texttt{leaves(T)} = \texttt{f\_internal(T)} + 1.$$

Prove that P(T) holds for all binary trees T.

Ex3.3. Let `occur(T)` denote the number of occurrences of τ in the representation of the binary tree T. Prove that for any binary tree T,

$$\texttt{occur(T)} = \texttt{count(T)} + 1.$$

Ex4.1. Let T(n) denote the number of binary trees with n nodes. For example, T(3) = 5 because there are five binary trees with three nodes:



Using the recursive definition of a binary tree structure, or otherwise, derive a recurrence equation for T(n).

Ex4.2. A **full binary tree** is a non-empty binary tree where every node has either two non-empty children (i.e., a fully-internal node) or two empty children (i.e., a leaf). Let B(n) denote the number of full binary trees with n nodes.
Derive an expression for B(n), involving T(n′) where n′ ≤ n.

Ex4.3. The n-th Catalan number, denoted as C_n, is given by the formula:

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Prove that

a) $C_n = \binom{2n}{n} - \binom{2n}{n-1}$.
b) T(n) $= C_n$.

Ex5. Each ticket for a football match costs 50 dollars, and the ticket office has no change available. There are n people, each carrying a 50-dollar bill, and another n people, each carrying a 100-dollar bill, lining up to buy tickets. Each person can only purchase one ticket. If every person can buy their ticket without having to wait for change, it is called a *smooth purchase*. Find the probability of a *smooth purchase*.

[1] For technical reasons (that will become apparent) we assume that an empty tree has 0 leaves and −1 fully-internal nodes.

2.2 Hint

Ex1. $(((\tau,\tau),(\tau,\tau)),((\tau,\tau),\tau))$; 3, 3, 2.

Ex2. Yes. For example, $(((\tau,\tau),(\tau,\tau)),((\tau,\tau),(\tau,\tau)))$.

Ex3.1.

The function `count` can be recursively defined as follows:

(`count.B`) If $T=\tau$, then $\texttt{count}(T)=0$;

(`count.R`) If $T=(T_{\text{left}},T_{\text{right}})$, then

$$\texttt{count}(T)=1+\texttt{count}(T_{\text{left}})+\texttt{count}(T_{\text{right}}).$$

The function `leaves` can be recursively defined as follows:

(`leaves.B`) If $T=\tau$, then $\texttt{leaves}(T)=0$; If $T=(\tau,\tau)$, then $\texttt{leaves}(T)=1$;

(`leaves.R`) If $T=(T_{\text{left}},T_{\text{right}})$, T_{left} and T_{right} are not both τ, then

$$\texttt{leaves}(T)=\texttt{leaves}(T_{\text{left}})+\texttt{leaves}(T_{\text{right}}).$$

The function `f_internal` can be recursively defined as follows:

(`f_internal.B`) If $T=\tau$, then $\texttt{f\_internal}(T)=-1$; If $T=(\tau,\tau)$, then $\texttt{f\_internal}(T)=0$;

(`f_internal.R`) If $T=(T_{\text{left}},T_{\text{right}})$, T_{left} and T_{right} are not both τ, then

$$\texttt{f\_internal}(T)=1+\texttt{f\_internal}(T_{\text{left}})+\texttt{f\_internal}(T_{\text{right}}).$$

Ex3.2. By structure induction.

Ex3.3. Note that the following two formulas hold, along with the conclusion of *Ex3.2.*.

$$\texttt{occur}(T)=2\texttt{leaves}(T)+\big(\texttt{internal}(T)-\texttt{f\_internal}(T)\big),\text{ and}$$

$$\texttt{count}(T)=\texttt{leaves}(T)+\texttt{internal}(T).$$

Ex4.1. Please note that different structures of left and right subtrees correspond to different tree structures. Therefore, we can leverage this characteristic to define a recursive approach.

$$T(0)=T(1)=1,\quad T(n)=\sum_{i=0}^{n-1}T(i)T(n-1-i).$$

Ex4.2. Find a way to show that there exists a bijection $f:T_k\to B_{2k+1}$.

$$B(n)=\begin{cases}T\left(\frac{n-1}{2}\right), & \text{if } n \text{ is an odd positive integer,}\\ 0, & \text{if } n \text{ is an even positive integer.}\end{cases}$$

Ex4.3a. By the definition of *combination numbers.*

Ex4.3b. This is an exceptionally challenging problem. The core idea of the proof is to transform a full binary tree with $2n+1$ nodes into a sequence of n left arrows and n right arrows. The goal is to calculate the number of such sequences in which, at any position in the sequence, the cumulative number of left arrows from the start to that position is always greater than or equal to the cumulative number of right arrows. Selecting n left arrows from $2n$ positions yields $\binom{2n}{n}$ combinations. The critical step lies in excluding those sequences that do not meet the condition. Providing a full explanation would take considerable time (and is omitted here), but I encourage those interested to think it through independently.

Ex5. The cardinality of the sample space is $\binom{2n}{n}$. Through isomorphism, this problem can be transformed into the task of placing n left arrows (representing \$50) and n right arrows (representing \$100) such that, at any position in the sequence, the cumulative number of left arrows from the start to that position is not less than the cumulative number of right arrows. This corresponds directly to the problem analyzed in *Ex 4.3b.* Thus, the number of *smooth purchases* is C_n. Consequently, the probability of a *smooth purchase* is $\frac{C_n}{\binom{2n}{n}} = \frac{1}{n+1}$.