Assignment 1 – Discrete Review, Asymptotic Analysis & Graphs

CS 577 Summer 2023

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Related Readings: `http://pages.cs.wisc.edu/~hasti/cs240/readings/`

Name: Yuhan Wang Wisc id: ywang2558

Logic

1. Using a truth table, show the equivalence of the following statements.

   (a) $P \vee (\neg P \wedge Q) \equiv P \vee Q$

Solution:

P	Q	$\neg P$	$\neg P \wedge Q$	$P \vee (\neg P \wedge Q)$	$P \vee Q$
T	T	F	F	T	T
T	F	F	F	T	T
F	T	T	T	T	T
F	F	T	F	F	F

   (b) $\neg P \vee \neg Q \equiv \neg(P \wedge Q)$

Solution:

P	Q	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$	$\neg(P \wedge Q)$
T	T	F	F	F	F
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	T	T

(c) $\neg P \vee P \equiv \text{true}$

Solution:

P	$\neg P$	$\neg P \vee P$	true
T	F	T	T
F	T	T	T

(d) $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

Solution:

P	Q	R	$Q \wedge R$	$P \vee (Q \wedge R)$	$P \vee Q$	$P \vee R$	$(P \vee Q) \wedge (P \vee R)$
T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	F
F	F	T	F	F	F	T	F
F	F	F	F	F	F	F	F

Sets

2. Based on the definitions of the sets A and B, calculate the following: $|A|$, $|B|$, $A \cup B$, $A \cap B$, $A \setminus B$, $B \setminus A$.

   (a) $A = \{1, 2, 6, 10\}$ and $B = \{2, 4, 9, 10\}$

   Solution: $|A| = 4$, $|B| = 4$, $A \cup B = \{1, 2, 4, 6, 9, 10\}$, $A \cap B = \{2, 10\}$, $A \setminus B = \{1, 6\}$, $B \setminus A = \{4, 9\}$

   (b) $A = \{x \mid x \in \mathbb{N}\}$ and $B = \{x \in \mathbb{N} \mid x \text{ is even}\}$

   Solution: $|A| = \infty$, $|B| = \infty$, $A \cup B = \{x \mid x \in \mathbb{N}\}$, $A \cap B = \{x \in \mathbb{N} \mid x \text{ is even}\}$, $A \setminus B = \{x \in \mathbb{N} \mid x \text{ is odd}\}$, $B \setminus A = \emptyset$

Relations and Functions

3. For each of the following relations, indicate if it is reflexive, antireflexive, symmetric, antisymmetric, or transitive.

   (a) $\{(x, y) : x \leq y\}$

   Solution: It is reflexive, antisymmetric, and transitive.

   (b) $\{(x, y) : x > y\}$

   Solution: It is antireflexive, antisymmetric, and transitive.

(c) $\{(x, y) : x < y\}$

Solution: It is antireflexive, asymmetric, and transitive.

(d) $\{(x, y) : x = y\}$

Solution: It is reflexive, symmetric, and transitive.

4. For each of the following functions (assume that they are all $f : \mathbb{Z} \to \mathbb{Z}$), indicate if it is surjective (onto), injective (one-to-one), or bijective.

(a) $f(x) = x$

Solution: It is bijective.

(b) $f(x) = 2x - 3$

Solution: It is not surjective but it is injective.

(c) $f(x) = x^2$

Solution: It is not surjective and it is not injective.

5. Show that $h(x) = g(f(x))$ is a bijection if $g(x)$ and $f(x)$ are bijections.

Solution:
Injective: Assume $h(x) = h(y)$. Then $g(f(x)) = g(f(y))$. Since $g(x)$ is injective, $f(x) = f(y)$. Since $f(x)$ is injective, $x = y$.
Surjective: Let $z \in \mathbb{Z}$. Since $g(x)$ is surjective, there exists $y \in \mathbb{Z}$ such that $g(y) = z$. Since $f(x)$ is surjective, there exists $x \in \mathbb{Z}$ such that $f(x) = y$. Then $h(x) = g(f(x)) = g(y) = z$.
Therefore $h(x)$ is bijective.

Induction

6. Prove the following by induction.

   (a) $\sum_{i=1}^{n} i = n(n+1)/2$

Solution:
Base case: $P(1) = 1 = 1(1+1)/2$
Inductive step: Assume $P(k)$ is true for some $k \geq 1$. Then $\sum_{i=1}^{k} i = k(k+1)/2$.
Consider $P(k+1)$.

$$\begin{aligned}\sum_{i=1}^{k+1} i &= \sum_{i=1}^{k} i + (k+1)\\ &= \frac{k(k+1)}{2} + (k+1)\\ &= \frac{(k+1)(k+2)}{2}\end{aligned}$$

Thus $P(k+1)$ is true.
By the principle of mathematical induction, $P(n)$ is true for all $n \geq 1$.

   (b) $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$

Solution:
$P(1) = 1 = 1(1+1)(2(1)+1)/6$
Assume $P(k)$ is true for some $k \geq 1$. Then $\sum_{i=1}^{k} i^2 = k(k+1)(2k+1)/6$.
Consider $P(k+1)$.

$$\begin{aligned}\sum_{i=1}^{k+1} i^2 &= \sum_{i=1}^{k} i^2 + (k+1)^2\\ &= \frac{k(k+1)(2k+1)}{6} + (k+1)^2\\ &= \frac{(k+1)(k+2)(2k+3)}{6}\end{aligned}$$

Thus $P(k+1)$ is true.
By the principle of mathematical induction, $P(n)$ is true for all $n \geq 1$.

   (c) $\sum_{i=1}^{n} i^3 = n^2(n+1)^2/4$

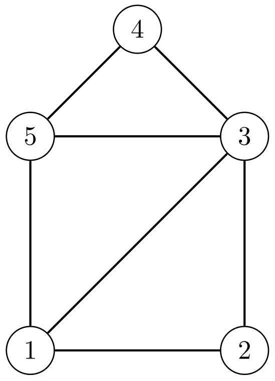
Solution:
$P(1) = 1 = 1^2(1+1)^2/4$
Assume $P(k)$ is true for some $k \geq 1$. Then $\sum_{i=1}^{k} i^3 = k^2(k+1)^2/4$.
Consider $P(k+1)$.

$$\begin{aligned}\sum_{i=1}^{k+1} i^3 &= \sum_{i=1}^{k} i^3 + (k+1)^3 \\ &= \frac{k^2(k+1)^2}{4} + (k+1)^3 \\ &= \frac{(k+1)^2(k^2+4k+4)}{4} \\ &= \frac{(k+1)^2(k+2)^2}{4}\end{aligned}$$

Thus $P(k+1)$ is true.
By the principle of mathematical induction, $P(n)$ is true for all $n \geq 1$.

Graphs and Trees

7. Give the adjacency matrix, adjacency list, edge list, and incidence matrix for the following graph.



Solution:
adjacency matrix:

	1	2	3	4	5
1	0	1	1	0	1
2	1	0	1	0	0
3	1	1	0	1	1
4	0	0	1	0	1
5	1	0	1	1	0

adjacency list:
1: 2, 3, 5
2: 1, 3
3: 1, 2, 4, 5
4: 3, 5
5: 1, 3, 4
edge list:
(1, 2), (1, 3), (1, 5), (2, 3), (3, 4), (3, 5), (4, 5)
incidence matrix:

	(1,2)	(1,3)	(1,5)	(2,3)	(3,4)	(3,5)	(4,5)
1	1	1	1	0	0	0	0
2	-1	0	0	1	0	0	0
3	0	-1	0	-1	1	1	0
4	0	0	0	0	-1	0	1
5	0	0	-1	0	0	-1	-1

8. How many edges are there is a complete graph of size n? Prove by induction.

Solution: For $n = 1$, there are 0 edges.
Assume $P(k)$ is true for some $k \geq 1$. Then there are $k(k-1)/2$ edges in a complete graph of size k.
Consider $P(k+1)$.

$$\begin{aligned}\frac{k(k-1)}{2} + k &= \frac{k^2 - k + 2k}{2} \\ &= \frac{k^2 + k}{2} \\ &= \frac{k(k+1)}{2}\end{aligned}$$

Thus $P(k+1)$ is true.
By the principle of mathematical induction, $P(n)$ is true for all $n \geq 1$.

9. Draw all possible (unlabelled) trees with 4 nodes.

Solution:

10. Show by induction that, for all trees, $|E| = |V| - 1$.

Solution: For $n = 1$, there are 0 edges.
Assume $P(k)$ is true for some $k \geq 1$. Then there are $k - 1$ edges in a tree of size k.
Consider $P(k + 1)$.

$$k - 1 + 1 = k$$

This is true because a node can only be connected to one other node in a tree. Thus $P(k + 1)$ is true.
By the principle of mathematical induction, $P(n)$ is true for all $n \geq 1$.

Counting

11. How many 3 digit pin codes are there?

Solution: $10^3 = 1000$

12. What is the expression for the sum of the ith line (indexing starts at 1) of the following:

1
2 3
4 5 6
7 8 9 10
$\vdots$

Solution:
$i(i + 1)/2$ is the last number in the ith line.
The first number in the ith line is $i(i - 1)/2 + 1$.
Thus the sum of the ith line is:

$$\left(\frac{i(i-1)}{2} + 1 + \frac{i(i+1)}{2}\right) \times \frac{i}{2} = \frac{i^3 + i}{2}$$

13. A standard deck of 52 cards has 4 suits, and each suit has card number 1 (ace) to 10, a jack, a queen, and a king. A standard poker hand has 5 cards. For the following, how many ways can the described hand be drawn from a standard deck.
    (a) A royal flush: all 5 cards have the same suit and are 10, jack, queen, king, ace.

Solution: 4

(b) A straight flush: all 5 cards have the same suit and are in sequence, but not a royal flush.

Solution: $4 \times 10 - 4 = 36$

(c) A flush: all 5 cards have the same suit, but not a royal or straight flush.

Solution: $4 \times \binom{13}{5} - 4 - 36 = 5108$

(d) Only one pair (2 of the 5 cards have the same number/rank, while the remaining 3 cards all have different numbers/ranks):

Solution: $13 \times \binom{4}{2} \times \binom{12}{3} \times 4^3 = 1098240$

Proofs

14. Show that $2x$ is even for all $x \in \mathbb{N}$.

(a) By direct proof.

Solution: Let $x \in \mathbb{N}$.
Then $2x/2 = x \in \mathbb{N}$.
Thus $2x$ is even.

(b) By contradiction.

Solution: Assume $2x$ is odd.
Then $2x = 2k + 1$ for some $k \in \mathbb{N}$.
But $2x/2 = x = k + 1/2 \notin \mathbb{N}$.
Thus $2x$ is even.

15. For all $x, y \in \mathbb{R}$, show that $|x + y| \leq |x| + |y|$. (Hint: use proof by cases.)

Solution: Let $x, y \in \mathbb{R}$.
Note that $|x| \geq x$ and $|x| \geq -x$, and similarly for y.
Consider the following cases:

- $x + y \geq 0$
  Then $|x + y| = x + y \leq |x| + |y|$.
- $x + y < 0$
  Then $|x + y| = -(x + y) = -x - y \leq |x| + |y|$.

Thus $|x + y| \leq |x| + |y|$.

Program Correctness (and Invariants)

16. For the following algorithms, describe the loop invariant(s) and prove that they are sound and complete.

(a)

```
Algorithm 1: findMin
  Input: a: A non-empty array of integers (indexed starting at 1)
  Output: The smallest element in the array
  begin
    $min \leftarrow \infty$
    for $i \leftarrow 1$ to $len(a)$ do
      if $a[i] < min$ then
        $min \leftarrow a[i]$
      end
    end
    return $min$
  end
```

Solution: For the loop invariant, we use the following:

- **Initialization**:
  Before the first iteration, $min = \infty$.
  Thus min is the smallest element in the array.
- **Maintenance**:
  Assume min is the smallest element in the array before the ith iteration.
  Then $min \leq a[i]$.
  After the ith iteration, min is updated to $a[i]$ if $a[i] < min$.
  Thus min is the smallest element in the array after the ith iteration.
- **Termination**:
  The loop terminates when $i = \text{len}(a) + 1$.
  By the maintenance property, min is the smallest element in the array.
  Thus min is the smallest element in the array.

For soundness, we have shown that the loop invariant holds before the first iteration, and that if it holds before an iteration, it holds after the iteration.
For completeness, we have shown that if the loop terminates, the loop invariant holds for all inputs a.

(b)

Algorithm 2: InsertionSort

```
Input: $a$: A non-empty array of integers (indexed starting at 1)
Output: $a$ sorted from largest to smallest
begin
    for $i \leftarrow 2$ to $len(a)$ do
        $val \leftarrow a[i]$
        for $j \leftarrow 1$ to $i - 1$ do
            if $val > a[j]$ then
                shift $a[j..i-1]$ to $a[j+1..i]$
                $a[j] \leftarrow val$
                break
            end
        end
    end
    return $a$
end
```

Solution:

- **Sound**:
  The loop invariant is that $a[1..i-1]$ is sorted from largest to smallest.
  This is true before the first iteration of the outer loop, since $a[1]$ is trivially sorted.
  Now assume that $a[1..i-2]$ is sorted from largest to smallest.
  Then the inner loop shifts $a[j..i-1]$ to $a[j+1..i]$ until $a[j] \geq val$.
  Thus $a[1..i-1]$ is sorted from largest to smallest.

- **Complete**:
  The loop terminates when $j = 1$ and $a[j] \geq val$.
  Thus $a[1..i]$ is sorted from largest to smallest.
  Since i is incremented after each iteration of the outer loop, the algorithm terminates with $a[1..len(a)]$ sorted from largest to smallest for all inputs a.

Recurrences

17. Solve the following recurrences.

(a) $c_0 = 1; c_n = c_{n-1} + 4$

Solution: $c_n = c_{n-1} + 4 = c_{n-2} + 4 + 4 = c_{n-3} + 4 + 4 + 4 = \ldots = c_0 + 4n = 1 + 4n$

(b) $d_0 = 4; d_n = 3 \cdot d_{n-1}$

Solution: $d_n = 3 \cdot d_{n-1} = 3 \cdot 3 \cdot d_{n-2} = 3^2 \cdot 3 \cdot d_{n-3} = \ldots = 3^n \cdot 4$

(c) $T(1) = 1; T(n) = 2T(n/2) + n$ (An upper bound is sufficient.)

Solution:
$T(n) = 2T(n/2) + n = 2(2T(n/4) + n/2) + n = 2^2T(n/2^2) + 2n = \ldots = 2^kT(n/2^k) + kn$
$n/2^k = 1 \implies k = \log_2 n$
$T(n) = 2^{\log_2 n}T(1) + n\log_2 n = n + n\log_2 n = O(n \log n)$

(d) $f(1) = 1; f(n) = \sum_1^{n-1}(i \cdot f(i))$
(Hint: compute $f(n+1) - f(n)$ for $n > 1$)

Solution:
$f(2) = \sum_1^1 (i \cdot f(i)) = 1 \cdot f(1) = 1$
$f(n+1) - f(n) = \sum_1^n (i \cdot f(i)) - \sum_1^{n-1}(i \cdot f(i)) = n \cdot f(n)$
$f(n+1) - f(n) = n \cdot f(n)$
$f(n+1) = (n+1) \cdot f(n)$
$f(n+1) = (n+1) \cdot n \cdot f(n-1) = (n+1) \cdot n \cdot (n-1) \cdot f(n-2) = \ldots = (n+1)!/2$
$f(n) = n!/2$

Coding Question: Hello World

Most assignments will have a coding question. You can code in C, C++, C#, Java, Python, or Rust. You will submit a Makefile and a source code file.

Makefile: In the Makefile, there needs to be a build command and a run command. Below is a sample Makefile for a C++ program. You will find this Makefile in assignment details. Download the sample Makefile and edit it for your chosen programming language and code.

```
#Build commands to copy:
#Replace g++ -o HelloWorld HelloWord.cpp below with the appropriate command.
#Java:
#       javac source_file.java
#Python:
#       echo "Nothing to compile."
#C#:
#       mcs -out:exec_name source_file.cs
#C:
#       gcc -o exec_name source_file.c
#C++:
#       g++ -o exec_name source_file.cpp
#Rust:
#       rustc source_file.rs

build:
        echo "Nothing to compile."

#Run commands to copy:
#Replace ./HelloWorld below with the appropriate command.
#Java:
#       java source_file
#Python 3:
#       python3 source_file.py
#C#:
#       mono exec_name
#C/C++:
#       ./exec_name
#Rust:
#       ./source_file

run:
        python3 source_file.py
```

18. **HelloWorld Program Details**

    The input will start with a positive integer, giving the number of instances that follow. For each instance, there will be a string. For each string s, the program should output Hello, s! on its own line.

    A sample input is the following:

```
3
World
Marc
Owen
```

    The output for the sample input should be the following:

```
Hello, World!
Hello, Marc!
Hello, Owen!
```

Asymptotic Analysis

19. *Kleinberg, Jon. Algorithm Design (p. 67, q. 3, 4).* Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

    (a) $f_1(n) = n^{2.5}$
    $f_2(n) = \sqrt{2n}$
    $f_3(n) = n + 10$
    $f_4(n) = 10n$
    $f_5(n) = 100n$
    $f_6(n) = n^2 \log n$

    Solution:

    $f_2(n), f_3(n), f_4(n), f_5(n), f_6(n), f_1(n)$

    (b) $g_1(n) = 2^{\log n}$
    $g_2(n) = 2^n$
    $g_3(n) = n(\log n)$
    $g_4(n) = n^{4/3}$
    $g_5(n) = n^{\log n}$
    $g_6(n) = 2^{(2^n)}$
    $g_7(n) = 2^{(n^2)}$

    Solution:

    $g_1(n), g_3(n), g_4(n), g_5(n), g_2(n), g_7(n), g_6(n)$

20. *Kleinberg, Jon. Algorithm Design (p. 68, q. 5).* Assume you have a positive, non-decreasing function f and a positive, non-decreasing function g such that $g(n) \geq 2$ and $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

(a) $\log_2 f(n)$ is $O(\log_2 g(n))$

Solution:

True.
$f(n)$ is $O(g(n))$
$f(n) \leq c \cdot g(n)$
$\log_2 f(n) \leq \log_2 c \cdot g(n)$
$\log_2 f(n) \leq \log_2 c + \log_2 g(n)$
$\log_2 f(n)$ is $O(\log_2 g(n))$

(b) $2^{f(n)}$ is $O(2^{g(n)})$

Solution:

True.
$f(n)$ is $O(g(n))$
$f(n) \leq c \cdot g(n)$
$2^{f(n)} \leq 2^{c \cdot g(n)}$
$2^{f(n)} \leq 2^c \cdot 2^{g(n)}$
$2^{f(n)}$ is $O(2^{g(n)})$

(c) $f(n)^2$ is $O(g(n)^2)$

Solution:

True.
$f(n)$ is $O(g(n))$
$f(n) \leq c \cdot g(n)$, $f(n) \geq 0$, $g(n) \geq 0$
$f(n)^2 \leq c^2 \cdot g(n)^2$
$f(n)^2$ is $O(g(n)^2)$

21. *Kleinberg, Jon. Algorithm Design (p. 68, q. 6).* You're given an array A consisting of n integers. You'd like to output a two-dimensional n-by-n array B in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$ that is, the sum $A[i] + A[i+1] + ... + A[j]$. (Whenever $i \geq j$, it doesn't matter what is output for $B[i, j]$.) Here's a simple algorithm to solve this problem.

```
for i = 1 to n
  for j = i + 1 to n
    add up array entries A[i] through A[j]
    store the result in B[i, j]
  endfor
endfor
```

(a) For some function f that you should choose, give a bound of the form $O(f(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm).

Solution: For each i, the inner loop runs $n - i$ times.

$T(n) = \sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i = n^2 - \frac{n(n+1)}{2} = \frac{n(n-1)}{2} = O(n^2)$

(b) For this same function f, show that the running time of the algorithm on an input of size n is also $\Omega(f(n))$. (This shows an asympto tically tight bound of $\Theta(f(n))$ on the running time.)

Solution:

For each i, the inner loop runs $n - i$ times.
$T(n) = \sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i = n^2 - \frac{n(n+1)}{2} = \frac{n(n-1)}{2} = \Omega(n^2)$

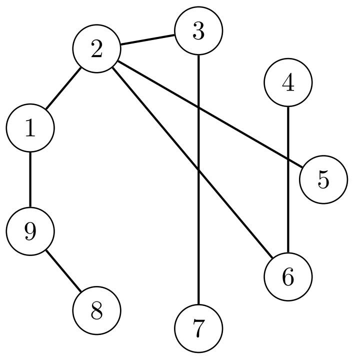
(c) Although the algorithm provided is the most natural way to solve the problem, it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$.

Solution:

Let S be an array of size n where $S[i] = \sum_{j=1}^{i} A[j]$
$S[1] = A[1]$
$S[i] = S[i-1] + A[i]$
$B[i, j] = S[j] - S[i-1]$
$T(n) = n + 1 = O(n)$
This algorithm is $O(n)$ and satisfies the requirements.

Graphs

22. Given the following graph, list a possible order of traversal of nodes by breadth-first search and by depth-first search. Consider node 1 to be the starting node.



Solution:

Breadth-First Search: in the groups, order doesn't matter
1, [2, 9], [3, 5, 6, 8] , [4, 7]

Depth-First Search: not an exhaustive list of solutions
Some possible solutions:
1, 9, 8, 2, 3, 7, 6, 4, 5
1, 2, 3, 7, 6, 4, 5, 9, 8
1, 2, 6, 4, 5, 3, 7, 9, 8

23. *Kleinberg, Jon. Algorithm Design (p. 108, q. 5).* A binary tree is a rooted tree in which each node has at most two children. Show by induction that in any binary tree the number of nodes with two children is exactly one less than the number of leaves.

Solution:

Base Case:
A tree with one node has no leaves and no nodes with two children.
Inductive Hypothesis:
Assume that for a tree with n nodes, the number of nodes with two children is exactly one less than the number of leaves.
Inductive Step:
Consider a tree with $n + 1$ nodes. Let us add a node to the tree with n nodes.
If we add to a leaf, then the number of leaves and nodes with two children both stay the same.
If we add to a node with one child, then the number of leaves and nodes with two children both increase by one.
Thus the number of nodes with two children is still one less than the number of leaves.

24. *Kleinberg, Jon. Algorithm Design (p. 108, q. 7).* Some friends of yours work on wireless networks, and they're currently studying the properties of a network of n mobile devices. As the devices move around, they define a graph at any point in time as follows:

> There is a node representing each of the n devices, and there is an edge between device i and device j if the physical locations of i and j are no more than 500 meters apart. (If so, we say that i and j are in range of each other.)

They'd like it to be the case that the network of devices is connected at all times, and so they've constrained the motion of the devices to satisfy the following property: at all times, each device i is within 500 meters of at least $\frac{n}{2}$ of the other devices. (We'll assume n is an even number.) What they'd like to know is: Does this property by itself guarantee that the network will remain connected?

Heres a concrete way to formulate the question as a claim about graphs.

Claim: Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least $\frac{n}{2}$, then G is connected.

Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

Solution:

Claim: Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least $\frac{n}{2}$, then G is connected.
Proof:
Assume that G is not connected.
Then there exists a node v such that there is no path from v to some other node u.
Since G is not connected, there must be at least one other node w such that there is no path from w to u.

Coding Question: DFS

25. Implement depth-first search in either C, C++, C#, Java, Python, or Rust. Given an undirected graph with n nodes and m edges, your code should run in $O(n + m)$ time. Remember to submit a makefile along with your code, just as with the first coding question.

Input: the first line contains an integer t, indicating the number of instances that follows. For each instance, the first line contains an integer n, indicating the number of nodes in the graph. Each of the following n lines contains several space-separated strings, where the first string s represents the name of a node, and the following strings represent the names of nodes that are adjacent to node s.

The input order of the nodes is important as it will be used as the tie-breaker. For example, consider two consecutive lines of an instance:

```
0, F
B, C, a
```

Note that the tie break priority would be 0 < F < B < C < a.

Input constraints:

- $1 \leq t \leq 1000$
- $1 \leq n \leq 100$
- Strings only contain alphanumeric characters
- Strings are guaranteed to be the names of the nodes in the graph.

Output: for each instance, print the names of nodes visited in depth-first traversal of the graph, *with ties between nodes visiting the first node in input order.* Start your traversal with the first node in input order. The names of nodes should be space-separated, and each line should be terminated by a newline.

Sample:

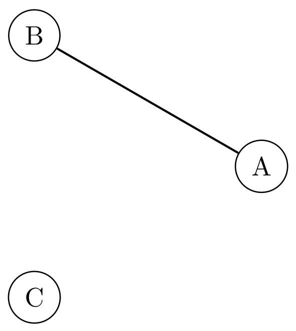
Input:

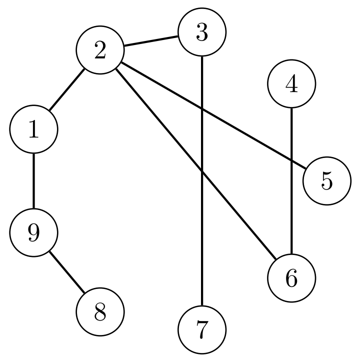
```
2
3
A B
B A
C
9
1 2 9
2 1 6 5 3
4 6
6 2 4
5 2
3 2 7
7 3
8 9
9 1 8
```

Output:

```
A B C
1 2 6 4 5 3 7 9 8
```

The sample input has two instances. The first instance corresponds to the graph below on the left. The second instance corresponds to the graph below on the right.


B
A
C


2
3
4
1
5
9
6
8
7