&

( )

Question 1 &

Input : length & width (sides)

*Output : length*width (area)*

Question 2 &

Input : a, b, c

Output : (a+b+c)/3

Question 3 &

Input : a, b, c

Output : Largest of a, b, c

Question 4 &

Input : n

Output: Even or Odd

%

- % ,

, %

,

Question 5 &

Input : none

Output: 12 15 18 21 24 27 30 33 36 39 42 45 48

, %

Variables & Data Types
(Assignment Questions)

Question 1 : In a program, input the side of a square. You have to output the area of the square.

Input : n (side)

*Output : n*n (area)*

Question 2 : Enter cost of 3 items from the user (using float data type) - a pencil, a pen and an eraser. You have to output the total cost of the items back to the user as their bill.

(Add on : You can also try adding 18% GST tax to the items in the bill as an advanced problem)

Question 3 : Build a Simple Interest Calculator.

Input : principal (P), rate (R), time (T)

*Output : (P*R*T) / 100*

Question 4 : Write a program to calculate the area of a circle.

Input : r (radius)

*Output : PI*r*r (area)*

Note - These questions are only to help you practice the concepts of this chapter. These are not designed to test your mathematical skills, just to understand logic building using C++.

Operators
(Assignment Questions)

Question 1 : What'll be the output of the following programs :

A.

```
int main() {
   int x = 2, y = 5;
   int exp1 = (x * y / x);
   int exp2 = (x * (y / x));
   cout << exp1 << ",";
   cout << exp2 << "\n";
}
```

B.

```
int main() {
   int x = 10, y = 5;
   int exp1 = (y * (x / y + x / y));
   int exp2 = (y * x / y + y * x / y);
   cout << exp1 << " ";
   cout << exp2 << "\n";
}
```

C.

```
int main() {
   int x = 200, y = 50, z = 100;
   if(x > y && y > z){
       cout << "Hello \n";
   }

   if(z > y && z < x){
       cout << "C++ \n";
   }

   if((y+200) < x && (y+150) < z){
       cout << "Hello C++ \n";
   }
}
```

Question 2 : Read up about **Operator Precedence**.

When multiple operators are used in a single statement, it is operator precedence which decides which operation is performed first & so on. (Similar to the rule of BODMAS used in math)

Note : Some of the operators mentioned in the table will be covered in later lectures.

( ) [ ]	Operators within parenthesis are performed first	Higher
++, --	Postfix increment / decrement	
++, --	Prefix increment / decrement	
*, /, %	Multiplication, Division, Modulus	
+, -	Addition, Subtraction	
<, <=, >, >=	Less than, Less than or equal to, Greater than, Greater than or equal to	
==, !=	Equal to, Not equal to	
&&	Logical AND	
||	Logical OR	
?:	Conditional Operator	
=	Simple Assignment	
+=, -=, *=, /=	Shorthand operators	
,	Comma operator	Lower

Conditional Statements
(Assignment Questions)

Question 1 : Write a C++ program to get a number from the user and print whether it's positive, negative or zero.

Question 2 : Write a C++ program that takes a year from the user and print whether that year is a leap year or not.

.

Hint *: A leap year is exactly divisible by 4 except for century years (years ending with 00). The century year is a leap year only if it is perfectly divisible by 400.*

Eg : 1999 is not a leap year
2000 is a leap year
2004 is a leap year

Question 3 : What will be the value of x & y in the following program:

```
int main() {
   int a = 63, b = 36;
   bool x = (a < b) ? true : false;
   int y = (a > b) ? a : b;
   cout << x << "," << y << endl;
   return 0;
}
```

Question 4 : What'll be the output of the program:

```
int main() {
   int a = 5;

   if (++a*5 <= 25) {
       cout<<"Hello\n";
   } else {
       cout<<"Bye\n";
   }
```

```
    return 0;
}
```

Question 5 : For any 3 digit number check whether it's an Armstrong number or not.
Armstrong number is a number that is equal to the sum of cubes of its digits.
Eg : 371 is an armstrong number.
3*3*3 + 7*7*7 + 1*1*1 = 371

Bonus : Read up about the difference between typedef (keyword), macros & const (keyword) in C++.

Loops
(Assignment Questions)

Question 1 : WAP to find the **Factorial** of a number entered by the user.

Hint *: factorial of a number (n) = n * (n-1) * (n-2) * (n-3) * ..... * 1*
and exists for positive numbers only. We write factorial as n!
So, factorial of 0! = 1, 1! = 1, 2! = 2, 3! = 6, 4! = 24 and so on.
Note - Please do not confuse factorial with NOT EQUAL TO operator, they are not the same.

Question 2 : WAP to print the multiplication table of a number, entered by the user.

Question 3 : WAP to input a number and check whether the number is an **Armstrong** number or not.

*An **Armstrong** number is a number that is equal to the sum of cubes of its digits.*

Question 4 : For a positive N , WAP that prints all the prime numbers from 2 to N.
(Assume N >= 2)

Question 5 : For a positive N , WAP that prints the first N **Fibonacci** numbers.
(Assume N >= 2)
Fibonacci *series : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ….*
This is a series where each number is a sum of previous 2 numbers in the series.
Eg : ***1*** *= 0 + 1,*
2 *= 1 + 1,*
3 *= 1 + 2,*
5 *= 2 + 3,*
8 *= 3 + 5 & so on.*

Patterns

(Assignment Questions)

Question 1 : Print the 0-1 Triangle Pattern.

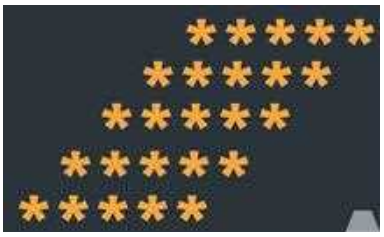
For n = 5

```
1
01
101
0101
10101
```


Question 2 : Print the Rhombus Pattern.

For n = 5

```
 *****
 *****
 *****
 *****
*****
```



Question 3 : Print the Palindromic Pattern with Numbers.

For n = 5

Functions
(Assignment Questions)

Question 1 : Write a function to check if a number is a palindrome in C++.
(121 is a palindrome, 321 is not)

A number is called a palindrome if the number is equal to the reverse of a number.
Eg : 121 is a palindrome because the reverse of 121 is 121 itself. On the other hand, 321 is not a palindrome because the reverse of 321 is 123, which is not equal to 321.

Question 2 : Write a function to calculate the sum of digits of a number.

Question 3 : Write a function which takes 2 numbers as parameters (a & b) and outputs : a^2 + b^2 + 2*ab.

Question 4 : Write a function that prints the largest of 3 numbers.

Question 5 : Write a function that accepts a character (ch) as parameters & returns the character that occurs after ch in the English alphabet.
Eg : input = 'c', return value = 'd'

Note : for ch = 'z', return 'a'.

Binary Number System
(Assignment Questions)

*Try to use the proper as well as the shortcut method of powers of 2, as taught in the lecture.

Question 1 : Convert the following binary numbers into decimal forms :

- 111111
- 10110
- 10011
- 110010

Question 2 : Convert the following decimal numbers into binary forms :

- 25
- 49
- 31
- 88

Question 3 : Following are the rules of adding 2 binary digits :

0 + 0 = 0, carry = 0
1 + 0 = 1, carry = 0
0 + 1 = 1, carry = 0
1 + 1 = 0, carry = 1

So, in math if 2 + 3 = 5, in binary it looks like

```
  1 0
+ 1 1
-----
1 0 1
```

Using this method, try to add these 2 numbers (63 & 22) in their binary form and verify that the binary output is equal to the decimal value 85.

Bonus : Try to read up about 3 Bitwise Operators in C++: OR (|), AND (&) and NOT (~). [Refer Link]

Introduction to Pointers
(Assignment Questions)

Question 1 : What will be the output of the following code :

```
int x;
int *ptr;
x = 7;
ptr = &x;
cout << *ptr;
```

Question 2 : What will be the output of the following code :

```
void multipleBy2(int &a, int &b, int &c) {
    a *= 2;
    b *= 2;
    c *= 2;
}

int main() {
    int x = 1, y = 2, z = 3;
    multipleBy2(x, y, z);
    cout << x << y << z << "\n";
    return 0;
}
```

Question 3 : What will be the output of the following code :

```
int a = 32;
int *ptr = &a;

char ch = 'A';
char &cho = ch;

cho += a;
*ptr += ch;
cout << a << ", " << ch << endl;
```

Bonus : Try to read up about Dangling Pointers.

Arrays

(Assignment Questions)

(EASY)

Question 1 : Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct. [link]

Examples :

Input: nums = [1,2,3,4]

Output: false

Input: nums = [1,1,1,3,3,4,3,2,4,2]

Output: true

(MEDIUM)

Question 2 : There is an integer array nums sorted in ascending order (with distinct values).

Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].

Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums.

You must write an algorithm with O(log n) runtime complexity. [link]

Examples :

Input: nums = [4,5,6,7,0,1,2], target = 0

Output: 4

Input: nums = [4,5,6,7,0,1,2], target = 3

Output: -1

(MEDIUM)

Question 3 : Given an integer array nums, find a subarray that has the largest product, and return the product. The test cases are generated so that the answer will fit in a 32-bit integer. [link]

Note - This Qs might feel difficult as a beginner because it uses DP approach.

Examples :
Input: nums = [2,3,-2,4]
Output: 6
Explanation: [2,3] has the largest product 6.

Input: intervals =nums = [-2,0,-1]
Output: 0
Explanation: The result cannot be 2, because [-2,-1] is not a subarray.

Basic Sorting Algorithms

(Assignment Questions)

Question 1 : Use the following sorting algorithms to sort an array in DESCENDING order :

a. Bubble Sort
b. Selection Sort
c. Insertion Sort
d. Counting Sort

You can use this array as an example : **[3, 6, 2, 1, 8, 7, 4, 5, 3, 1]**

2D Arrays
(Assignment Questions)

Question 1 : Print the number of all 7's that are in the 2d array.

Example :

Input - int arr[ ][ ] = { {4,7,8}, {8,8,7} }; n = 2, m = 3

Output - 2

Question 2 : Print out the sum of the numbers in the second row of the "nums" array.

Example :

Input - int nums[ ][ ] = { {1,4,9}, {11,4,3}, {2,2,3} };

Output - 18

Question 3 : Write a program to Find Transpose of a Matrix.

What is ***Transpose***?

Transpose of a matrix is the process of swapping the rows to columns. For a 2x3 matrix,

Matrix

a11 a12 a13

a21 a22 a23

Transposed Matrix

a11 a21

a12 a22

a13 a23

Extra Questions

Question 4 : Go to Qs

Question 5 : Go to Qs

Strings

(Assignment Questions)

Question 1 : Count how many times lowercase vowels occurred in a String entered by the user.

Question 2 : You are given two strings s1 and s2 of equal length. A string swap is an operation where you choose two indices in a string (not necessarily different) and swap the characters at these indices.

Return true if it is possible to make both strings equal by performing at most one string swap on exactly one of the strings. Otherwise, return false.

Example :

Input: s1 = "bank", s2 = "kanb"

Output: true

Explanation: For example, swap the first character with the last character of s2 to make "bank".

Extra Questions

Question 3 : Go to Qs

Question 4 : Go to Qs

Vectors

(Assignment Questions)

Question 1 : You have a set of integers, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another number in the set, which results in repetition of one number and loss of another number.

You are given an integer array nums representing the data status of this set after the error.

Find the number that occurs twice and the number that is missing and return them in the form of an array. [Go to Qs]

Example :

Input: nums = [1,2,2,4]

Output: [2,3]

Question 2 : You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).

Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store.

Notice that you may not slant the container. [Go to Qs]

Input: height = [1,8,6,2,5,4,8,3,7]

Output: 49

Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

Question 3 : Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

Notice that the solution set must not contain duplicate triplets. [Go to Qs]

Input: nums = [-1,0,1,2,-1,-4]
Output: [[-1,-1,2],[-1,0,1]]
Explanation:
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.
The distinct triplets are [-1,0,1] and [-1,-2].
Notice that the order of the output and the order of the triplets does not matter.

Input: nums = [0,1,1]
Output: [ ]
Explanation: The only possible triplet does not sum up to 0.

Note - Read up about the in-built sorting of vectors.

Bit Manipulation

(Assignment Questions)

Question 1 : Write a Function to clear the range of bits from i to j in a given number.
(i & j are counted backwards from the right end of the number)

Examples :

Input : num = 15, i = 1, j = 3
Output : 1

Explanation :
15 in binary form is => 00001111 [i=1st & j=3rd bit underlined]
After bits are cleared, number will become 00000001

Input : num = 31, i = 1, j = 3
Output : 17

Note - Think about what type of bit mask is needed. It can also be a combination of 2 numbers.

Question 2 : Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space. [Go to Qs]

Hint - Think XOR.

Question 3 : You are given a 0-indexed integer array nums.

The effective value of three indices i, j, and k is defined as ((nums[i] | nums[j]) & nums[k]).

The xor-beauty of the array is the XORing of the effective values of all the possible triplets of indices (i, j, k) where 0 <= i, j, k < n.

Return the xor-beauty of nums. [Go to Qs]

Extra Qs : Given two integers dividend and divisor, divide two integers without using multiplication, division, and mod operator. [Go to Qs]

Object Oriented Programming

(Assignment Questions)

Question 1 : Create a class to store Complex numbers. Using operator overloading, create the logic to subtract one complex number from another.

Note - In Complex numbers, the real part of 1st gets subtracted from the real part of 2nd number. Same goes for the imaginary part.

Question 2 : Create a class **BankAccount** with private attributes *accountNumber* and *balance*. Implement public methods *deposit*(), *withdraw*(), and *getBalance*() to manage the account.

Question 3 : Create a base class **Person** with attributes *name* and *age*.
Derive a class **Student** from **Person** and add an additional attribute *studentID*.
Implement a method *displayStudentInfo( )* in the **Student** class to display all details.

In main function Student class object will be created in this format:

Student student("Alice", 20, "S12345");

Note - When we initialize an object of a derived class, the base class part has to be constructed first.
If we don't initialize it ourselves in the derived class' constructor by calling one of its constructors, the compiler will attempt to use the default constructor of the base class.
To invoke the parent's parameterized constructor in Child's constructor, syntax is :

```
Child(int x) : Parent(x)
{ ..... }
```

Object Oriented Programming

(Practice MCQs)

1. Which of the following is not a feature of OOP in C++?

    a) Encapsulation

    b) Inheritance

    c) Polymorphism

    d) Compilation

2. What is encapsulation?

    a) Bundling data and methods that operate on the data into a single unit

    b) The ability to create a new class from an existing class

    c) The ability to redefine methods in derived classes

    d) None of the above

3. Which of the following access specifiers is not available in C++?

    a) Public

    b) Private

    c) Protected

    d) Friendly

4. Which keyword is used to define a base class in C++?

    a) base

    b) super

    c) class

    d) parent

5. Which type of inheritance is not supported directly by C++?

a) Single inheritance

b) Multiple inheritance

c) Multilevel inheritance

d) Hybrid inheritance

6. What is polymorphism in C++?

a) The ability of a function or operator to behave in different ways

b) The process of hiding data

c) The ability to create a new class from an existing class

d) None of the above

7. What is a virtual function in C++?

a) A function defined in a base class that can be overridden in a derived class

b) A function that exists in memory but is not used

c) A function that is called during object creation

d) None of the above

8. What is the output of the following code?

```
class Base {
public:
   void show() { cout << "Base" << endl; }
};

class Derived : public Base {
public:
   void show() { cout << "Derived" << endl; }
};

int main() {
```

```
    Base* b;
    Derived d;
    b = &d;
    b->show();
    return 0;
}
```

a) Base

b) Derived

c) Compilation error

d) Runtime error

9. What is the purpose of a constructor in C++?

a) To deallocate memory

b) To initialize objects

c) To create a new class

d) None of the above

10. Which of the following statements about destructors is true?

a) A class can have multiple destructors

b) Destructors are called manually by the programmer

c) Destructors are used to release resources

d) Destructors can be overloaded

11. What is the output of the following code?

```
class A {
public:
    A() { cout << "A"; }
    ~A() { cout << "~A"; }
};
```

```
int main() {
   A obj;
   return 0;
}
```

a) A

b) ~A

c) A~A

d) Compilation error

12. Which of the following is not a type of constructor in C++?

a) Default constructor

b) Parameterized constructor

c) Copy constructor

d) Virtual constructor

13. How is dynamic polymorphism achieved in C++?

a) Using overloaded functions

b) Using function overriding

c) Using function templates

d) Using default arguments

14. Which of the following can be declared as a friend in C++?

a) Function

b) Class

c) Another object

d) Both a and b

15. What is the output of the following code?

```
Class Base {
public:
   virtual void print() { cout << "Base"; }
};

class Derived : public Base {
public:
   void print() { cout << "Derived"; }
};

int main() {
   Base* b = new Derived();
   b->print();
   delete b;
   return 0;
}
```

a) Base

b) Derived

c) Compilation error

d) Runtime error

16. Which of the following is true about pure virtual functions?

a) They have no implementation in the base class

b) They must be implemented in the derived class

c) They are declared using the syntax = 0

d) All of the above

17. What is an abstract class in C++?

a) A class that cannot be instantiated

b) A class with at least one pure virtual function

c) A class with all its functions pure virtual

d) Both a and b

18. What is the use of the *this* pointer in C++?

a) To access the static members of the class

b) To differentiate between local and global variables

c) To access the object's members within the class methods

d) None of the above

19. What is the default access specifier for members of a class in C++?

a) Public

b) Private

c) Protected

d) None

20. Which of the following is correct about operator overloading in C++?

a) It allows defining new operators

b) It allows using operators with user-defined data types

c) It changes the syntax of the language

d) None of the above

21. What is the correct way to define a copy constructor?

```
Class A {
public:
```

```
    A(const A &obj) { /*...*/ }
};
```

a) A(const A obj) { /.../ }

b) A(A &obj) { /.../ }

c) A(A obj) { /.../ }

d) A(const A &obj) { /.../ }

22. Which of the following is a correct way to declare an array of objects in C++?

a) ClassName obj[5];

b) ClassName obj = new ClassName[5];

c) ClassName obj{5};

d) ClassName obj{};

23. Which of the following is true about inheritance in C++?

a) Derived class inherits private members of the base class

b) Derived class can access protected members of the base class

c) Derived class cannot override base class methods

d) None of the above

24. What does the *protected* access specifier mean?

a) Members are accessible only within the same class

b) Members are accessible within the same class and derived classes

c) Members are accessible within the same class and friend classes

d) Members are accessible from anywhere in the program

25. What is a virtual destructor in C++?

a) A destructor that does nothing

b) A destructor that can be called manually

c) A destructor that ensures derived class destructors are called

d) A destructor that can be overridden

26. What is the output of the following code?

```
class Base {
public:
   Base() { cout << "Base"; }
};

class Derived : public Base {
public:
   Derived() { cout << "Derived"; }
};

int main() {
   Derived obj;
   return 0;
}
```

a) Base

b) Derived

c) BaseDerived

d) DerivedBase

27. Which of the following is true about constructors and inheritance?

a) Base class constructor is called after derived class constructor

b) Derived class constructor is called after base class constructor

c) Constructors are not called in inheritance

d) Constructors are called in any order

28. How is operator overloading done in C++?

a) Using the *operator* keyword

b) Using function overloading

c) Using the *overload* keyword

d) Using inheritance

29. What does the *delete* operator do in C++?

a) Deletes an object from memory

b) Deletes a class

c) Deletes a function

d) Deletes an attribute

30. What is the output of the following code**?**

```
class A {
public:
   virtual void show() { cout << "A"; }
};

class B : public A {
public:
   void show() { cout << "B"; }
};

int main() {
   A* a = new B();
   a->show();
   return 0;
}
```

a) A

b) B

c) AB

d) Compilation error

Solutions:

1. d) Compilation
2. a) Bundling data and methods that operate on the data into a single unit
3. d) Friendly
4. c) class
5. d) Hybrid inheritance
6. a) The ability of a function or operator to behave in different ways
7. a) A function defined in a base class that can be overridden in a derived class
8. a) Base
9. b) To initialize objects
10. c) Destructors are used to release resources
11. c) A~A
12. d) Virtual constructor
13. b) Using function overriding
14. d) Both a and b
15. b) Derived
16. d) All of the above
17. d) Both a and b
18. c) To access the object's members within the class methods
19. b) Private
20. b) It allows using operators with user-defined data types
21. d) A(const A &obj) { /.../ }
22. a) ClassName obj[5];
23. b) Derived class can access protected members of the base class
24. b) Members are accessible within the same class and derived classes
25. c) A destructor that ensures derived class destructors are called
26. c) BaseDerived
27. b) Derived class constructor is called after base class constructor
28. a) Using the `operator` keyword
29. a) Deletes an object from memory
30. b) B

Recursion

(Assignment Questions)

Question 1 : Write a recursive function to perform **Binary Search.**

Input : arr[ ] = {1, 2, 3, 4, 5, 6, 7}, n = 7, key = 5

Output : 4 (index of key)

*Use the starting index & ending index logic used in rotated, sorted array Qs.

Question 2 : For a given integer array of size N. You have to find all the occurrences (indices) of a given element (Key) and print them.

Use a recursive function to solve this problem.

Sample Input : arr[ ] = {3, 2, 4, 5, 6, 2, 7, 2, 2}, key = 2

Sample Output : 1 5 7 8

Question 3 : We are given a string S, we need to find the count of all contiguous substrings starting and ending with the same character. [Leetcode Premium Qs]

Sample Input 1 : S = "abcab"

Sample Output 1 : 7

There are 15 substrings of "abcab" : a, ab, abc, abca, abcab, b, bc, bca, bcab, c, ca, cab, a, ab, b

Out of the above substrings, there are 7 substrings : a, abca, b, bcab, c, a and b. So, only 7 contiguous substrings start and end with the same character.

Sample Input 2 : S = "aba"

Sample Output 2 : 4

The substrings are a, b, a and aba

Question 4 : TOWER OF HANOI (Important!)

You have 3 towers and N disks of different sizes which can slide onto any tower. The puzzle starts with disks sorted in ascending order of size from top to bottom (i.e., each disk sits on top of an even larger one).

You have the following constraints:

(1) Only one disk can be moved at a time.
(2) A disk is slid off the top of one tower onto another tower.
(3) A disk cannot be placed on top of a smaller disk. Write a program to move the disks from the first tower to the
last using Stacks.

Let rod 1 = 'A', rod 2 = 'B', rod 3 = 'C'.
An example with 2 disks i.e. N=2:
Step 1 : Shift the first disk from 'A' to 'B'.

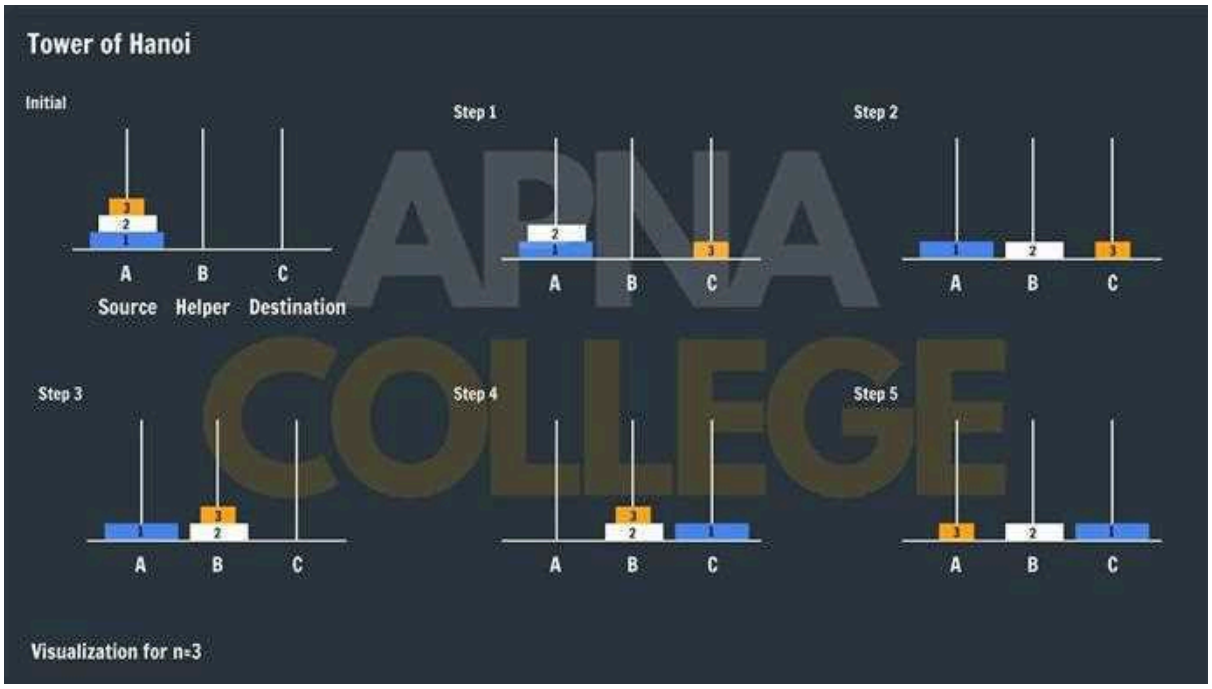
Step 2 : Shift the second disk from 'A' to 'C'.

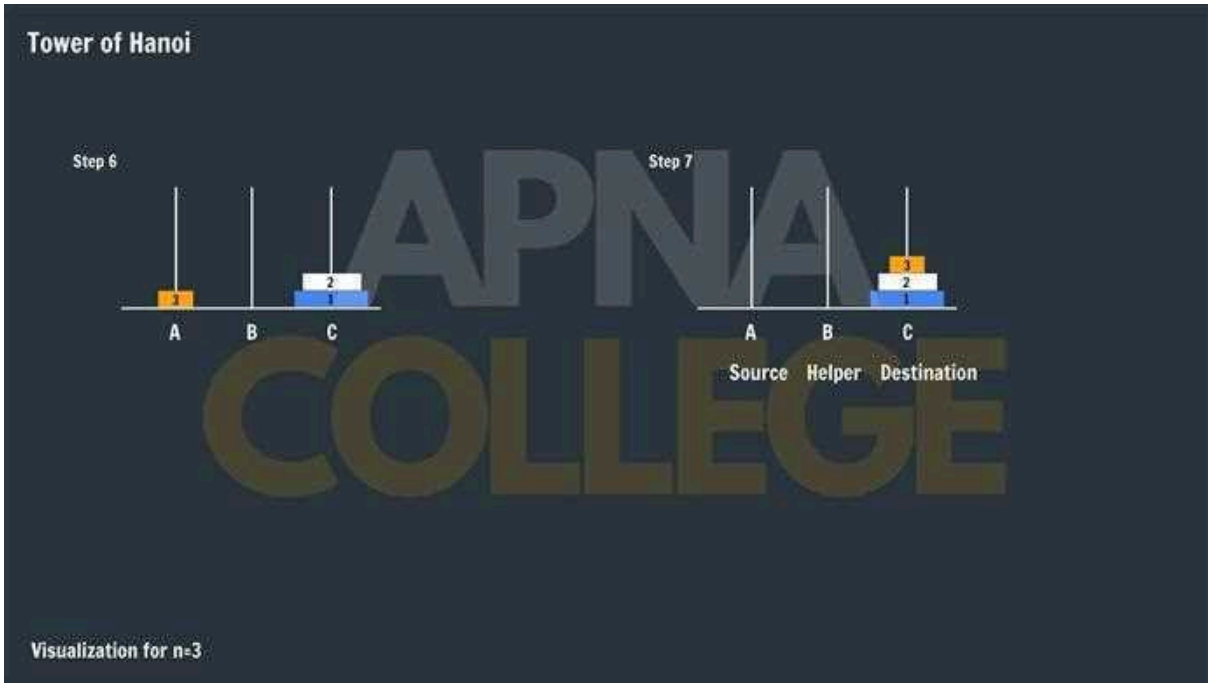
Step 3 : Shift the first disk from 'B' to 'C'.

An example with 3 disks i.e. N=3 :
Step 1 : Shift the first disk from 'A' to 'C'.
Step 2 : Shift second disk from 'A' to 'B'.
Step 3 : Shift the first disk from 'C' to 'B'.

Step 4 : Shift the third disk from 'A' to 'C'.

Step 5 : Shift the first disk from 'B' to 'A'.
Step 6 : Shift second disk from 'B' to 'C'.
Step 7 : Shift the first disk from 'A' to 'C'.





The Approach here is :

- Shift 'n-1' disks from 'A' to 'B', using C.
- Shift the last disk from 'A' to 'C'.
- Shift 'n-1' disks from 'B' to 'C', using A.

Question 5 : A digit string is good if the digits (0-indexed) at even indices are even and the digits at odd indices are prime (2, 3, 5, or 7).

For example, "2582" is good because the digits (2 and 8) at even positions are even and the digits (5 and 2) at odd positions are prime. However, "3245" is not good because 3 is at an even index but is not even.

Given an integer n, return the total number of good digit strings of length n. Since the answer may be large, return it modulo 109 + 7.

A digit string is a string consisting of digits 0 through 9 that may contain leading zeros.

[Go to Question]

Input: n = 1

Output: 5

Explanation: The good numbers of length 1 are "0", "2", "4", "6", "8".

Hint : This Qs uses the binary exponentiation technique we learnt in the Bit Manipulation chapter.

Divide & Conquer
(Assignment Questions)

Question 1 : Apply Merge sort to sort an array of Strings. (Assume that all the characters in all the Strings are in lowercase). (**EASY**)

Sample Input 1 : arr = { "sun", "earth", "mars", "mercury" }
Sample Output 1 : arr = { "earth", "mars", "mercury", "sun"}

Question 2 : Given an array nums of size n, return the majority element. (**MEDIUM**)

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Sample Input 1 : nums = [3,2,3]
Sample Output 1 : 3

Sample Input 2 : nums = [2,2,1,1,1,2,2]
Sample Output 2 : 2

Constraints (extra Conditions):

- n == nums.length
- 1 <= n <= 5 * 104
- -109 <= nums[i] <= 109

Question 3 : Given an array of integers. Find the Inversion Count in the array. (**HARD**)

Inversion Count: For an array, inversion count indicates how far (or close) the array is from being sorted. If the array is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the inversion count is the maximum.
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

Sample Input 1 : N = 5, arr[ ] = {2, 4, 1, 3, 5}
Sample Output 1 : 3, because it has 3 inversions - (2, 1), (4, 1), (4, 3).

Sample Input 2 : N = 5, arr[ ] = {2, 3, 4, 5, 6}
Sample Output 2 : 0, because the array is already sorted

Sample Input 3 : N = 3, arr[] = {5, 5, 5}
Sample Output 3 : 0, because all the elements of the array are the same & already in a sorted manner.

(**Hint** : A sorting algorithm will be used to solve this question.)
Note - This question is important. Even if you are not able to come up with the approach, please understand the solution.

Time & Space Complexity
(Assignment Questions)

Question : Find the Time Complexity of the following:

a)

```
int i, j, k = 0;
        for (i = n / 2; i <= n; i++) {
            for (j = 2; j <= n; j = j * 2) {
                k = k + n / 2;
            }
        }
```

A. O(n)
B. O(N log N)
C. O(n^2)
D. O(n^2Logn)

b)

```
for(int i=0;i<n;i++)
          i*=k
```

Here, k is some constant value

A. O(n)
B. O(k)
C. O(logkn) (= logn of base k)
D. O(lognk) (= logk of base n)

c)

Algorithm A and B have a worst-case running time of O(n) and O(logn), respectively. Therefore, algorithm B always runs faster than algorithm A.

A. True
B. False

d) Find the time & space complexity of floorSqrt function in the following code to calculate square root of a number :

```
int floorSqrt(int x)
   {
       if (x == 0 || x == 1)
           return x;

       int i = 1, result = 1;

       while (result <= x) {
           i++;
           result = i * i;
       }
       return i - 1;
   }

   int main()
   {
       int x = 11;
       cout << floorSqrt(x);
      return 0;
   }
}
```

e) Find the time & space complexity of the following code:

```
int a = 0;
for (int i = 0; i < n; ++i) {
   for (int j = n; j > i; --j) {
       a = a + i + j;
   }
}
```

Backtracking
(Assignment Questions)

Note - These are classical & important questions. Please positively solve them.

Question 1 :

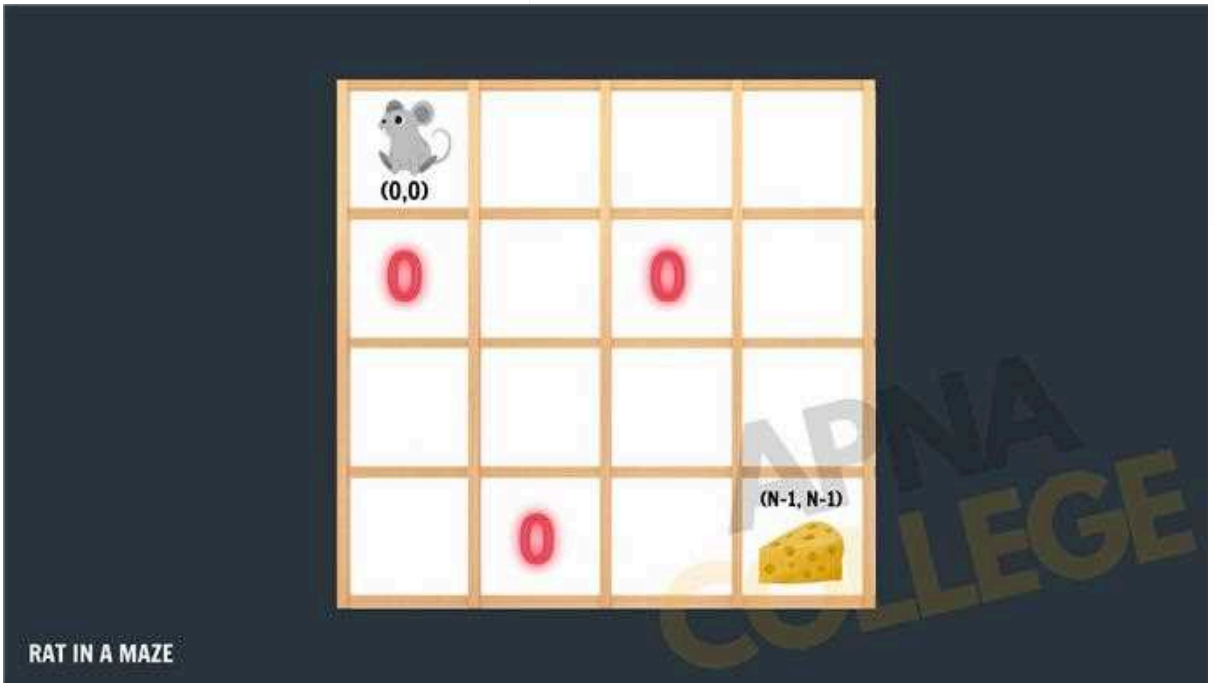
Rat in a Maze

You are given a starting position for a rat which is stuck in a maze at an initial point (0, 0) (the maze can be thought of as a 2-dimensional plane). The maze would be given in the form of a square matrix of order N * N where the cells with value 0 represent the maze's blocked locations while value 1 is the open/available path that the rat can take to reach its destination. The rat's destination is at (N - 1, N - 1).

Your task is to find all the possible paths that the rat can take to reach from source to destination in the maze.

The possible directions that it can take to move in the maze are 'U'(up) i.e. (x, y - 1) , 'D'(down) i.e. (x, y + 1) , 'L' (left) i.e. (x - 1, y), 'R' (right) i.e. (x + 1, y).

(This problem is similar to Grid ways.)



Sample Input : int maze[ ][ ] = { { 1, 0, 0, 0 },

{ 1, 1, 0, 1 },
{ 1, 1, 0, 0 },
{ 0, 1, 1, 1 } };

Sample Output : DDRDRR
DRDDRR

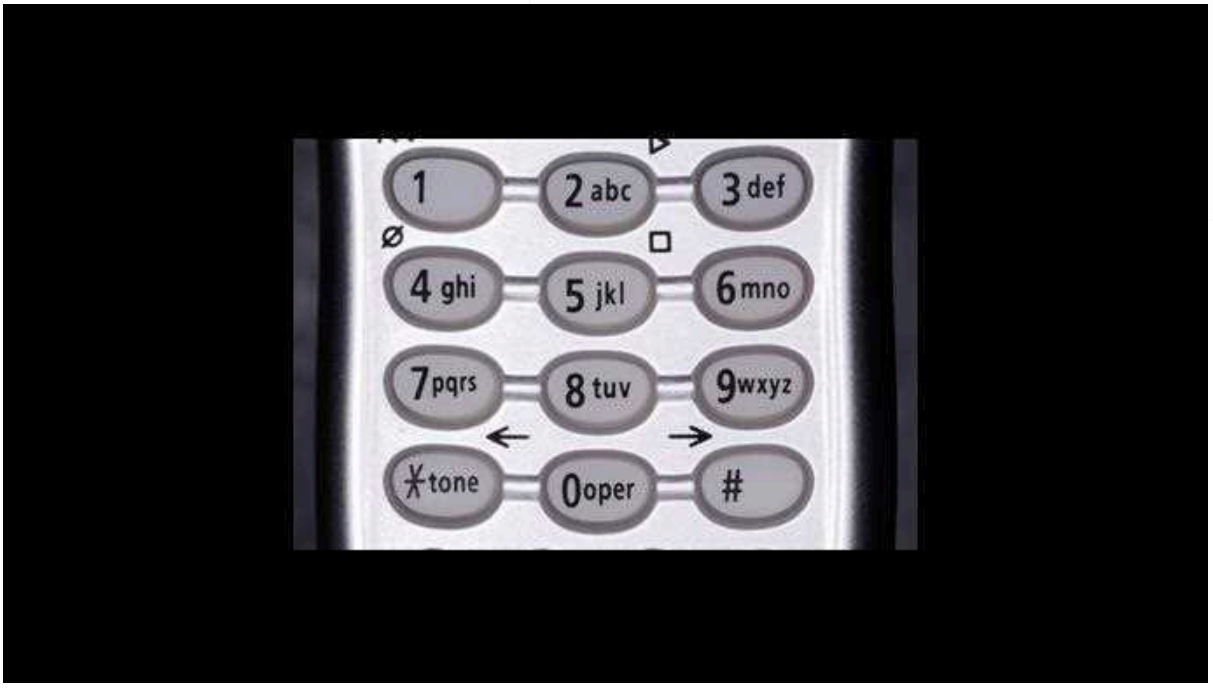
Hint : To track which cell has or not been visited, create a NxN vector called visited.
This vector will be initialized with false values for all cells & make the value for a particular cell to true when you have visited it.

Question 2 :

Keypad Combinations

Given a string containing digits from 2-9 inclusive, print all possible letter combinations that the number could represent. You can print the answer in any order.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



Sample Input 1 : digits = "23"
Sample Output 1 : "ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"

Sample Input 2 : digits = "2"
Sample Output 2 : "a", "b", "c"

Sample Input 3 : digits = ""
Sample Output 3 : ""

Question 3 :

Knight's Tour

Given a N*N board with the Knight placed on the first block of an empty board. Moving according to the rules of chess, knights must visit each square exactly once. Print the order of each cell in which they are visited.

Sample Input 1 : N = 8
Sample Output 1 :

```
0  59  38  33  30  17   8  63
37  34  31  60   9  62  29  16
58   1  36  39  32  27  18   7
35  48  41  26  61  10  15  28
42  57   2  49  40  23   6  19
47  50  45  54  25  20  11  14
56  43  52   3  22  13  24   5
51  46  55  44  53   4  21  12
```

(**Hint** : Similar to N Queens)

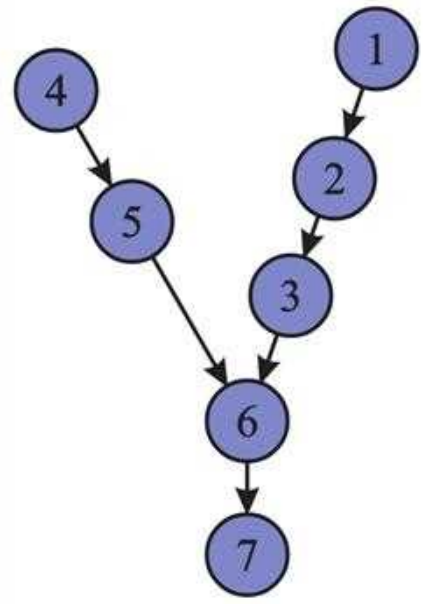
Linked List Questions

Note - These are classical questions. Please positively solve them.

Question 1 :

Intersection of Two Linked Lists

In a system there are two singly linked list. By some programming error, the end node of one of the linked lists got linked to the second list, forming an inverted Y-shaped list. Write a program to get the point where two linked lists merge.



`

We have to find the intersection part in this system.

Question 2 :

Delete N Nodes After M Nodes of a Linked List

We have a linked list and two integers M and N. Traverse the linked list such that you retain M nodes then delete next N nodes, continue the same till end of the linked list. Difficulty Level: Rookie.

Sample Input 1 : M=2 N=2 LL: 1->2->3->4->5->6->7->8
Sample Output 1 : 1->2->5->6

Sample Input 2 : M=3 N=2 LL: 1->2->3->4->5->6->7->8->9->10
Sample Output 2 : 1->2->3->6->7->8

Question 3 :

Swapping Nodes in a Linked List

We have a linked list and two keys in it, swap nodes for two given keys. Nodes should be swapped by changing links. Swapping data of nodes may be expensive in many situations when data contains many fields. It may be assumed that all keys in the linked list are distinct.

Sample Input 1 : 1->2->3->4, x = 2, y = 4
Sample Output 1 : 1->4->3->2

Question 4 :

Odd Even Linked List

We have a Linked List of integers, write a function to modify the linked list such that all even numbers appear before all the odd numbers in the modified linked list. Also, keep the order of even and odd numbers same.

Sample Input 1 : 8->12->10->5->4->1->6->NULL
Sample Output 1 : 8->12->10->4->6->5->1->NULL

Sample Input 2 : 1->3->5->7->NULL
Sample Output 2 : `1->3->5->7->NULL`

Question 5 :

Merge k Sorted Lists

We have K sorted linked lists of size N each, merge them and print the sorted output.

Sample Input 1 : `k = 2, n =  2`

```
l1 = 1->3->NULL
l2 = 6->8->NULL
l3 = 9->10->NULL
```

Sample Output 1 : 1>3->6->8->9->10->NULL

Stacks

(Assignment Questions)

Question 1 : Given the head of a singly linked list, return true if it is a Palindrome or false otherwise.[Go to Qs]

Examples :

Input: head = [1,2,2,1]
Output: true

Input: head = [1,2,3]
Output: false

Note - Use a stack to solve this problem.

Question 2 : Given an encoded string, return its decoded string.

The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there will not be input like 3a or 2[4]. [Go to Qs]

Examples :

Input: s = "3[a]2[bc]"
Output: "aaabcbc"

Input: s = "2[abc]3[cd]ef"
Output: "abcabccdcdcdef"

Question 3 : We have an absolute path for a file(Unix-style), simplify it. Note that absolute path always begins with '/'(rootdirectory), a dot in path represents current directory and double dot represents parent directory.[Go to Qs]

Examples :

Input: path = "/home//foo/"

Output: "/home/foo"

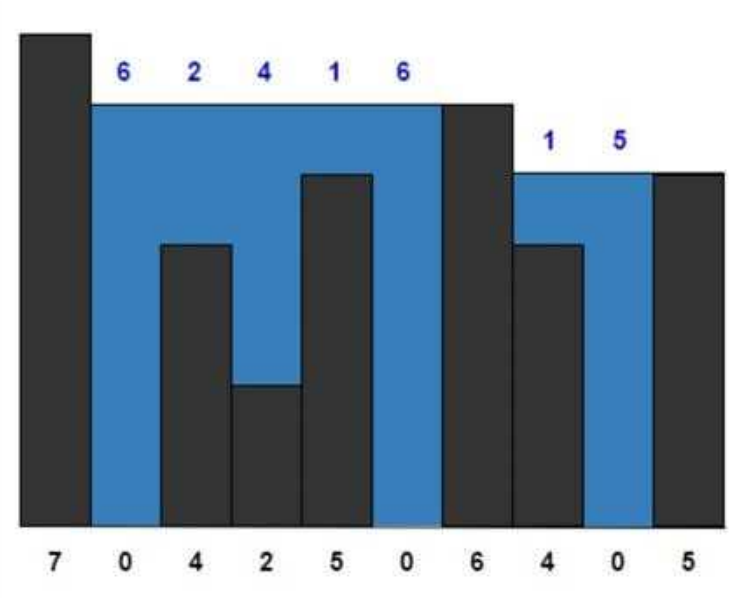
Explanation: Multiple consecutive slashes are replaced by a single one.

Input: path = "/home/user/Documents/../Pictures"

Output: "/home/user/Pictures"

Explanation: A double period ".." refers to the directory up a level.

Question 4 : Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.[Go to Qs]



Examples :

Input: height = [7, 0, 4, 2, 5, 0, 6, 4, 0, 5]

Output: 25

Explanation: The above elevation map (black section) is represented by the array [7, 0, 4, 2, 5, 0, 6, 4, 0, 5]. In this case, 25 units of rain water (blue section) are being trapped.

Queues
(Assignment Questions)

Question 1 : There are n people in a line queuing to buy tickets, where the 0th person is at the front of the line and the (n - 1)th person is at the back of the line.

You are given a 0-indexed integer array tickets of length n where the number of tickets that the ith person would like to buy is tickets[i].

Each person takes exactly 1 second to buy a ticket. A person can only buy 1 ticket at a time and has to go back to the end of the line (which happens instantaneously) in order to buy more tickets. If a person does not have any tickets left to buy, the person will leave the line.

Return the time taken for the person at position k (0-indexed) to finish buying tickets.[Go to Qs]

Examples :

Input: tickets = [2,3,2], k = 2

Output: 6

Explanation: In 1st pass, everyone in the line buys a ticket and the line becomes [1, 2, 1].

- In 2nd pass, everyone in the line buys a ticket and the line becomes [0, 1, 0].

The person at position 2 has successfully bought 2 tickets and it took 3 + 3 = 6 seconds.

Question 2 : There are n gas stations along a circular route, where the amount of gas at the ith station is gas[i].

You have a car with an unlimited gas tank and it costs => costs[i] of gas to travel from the ith station to its next (i + 1)th station. You begin the journey with an empty tank at one of the gas stations.

Given two integer arrays gas and cost, return the starting gas station's index if you can travel around the circuit once in the clockwise direction, otherwise return -1. If there exists a solution, it is guaranteed to be unique [Go to Qs]

Examples :

Input: gas = [1,2,3,4,5], cost = [3,4,5,1,2]

Output: 3

Explanation:

Start at station 3 (index 3) and fill up with 4 unit of gas. Your tank = 0 + 4 = 4

Travel to station 4. Your tank = 4 - 1 + 5 = 8

Travel to station 0. Your tank = 8 - 2 + 1 = 7

Travel to station 1. Your tank = 7 - 3 + 2 = 6

Travel to station 2. Your tank = 6 - 4 + 3 = 5

Travel to station 3. The cost is 5. Your gas is just enough to travel back to station 3.

Therefore, return 3 as the starting index.

Note : Use Deque to solve the question.

Question 3 : Given an integer K and a queue of integers, we need to reverse the order of the first K elements of the queue, leaving the other elements in the same relative order.

Only following standard operations are allowed on queue.

push(x) : Add an item x to rear of queue

pop() : Remove an item from front of queue

size() : Returns the number of elements in the queue.

front() : Finds front item.

Example :

Input : Queue is [1, 2, 3, 4, 5] & K = 3

Output: [3, 2, 1, 4, 5]

Explanation:

After reversing the given input from the 3rd position the output will be 3 2 1 4 5.

Bonus Question : Design a data structure that follows the constraints of a Least Recently Used (LRU) cache. Implement the LRUCache class:

LRUCache(int capacity) : Initialize the LRU cache with positive size capacity.
int get(int key) : Return the value of the key if the key exists, otherwise return -1.
void put(int key, int value) : Update the value of the key if the key exists.
Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity from this operation, evict the least recently used key.

The functions ***get*** and ***put*** must each run in O(1) average time complexity.[Go to Qs]

Note : This question uses an additional data structure, map, that we haven't covered yet. It will be covered in the later chapters. But you can read up about it & try to solve the Qs on your own.

Greedy Algorithms
(Assignment Questions)

Question 1 : ***Split a String in Balanced Strings***

Balanced strings are those that have an equal quantity of 'L' and 'R' characters.

Given a balanced string s, split it into some number of substrings such that: Each substring is balanced.

Return the maximum number of balanced strings you can obtain.[Go to Qs]

Examples :

Input: s = "RLRRLLRLRL"

Output: 4

Explanation: s can be split into "RL", "RRLL", "RL", "RL", each substring contains the same number of 'L' and 'R'.

Question 2 : ***Largest Odd Number in String***

You are given a string num, representing a large integer. Return the largest-valued odd integer (as a string) that is a non-empty substring of num, or an empty string "" if no odd integer exists.

A substring is a contiguous sequence of characters within a string. [Go to Qs]

Examples :

Input: num = "52"

Output: "5"

Explanation: The only non-empty substrings are "5", "2", and "52". "5" is the only odd number.

Question 3 : ***Smallest String With A Given Numeric Value***

The numeric value of a lowercase character is defined as its position (1-indexed) in the alphabet, so the numeric value of a is 1, the numeric value of b is 2, the numeric value of c is 3, and so on.

The numeric value of a string consisting of lowercase characters is defined as the sum of its characters' numeric values. For example, the numeric value of the string "abe" is equal to 1 + 2 + 5 = 8.

You are given two integers n and k. Return the lexicographically smallest string with length equal to n and numeric value equal to k.

Note that a string x is lexicographically smaller than string y if x comes before y in dictionary order, that is, either x is a prefix of y, or if i is the first position such that x[i] != y[i], then x[i] comes before y[i] in alphabetic order.[Go to Qs]

Example :

Input: n = 3, k = 27

Output: "aay"

Explanation: The numeric value of the string is 1 + 1 + 25 = 27, and it is the smallest string with such a value and length equal to 3.

Question 4 : ***Best Time to Buy and Sell Stock***

You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.[Go to Qs]

Example :

Input: prices = [7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Note : This question was already covered in class lectures of previous chapters.

Question 5 : ***Split Array Largest Sum***

Given an integer array nums and an integer k, split nums into k non-empty subarrays such that the largest sum of any subarray is minimized. Return the minimized largest sum of the split. (A subarray is a contiguous part of the array.) [Go to Qs]

Example :

Input: nums = [7,2,5,10,8], k = 2

Output: 18

Explanation: There are four ways to split nums into two subarrays.

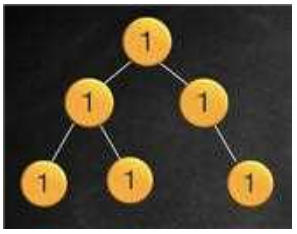
The best way is to split it into [7,2,5] and [10,8], where the largest sum among the two subarrays is only 18.

Binary Trees
(Assignment Questions)

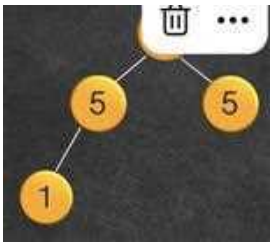
Question 1 : A binary tree is uni-valued if every node in the tree has the same value. Given the root of a binary tree, return true if the given tree is uni-valued, or false otherwise. [Go to Qs]

Examples :

Input: root Node of tree
Output: true



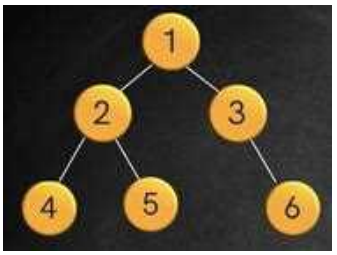
Input: root Node of tree
Output: false



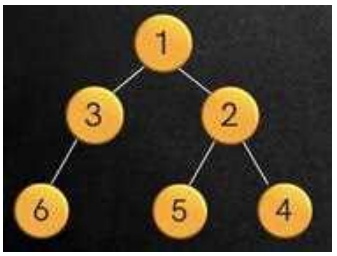
Question 2 : Given the root of a binary tree, invert the tree, and return its root.
[Go to Qs]

Examples :

Input: Binary Tree



Output: Inverted Tree

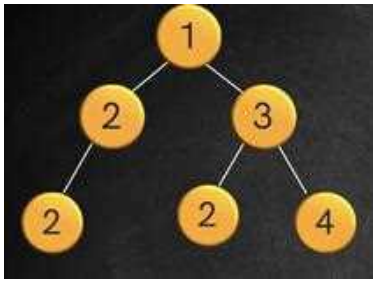


Question 3 : Given a binary tree root and an integer target, delete all the leaf nodes with value target.

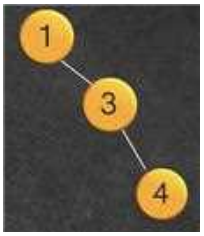
Note that once you delete a leaf node with value target, if its parent node becomes a leaf node and has the value target, it should also be deleted (you need to continue doing that until you cannot).[Go to Qs]

Examples :

Input: target = 2



Output:

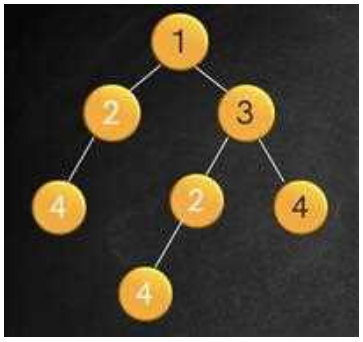


Question 4 : Given the root of a binary tree, return all duplicate subtrees.

For each kind of duplicate subtree, you only need to return the root node of any one of them. Two trees are duplicate if they have the same structure with the same node values. [Go to Qs]

Examples :

Input:



Output: [[2, 4], [4]]
First duplicate subtree with nodes [2, 4]
Second duplicate subtree with node [4]

Question 5 : A path in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence at most once. Note that the path does not need to pass through the root.

The path sum of a path is the sum of the node's values in the path.

Given the root of a binary tree, return the maximum path sum of any non-empty path.
[Go to Qs]

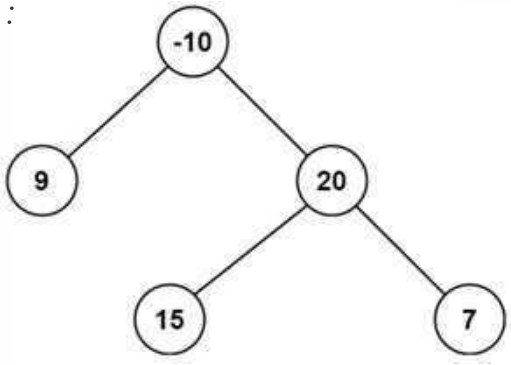
Examples :

Input:

```
  4
 / \
2   7
```

Output : 13

Input :



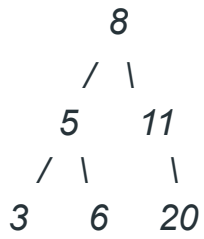
Output : 42

Binary Search Trees
(Assignment Questions)

Question 1 : Given the root node of a binary search tree and two integers low and high, return the sum of values of all nodes with a value in the inclusive range [low, high]. [Go to Qs] **(EASY)**

Question 2 : We have a binary search tree and a target node K. The task is to find the node with minimum absolute difference with given target value K.

Examples :



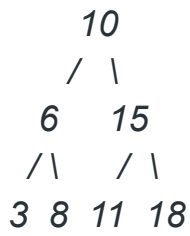
Input 1: K = 5
Output 1 : ans = 5 (abs diff = 0)

Input 2 : K = 19
Output 2: ans =20 (abs diff = 1)

Question 3 : Given the root of a binary search tree, and an integer k, return the kth smallest value (1-indexed) of all the values of the nodes in the tree.[Go to Qs] **(MEDIUM)**

Question 4 : Given two binary search trees, return True if and only if there is a node in the first tree and a node in the second tree whose values sum up to a given integer target.

Examples :



x = 16 (Target)

Output: ans = 3, The pairs are: (5, 11), (6, 10) and (8, 8)

Question 5 : Given a binary tree root, return the maximum sum of all keys of any sub-tree which is also a Binary Search Tree (BST).

Assume a BST is defined as follows:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- Both the left and right subtrees must also be binary search trees.

[Go to Qs] **(HARD)**

Self Practice Activity

Self Balancing BSTs : BSTs that automatically keep the height as small as possible when insertion and deletion operations are performed on the tree.

Eg : AVL Trees, Red Black Trees etc.

Read up about the following 2 types of self-balancing BSTs :

1. **AVL Trees**
2. **Red Black Trees**

Heaps Questions

Question 1 :

Last Stone Weight

You are given an array of integers stones where stones[i] is the weight of the ith stone.

We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with x <= y. The result of this smash is:

If x == y, both stones are destroyed, and
If x != y, the stone of weight x is destroyed, and the stone of weight y has new weight y - x.
At the end of the game, there is at most one stone left.

Return the weight of the last remaining stone. If there are no stones left, return 0. [Go to Qs]

Question 2 :

Kth Largest Element in Array

Given an integer array nums and an integer k, return the kth largest element in the array.
Note that it is the kth largest element in the sorted order, not the kth distinct element.
Can you solve it without sorting? [Go to Qs]

Question 3 :

Task Scheduler

You are given an array of CPU tasks, each represented by letters A to Z, and a cooling time, n. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: identical tasks must be separated by at least n intervals due to cooling time.

Return the minimum number of intervals required to complete all tasks. [Go to Qs]

Question 4 :

Design Twitter

Design a simplified version of Twitter where users can post tweets, follow/unfollow another user, and is able to see the 10 most recent tweets in the user's news feed.

Implement the Twitter class:

Twitter() Initializes your twitter object.

- ***void postTweet(int userId, int tweetId)*** : Composes a new tweet with ID tweetId by the user userId. Each call to this function will be made with a unique tweetId.
- ***List<Integer> getNewsFeed(int userId)*** : Retrieves the 10 most recent tweet IDs in the user's news feed. Each item in the news feed must be posted by users who the user followed or by the user themself. Tweets must be ordered from most recent to least recent.
- ***void follow(int followerId, int followeeId)*** : The user with ID followerId started following the user with ID followeeId.
- ***void unfollow(int followerId, int followeeId)*** : The user with ID followerId started unfollowing the user with ID followeeId.

[Go to Qs]

Question 5 :

Find Median from Data Stream

The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value, and the median is the mean of the two middle values.

For example, for arr = [2,3,4], the median is 3.

For example, for arr = [2,3], the median is (2 + 3) / 2 = 2.5.

Implement the MedianFinder class:

MedianFinder() initializes the MedianFinder object.

void addNum(int num) adds the integer num from the data stream to the data structure.

double findMedian() returns the median of all elements so far. Answers within 10-5 of the actual answer will be accepted. [Go to Qs]

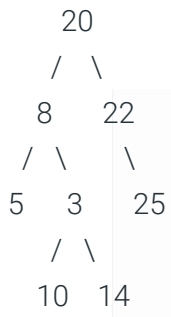
Hash Table Questions

Question 1 :

Bottom View of a Binary Tree

The top view of a binary tree is the set of nodes visible when the tree is viewed from the top. Given a binary tree, print the top view of it. The output nodes can be printed in any order.

Sample Input :



Sample Output : 5 10 3 14 25

Hint : Use the concept of Vertical Order

Question 2 :

Two Sum

Given an array of integers arr[ ]and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order. [Go to Qs]

Sample Input 1 : arr = [2, 7, 11, 15], target = 9

Sample Output 1 : [0, 1]

As arr[0] + arr[1] == 9, we return [0, 1].

Sample Input 2 : arr = [3,2,4], target = 6

Sample Output 2 : [1, 2]

Question 3 :

Sort by Frequency

Given a string s, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string.

Return the sorted string. If there are multiple answers, return any of them. [Go to Qs]

Sample Input 1 : s = "cccaaa"
Sample Output 1 : "aaaccc"
Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers.
Note that "cacaca" is incorrect, as the same characters must be together.

Sample Input 2 : s = "tree"
Sample Output 2 :"eert"
'e' appears twice while 'r' and 't' both appear once.
So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.

Question 4 :

Bulls & Cows

You are playing a game with your friend. You write down a secret number and ask your friend to guess what the number is. When your friend makes a guess, you provide a hint with the following info:

- The number of "bulls", which are digits in the guess that are in the correct position.
- The number of "cows", which are digits in the guess that are in your secret number but are located in the wrong position. Specifically, the non-bull digits in the guess that could be rearranged such that they become bulls.

Given the secret number secret and your friend's guess guess, return the hint for your friend's guess.

The hint should be formatted as "xAyB", where x is the number of bulls and y is the number of cows. Note that both secret and guess may contain duplicate digits. [Go to Qs]

Sample Input 1 : secret = "1807", guess = "7810"
Sample Output 1 : "1A3B"
Explanation: Bulls are highlighted with orange and cows are underlined:
"1**8**07", "7810"

Sample Input 2 : secret = "1123", guess = "0111"
Sample Output 2 : "1A1B"
Explanation: Bulls are highlighted with orange and cows are underlined:
"0**1**11", ("011" or "011")

TRIES QUESTIONS

Question 1 : MEDIUM

Group Anagrams Together

Given an array of strings strs, group the anagrams together. You can return the answer in any order.

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Sample Input 1 : strs = ["eat","tea","tan","ate","nat","bat"]
Sample Output 1 : [["bat"], ["nat", "tan"], ["ate", "eat", "tea"]]

Sample Input 2 : strs = [""]
Sample Output 2 : [[""]]

Sample Input 3 : strs = ["a"]
Sample Output 3 : [["a"]]

Question 2 : MEDIUM

Longest Word in Dictionary

Given an array of strings words representing an English Dictionary, return the longest word in words that can be built one character at a time by other words in words.

If there is more than one possible answer, return the longest word with the smallest lexicographical order. If there is no answer, return the empty string.

Note that the word should be built from left to right with each additional character being added to the end of a previous word.

Sample Input 1 : words = ["w","wo","wor","worl","world"]
Sample Output 1 : "world"
The word "world" can be built one character at a time by "w", "wo", "wor", and "worl".

Sample Input 2 : words = ["a","banana","app","appl","ap","apply","apple"]
Sample Output 2 : "apple"
Both "apply" and "apple" can be built from other words in the dictionary. However, "apple" is lexicographically smaller than "apply".

Graphs Questions

Question 1 : MEDIUM

Redundant Connection

You are given a graph that started as a tree with n nodes labeled from 1 to n, with one additional edge added. The added edge has two different vertices chosen from 1 to n, and was not an edge that already existed. The graph is represented as an array edges of length n where edges[i] = [ai, bi] indicates that there is an edge between nodes ai and bi in the graph.

Return an edge that can be removed so that the resulting graph is a tree of n nodes. If there are multiple answers, return the answer that occurs last in the input. [Go to Qs]

Question 2 : MEDIUM

Rotting Oranges

You are given an m x n grid where each cell can have one of three values:

- 0 representing an empty cell,
- 1 representing a fresh orange, or
- 2 representing a rotten orange.

Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.

Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1. [Go to Qs]

Question 3 : MEDIUM

Max Area of Island

You are given an m x n binary matrix grid. An island is a group of 1's (representing land) connected 4-directionally (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

The area of an island is the number of cells with a value 1 in the island.

Return the maximum area of an island in the grid. If there is no island, return 0. [Go to Qs]

Question 4 : MEDIUM

Word Ladder

A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

Every adjacent pair of words differs by a single letter.

Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.

sk == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return the number of words in the shortest transformation sequence from beginWord to endWord, or 0 if no such sequence exists. [Go to Qs]

Question 5 : HARD

Redundant Connection II

In this problem, a rooted tree is a directed graph such that, there is exactly one node (the root) for which all other nodes are descendants of this node, plus every node has exactly one parent, except for the root node which has no parents.

The given input is a directed graph that started as a rooted tree with n nodes (with distinct values from 1 to n), with one additional directed edge added. The added edge has two different vertices chosen from 1 to n, and was not an edge that already existed.

The resulting graph is given as a 2D-array of edges. Each element of edges is a pair [ui, vi] that represents a directed edge connecting nodes ui and vi, where ui is a parent of child vi.

Return an edge that can be removed so that the resulting graph is a rooted tree of n nodes. If there are multiple answers, return the answer that occurs last in the given 2D-array. [Go to Qs]

Question 6 : HARD

Couples Holding Hands

There are n couples sitting in 2n seats arranged in a row and want to hold hands.

The people and seats are represented by an integer array row where row[i] is the ID of the person sitting in the ith seat. The couples are numbered in order, the first couple being (0, 1), the second couple being (2, 3), and so on with the last couple being (2n - 2, 2n - 1).

Return the minimum number of swaps so that every couple is sitting side by side. A swap consists of choosing any two people, then they stand up and switch seats. [Go to Qs]

Question 7 : HARD

Course Schedule III

There are n different online courses numbered from 1 to n. You are given an array courses where courses[i] = [durationi, lastDayi] indicate that the ith course should be taken continuously for durationi days and must be finished before or on lastDayi.

You will start on the 1st day and you cannot take two or more courses simultaneously.

Return the maximum number of courses that you can take. [Go to Qs]

Question 8 : HARD

Alien Dictionary

There is a new alien language which uses the latin alphabet. However, the order among letters are unknown to you. You receive a list of non-empty words from the dictionary, where words are sorted lexicographically by the rules of this new language. Derive the order of letters in this language. [Go to Qs]

Input:

```
[
  "wrt",
  "wrf",
  "er",
  "ett",
  "rftt"
]
```

Output: "wertf"

Question 9 : MEDIUM

Number of Closed Islands

Given a 2D grid consists of 0s (land) and 1s (water). An island is a maximal 4-directionally connected group of 0s and a closed island is an island totally (all left, top, right, bottom) surrounded by 1s.

Return the number of closed islands [Go to Qs]

Question 10 : MEDIUM

Shortest Path with Alternating Colors

You are given an integer n, the number of nodes in a directed graph where the nodes are labeled from 0 to n - 1. Each edge is red or blue in this graph, and there could be self-edges and parallel edges.

You are given two arrays redEdges and blueEdges where:

redEdges[i] = [ai, bi] indicates that there is a directed red edge from node ai to node bi in the graph, and
blueEdges[j] = [uj, vj] indicates that there is a directed blue edge from node uj to node vj in the graph.
Return an array answer of length n, where each answer[x] is the length of the shortest path from node 0 to node x such that the edge colors alternate along the path, or -1 if such a path does not exist. [Go to Qs]

Catalan Numbers

Tom Davis
tomrdavis@earthlink.net
http://www.geometer.org/mathcircles
February 19, 2016

We begin with a set of problems that will be shown to be completely equivalent. The solution to each problem is the same sequence of numbers called the Catalan numbers. Later in the document we will derive relationships and explicit formulas for the Catalan numbers in many different ways.

1 Problems

1.1 Balanced Parentheses

Suppose you have n pairs of parentheses and you would like to form valid groupings of them, where "valid" means that each open parenthesis has a matching closed parenthesis. For example, "(()())" is valid, but "())()(" is not. How many groupings are there for each value of n?

Perhaps a more precise definition of the problem would be this: A string of parentheses is valid if there are an equal number of open and closed parentheses and if you begin at the left as you move to the right, add 1 each time you pass an open and subtract 1 each time you pass a closed parenthesis, then the sum is always non-negative.

Table 1 shows the possible groupings for $0 \le n \le 5$.

$n = 0$:	*	1 way
$n = 1$:	()	1 way
$n = 2$:	()(), (())	2 ways
$n = 3$:	()()(), ()(()), (())(), (()()), ((()))	5 ways
$n = 4$:	()()()(), ()()(()), ()(())(), ()(()()), ()((())), (())()(), (())(()), (()())(), ((()))(), (()()()), (()(())), ((()))(), ((()())), ((())()), ((()()))	14 ways
$n = 5$:	()()()()(), ()()()(()), ()()(())(), ()()(()()), ()()((())), ()(())()(), ()(())(()), ()(()())(), ()(((())))(), ()(()()()), ()(()(())), ()((()))(), ()((()())), ()((())()), ()((()())), (())()()(), (())()(()), (())(())(), (())(()()), (())((())), ((())()(), ((()))()(), ((()))(()), ((())())(), ((()()))(), (((())))(), (((()))()), (((())())), (((()()))), ((((()))), ((()())()), ((()(())), ((()(())), ((())()()), ((())(())), (((()))()), (((())())), (((()()))), ((((()))), (((()()))), ((((()))), ((((()))))	42 ways

Table 1: Balanced Parentheses

* It is useful and reasonable to define the count for $n = 0$ to be 1, since there is exactly one way of arranging zero parentheses: don't write anything. It will become clear later that this is exactly the right interpretation.

1.2 Mountain Ranges

How many "mountain ranges" can you form with n upstrokes and n downstrokes that all stay above the original line? If, as in the case above, we consider there to be a single mountain range with zero strokes, Table 2 gives a list of the possibilities for $0 \leq n \leq 3$:

$n = 0$:	*	1 way
$n = 1$:	/\	1 way
$n = 2$:	/\ /\/\, / \	2 ways
$n = 3$:	/\ /\ /\ /\/\ / \ /\/\/\, /\/ \, / \/\, / \, / \	5 ways

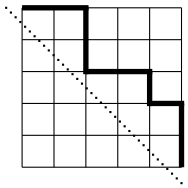
Table 2: Mountain Ranges

Note that these must match the parenthesis-groupings above. The "(" corresponds to "/" and the ") to "\". The mountain ranges for $n = 4$ and $n = 5$ have been omitted to save space, but there are 14 and 42 of them, respectively. It is a good exercise to draw the 14 versions with $n = 4$.

In our formal definition of a valid set of parentheses, we stated that if you add one for open parentheses and subtract one for closed parentheses that the sum would always remain non-negative. The mountain range interpretation is that the mountains will never go below the horizon.

1.3 Diagonal-Avoiding Paths

In a grid of $n \times n$ squares, how many paths are there of length $2n$ that lead from the upper left corner to the lower right corner that do not touch the diagonal dotted line from upper left to lower right? In other words, how many paths stay on or above the main diagonal?



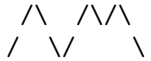


Figure 1: Corresponding Path and Range

This is obviously the same question as in the example above, with the mountain ranges running diagonally. In Figure 1 we can see how one such path corresponds to a mountain range.

Another equivalent statement for this problem is the following. Suppose two candidates for election, A and B, each receive n votes. The votes are drawn out of the voting urn one after the other. In how many ways can the votes be drawn such that candidate A is never behind candidate B?

1.4 Polygon Triangulation

If you count the number of ways to triangulate a regular polygon with $n + 2$ sides, you also obtain the Catalan numbers. Figure 2 illustrates the triangulations for polygons having 3, 4, 5 and 6 sides.

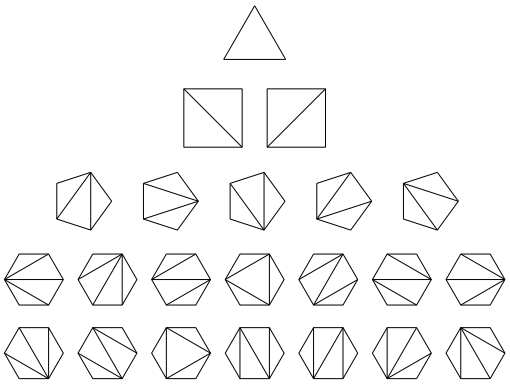


Figure 2: Polygon Triangulations

As you can see, there are 1, 2, 5, and 14 ways to do this. The "2-sided polygon" can also be triangulated in exactly 1 way, so the case where $n = 0$ also matches.

1.5 Hands Across a Table

If $2n$ people are seated around a circular table, in how many ways can all of them be simultaneously shaking hands with another person at the table in such a way that none of the arms cross each other? Figure 3 illustrates the arrangements for 2, 4, 6 and 8 people. Again, there are 1, 2, 5 and 14 ways to do this.

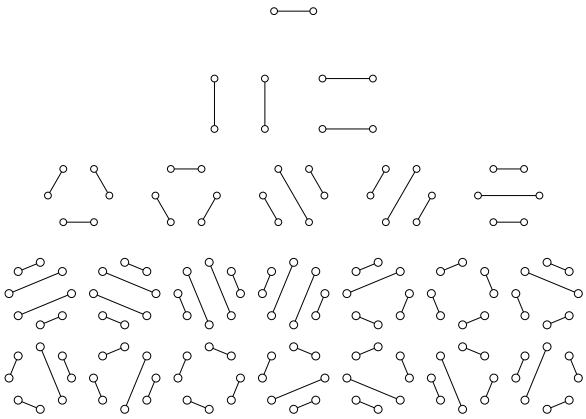


Figure 3: Hands Across the Table

1.6 Binary Trees

The Catalan numbers also count the number of rooted binary trees with n internal nodes. Illustrated in Figure 4 are the trees corresponding to $0 \leq n \leq 3$. There are $1, 1, 2$, and 5 of them. Try to draw the 14 trees with $n = 4$ internal nodes.

A rooted binary tree is an arrangement of points (nodes) and lines connecting them where there is a special node (the root) and as you descend from the root, there are either two lines going down or zero. Internal nodes are the ones that connect to two nodes below.

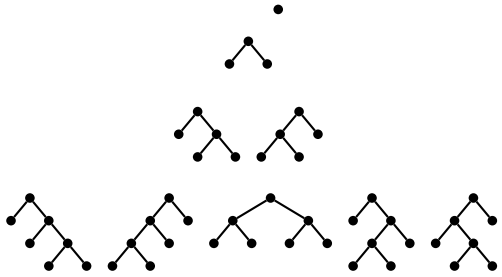


Figure 4: Binary Trees

1.7 Plane Rooted Trees

A plane rooted tree is just like the binary tree above, except that a node can have any number of sub-nodes; not just two.

Figure 5 shows a list of the plane rooted trees with n edges, for $0 \leq n \leq 3$. Try to draw the 14 trees with $n = 4$ edges.

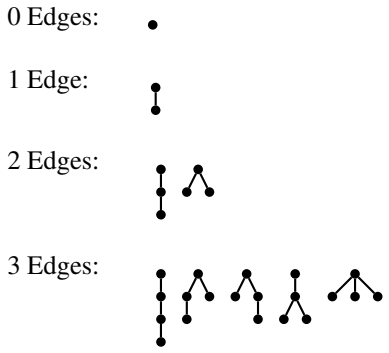


Figure 5: Plane Rooted Trees

1.8 Skew Polyominos

A polyomino is a set of squares connected by their edges. A skew polyomino is a polyomino such that every vertical and horizontal line hits a connected set of squares and such that the successive

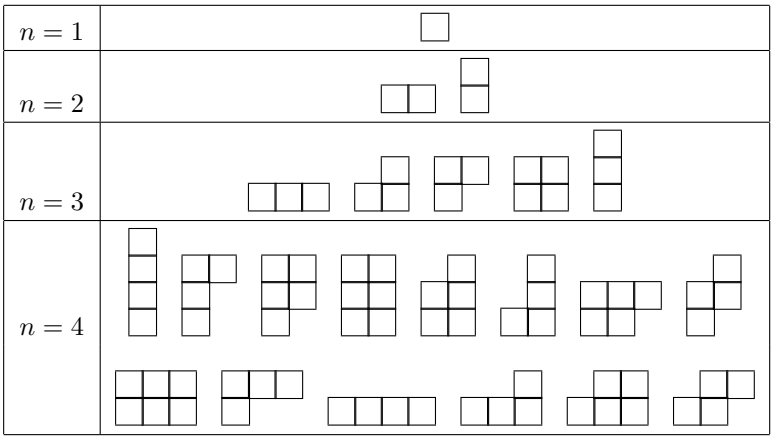


Table 3: Skew Polyominos with Perimeter $2n + 2$

columns of squares from left to right increase in height—the bottom of the column to the left is always lower or equal to the bottom of the column to the right. Similarly, the top of the column to the left is always lower than or equal to the top of the column to the right. Table 3 shows a set of such skew polyominos.

Another amazing result is that if you count the number of skew polyominos that have a perimeter of $2n + 2$, you will obtain C_n. Note that it is the perimeter that is fixed—not the number of squares in the polyomino.

1.9 Multiplication Orderings

Suppose you have a set of $n + 1$ numbers to multiply together, meaning that there are n multiplications to perform. Without changing the order of the numbers themselves, you can multiply the numbers together in many orders. Here are the possible multiplication orderings for $0 \leq n \leq 4$ multiplications. The groupings are indicated with parentheses and dot for multiplication in Table 4.

$n = 0$	(a)	1 way
$n = 1$	$(a{\cdot}b)$	1 way
$n = 2$	$((a{\cdot}b){\cdot}c),\ (a{\cdot}(b{\cdot}c))$	2 ways
$n = 3$	$(((a{\cdot}b){\cdot}c){\cdot}d),\ ((a{\cdot}b){\cdot}(c{\cdot}d)),\ ((a{\cdot}(b{\cdot}c)){\cdot}d),$ $(a{\cdot}((b{\cdot}c){\cdot}d)),\ (a{\cdot}(b{\cdot}(c{\cdot}d)))$	5 ways
$n = 4$	$((((a{\cdot}b){\cdot}c){\cdot}d){\cdot}e),\ (((a{\cdot}b){\cdot}c){\cdot}(d{\cdot}e)),\ (((a{\cdot}b){\cdot}(c{\cdot}d)){\cdot}e),$ $((a{\cdot}b){\cdot}((c{\cdot}d){\cdot}e)),\ ((a{\cdot}b){\cdot}(c{\cdot}(d{\cdot}e))),\ (((a{\cdot}(b{\cdot}c)){\cdot}d){\cdot}e),$ $((a{\cdot}(b{\cdot}c)){\cdot}(d{\cdot}e)),\ ((a{\cdot}((b{\cdot}c){\cdot}d)){\cdot}e),\ ((a{\cdot}(b{\cdot}(c{\cdot}d))){\cdot}e),$ $(a{\cdot}(((b{\cdot}c){\cdot}d){\cdot}e)),\ (a{\cdot}((b{\cdot}c){\cdot}(d{\cdot}e))),\ (a{\cdot}((b{\cdot}(c{\cdot}d)){\cdot}e)),$ $(a{\cdot}(b{\cdot}((c{\cdot}d){\cdot}e))),\ (a{\cdot}(b{\cdot}(c{\cdot}(d{\cdot}e))))$	14 ways

Table 4: Multiplication Arrangements

To convert the examples above to the parenthesis notation, erase everything but the dots and the

closed parentheses, and then replace the dots with open parentheses. For example, if we wish to convert $(a\cdot(((b\cdot c)\cdot d)\cdot e))$, first erase everything but the dots and closed parentheses: $\cdot\cdot)\cdot)\cdot))$. Then replace the dots with open parentheses to obtain: $(()()())$.

The examples in Table 4 are arranged in exactly the same order as the entries in Table 1 with the correspondence described in the previous paragraph. Try to convert a few yourself in both directions to make certain you understand the relationships.

2 A Recursive Definition

The examples above all seem to generate the same sequence of numbers. In fact it is obvious that some are equivalent: parentheses, mountain ranges and diagonal-avoiding paths, for example. Later on, we will prove that the other seqences are also the same. Once we're convinced that they are the same, we only need to have a formula that counts any one of them and the same formula will count them all.

If you have no idea how to begin with a counting problem like this, one good approach is to write down a formula that relates the count for a given n to previously-obtained counts. It is usually easy to count the configurations for $n = 0$, $n = 1$, and $n = 2$ directly, and from there, you can count more complex versions.

In this section, we'll use the example with balanced parentheses discussed and illustrated in Section 1.1. Let us assume that we already have the counts for $0, 1, 2, 3, \cdots, n-1$ pairs and we would like to obtain the count for n pairs. Let C_i be the number of configurations of i matching pairs of parentheses, so $C_0 = 1$, $C_1 = 1$, $C_2 = 2$, $C_3 = 5$, and $C_4 = 14$, which can be obtained by direct counts.

We know that in any balanced set, the first character has to be "(". We also know that somewhere in the set is the matching ")" for that opening one. In between that pair of parentheses is a balanced set of parentheses, and to the right of it is another balanced set:

$$(A)B,$$

where A is a balanced set of parentheses and so is B. Both A and B can contain up to $n-1$ pairs of parentheses, but if A contains k pairs, then B contains $n-k-1$ pairs. Notice that we will allow either A or B to contain zero pairs, and that there is exactly one way to do so: don't write down any parentheses.

Thus we can count all the configurations where A has 0 pairs and B has $n-1$ pairs, where A has 1 pair and B has $n-2$ pairs, and so on. Add them up, and we get the total number of configurations with n balanced pairs.

Here are the formulas. It is a good idea to try plugging in the numbers you know to make certain that you haven't made a silly error. In this case, the formula for C_3 indicates that it should be equal to $C_3 = 2\cdot 1 + 1\cdot 1 + 1\cdot 2 = 5$.

$$\begin{aligned}
C_1 &= C_0C_0 && (1)\\
C_2 &= C_1C_0 + C_0C_1 && (2)\\
C_3 &= C_2C_0 + C_1C_1 + C_0C_2 && (3)\\
C_4 &= C_3C_0 + C_2C_1 + C_1C_2 + C_0C_3 && (4)\\
\cdots &\qquad\qquad \cdots
\end{aligned}$$

$$C_n \quad = \quad C_{n-1}C_0 + C_{n-2}C_1 + \cdots + C_1C_{n-2} + C_0C_{n-1} \tag{5}$$

Beginning in the next section, we will be able to use these recursive formulas to show that the counts of other configurations (triangulations of polygons, rooted binary trees, rooted tress, et cetera) satisfy the same formulas and thus must generate the same sequence of numbers.

But simply by using the formulas above and a bit of arithmetic, it is easy to obtain the first few Catalan numbers: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, ...

2.1 Counting Polygon Triangulations

It is not hard to see that the polygon triangulations discussed in section 1.4 can be counted in much the same way as the balanced parentheses. See Figure 6.

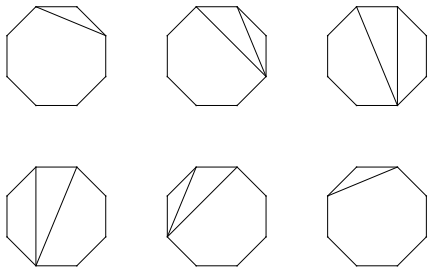


Figure 6: Octagon Triangulations

In the figure we consider the octagon, but it should be clear that the same argument applies to any convex polygon. Consider the horizontal line at the top of the polygon. After triangulation, it will be part of exactly one triangle, and in this case, there are exactly six possible triangles of which it can be a part. In each case, once that triangle is selected, there is a polygon (possibly empty) on the right and the left of the original triangle that must itself be triangulated.

What we would like to show is that a convex polygon with $n > 3$ sides can be triangulated in C_{n-2} ways. Thus the octagon should have $C_{8-2} = C_6$ triangulations.

For the example in the upper left of Figure 6, the triangle leaves a 7-sided figure on the left and an empty figure (essentially a two-sided polygon) on the right. This triangulation can be completed by triangulating both sides; the one on the left can be done in C_5 ways and the empty one on the right, C_0 ways, for a total of $C_5 \cdot C_0$. The middle example on the top leaves a pentagon and a triangle that, in total, can be trianguated in $C_4 \cdot C_1$ ways. Similar arguments can be made for all six positions of the triangle containing the top line, so we conclude that:

$$C_6 = C_5 \cdot C_0 + C_4 \cdot C_1 + C_3 \cdot C_2 + C_2 \cdot C_3 + C_1 \cdot C_4 + C_0 \cdot C_5,$$

which is exactly how the Catalan numbers are defined for the nested parentheses.

Convince yourself that a similar argument can be made for any size original convex polygon.

2.2 Counting Non-Crossing Handshakes

To count the number of hand-shakes discussed in Section 1.5 we can use an analysis similar to that used in section 2.1.

If there are $2n$ people at the table pick any particular person, and that person will shake hands with somebody. To admit a legal pattern, that person will have to leave an even number of people on each side of the person with whom he shakes hands. Of the remaining $n-1$ pairs of people, he can leave zero on the right and $n-1$ pairs on the left, 1 on the right and $n-2$ on the left, and so on. The pairs left on the right and left can independently choose any of the possible non-crossing handshake patterns, so again, the count C_n for n pairs of people is given by:

$$C_n = C_{n-1}C_0 + C_{n-2}C_1 + \cdots + C_1C_{n-2} + C_0C_{n-1},$$

which, together with the fact that $C_0 = C_1 = 1$, is just the definition of the Catalan numbers.

2.3 Counting Trees

Counting the binary trees discussed in Section 1.6 is similar to what we've done previously. Obviously there is one way to make a rooted binary tree with zero or one internal node. To work out the number of trees with n internal node, note that one of those n nodes is the root node, and then the $n-1$ additional internal nodes must be distributed on the left or the right below the root node. These can be distributed as 0 on the left and $n-1$ on the right, 1 on the left and $n-2$ on the right, and so on, yielding exactly the same formula that we had in every previous example.

To count the rooted plane trees discussed in Section 1.7 we use the same strategy. There is one example each for trees with zero and one edge, so the counts here are the same: $C_0 = C_1 = 1$.

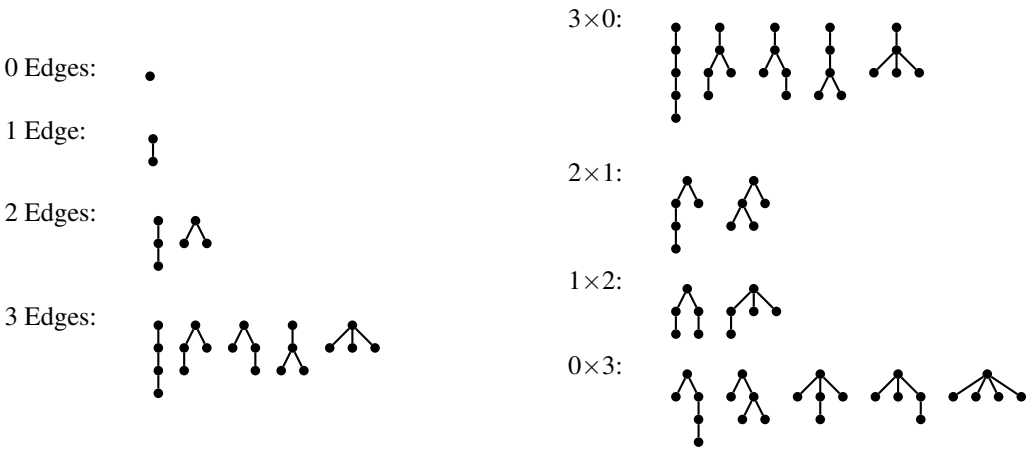


Figure 7: Plane Rooted Trees With 4 Edges

Now, to count the number of plane rooted trees with $n > 1$ edges we again begin from the root. There is at least one edge going down (leaving us with $n-1$ edges to draw). The remaining $n-1$ edges can be placed below that initial edge or hooked directly to the root node to the right of that edge. The $n-1$ edges, as before, can be distributed to these two locations as 0 and $n-1$, as 1 and

$n-2$, et cetera. It should be clear that the same formula defining the Catalan numbers will apply to the count of rooted plane trees.

In Figure 7 the table on the left dupliates the structure of trees with 3 or fewer edges and the table on the right shows how the trees with 4 edges are generated from them.

2.4 Counting Diagonal-Avoiding Paths

Up to now we do not have an explicit formula for the Catalan numbers. We know that a large collection of problems all have the same answers, and we have a recursive formula for those numbers, but it would be nice to have an explicit form.

Perhaps the easiest way to obtain an explicit formula for the Catalan numbers is to analyze the number of diagonal-avoiding paths discussed in Section 1.3. We will do so by counting the total number of paths through the grid and then subtract off the number of paths that hit the diagonal.

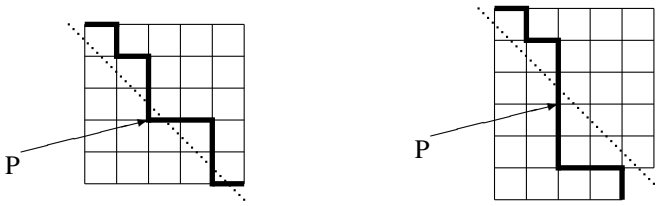


Figure 8: Modifying a Bad Path

Figure 8 illustrates a typical path that we do not want to count since it crosses the dotted diagonal line. Such a path may cross that line multiple times, but there is always a first time; in the figure, point P is the first grid point it touches on the wrong side of the diagonal. There will always be such a point P for every bad path.

For every such path, reflect the path beginning at P—every time the original path goes to the right, go down instead, and when the original path goes down, go to the right. It is clear that by the time the path reaches the point P it will have traveled one more step down than across, so it will have moved k steps to the right and $k+1$ steps down. The total path has n steps across and down, so there remain $n-k$ steps to the right and $n-k-1$ steps down. But since we swap steps to the right and steps down, the modified path with have a total of $(k)+(n-k-1)=n-1$ steps to the right and $(k+1)+(n-k)=n+1$ steps down. Thus every modified path ends at the same point, $n-1$ steps to the right and $n+1$ steps down.

Every bad path can be modified this way, and every path from the original starting point to this point $n-1$ to the right and $n+1$ down corresponds to exactly one bad path. Thus the number of bad paths is the total number of routes in a grid that is $(n-1)$ by $(n+1)$.

There are $\binom{m+k}{m}$ paths through an $k \times m$ grid[1]. Thus the total number of paths through the $n \times n$ grid is $\binom{2n}{n}$ and the total number of bad paths is $\binom{2n}{n+1}$. Thus C_n, the n^{th} Catalan number, or the total number of diagonal-avoiding paths through an $n \times n$ grid, is given by:

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

[1]To see this, remember that there are m steps down that need to be taken along the $k+1$ possible paths going down. Thus the problem reduces to counting the number of ways of putting m objects in $k+1$ boxes which is $\binom{m+k}{m}$.

3 Counting Mountain Ranges—Method 1

A very similar argument can be made as in the previous section if we use the interpretation of the Catalan numbers based on the count of mountain ranges as described in Section 1.2. In that section, we are seeking arrangements of n up-strokes and n down-strokes that form valid mountain ranges.

If we completely ignore whether the path is valid or not, we have n up-strokes that we can choose from a collection of $2n$ available slots. In other words, ignoring path validity, we are simply asking how many ways you can rearrange a collection of n up-strokes and n down-strokes. The answer is clearly $\binom{2n}{n}$.

Now we have to subtract off the bad paths. Every bad path goes below the horizon for the first time at some point, so from that point on, reverse all the strokes—replace up-strokes with down-strokes and vice-versa. It is clear that the new paths will all wind up 2 steps below the horizon, since they consist of $n+1$ down-strokes and $n-1$ up-strokes. Conversely, every path that ends two steps below the horizon must be of this form, so it corresponds to exactly one bad path.

How many such bad paths are there? The same number as there are ways to choose the $n+1$ up-strokes from among the $2n$ total strokes, or $\binom{2n}{n+1}$.

Thus the count of valid mountain ranges, or C_n, is given by exactly the same formula:

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

4 Counting Mountain Ranges—Method 2

Here is a different way to analyze the mountain problem. This time, imagine that we begin with $n+1$ up-strokes and only n down-strokes—we add an extra up-stroke to our collection.

First we solve the problem: How any arrangements can be made of these $2n+1$ symbols, without worrying about whether they form a "valid" mountain range (whatever that means with an unbalanced number of up-strokes and down-strokes). Clearly, if the ordering does not matter, there are $\binom{2n+1}{n}$ ways to do this.

One thing is certain, however. No matter how they are arranged, they mountain range will be one unit higher at the end, since we take $n+1$ steps up and only n steps down.

Let's look at a specific example with $n = 3$ (and $2n+1 = 7$): **up up down up up down down**. In Figure 4, we have arranged this sequence over and over and you can see that every 7 steps, the mountain range is one unit higher.

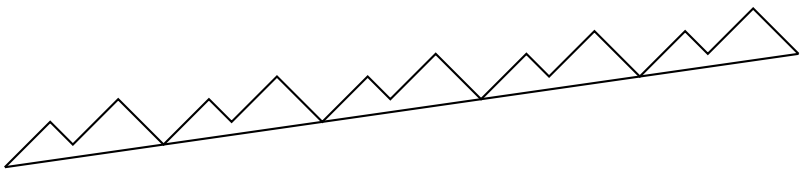


Figure 9: Growing Mountains

Since it is a repeating pattern, it's clear that we can draw a straight line below it that touches the bottom-most points of the growing mountain range.

In our example, this touching line seems to hit only once per complete set of 7 strokes, and we will show that this will always be the case, for any unbalanced number of up-strokes and down-strokes.

We can draw our mountain range on a grid, and it's clear that the slope of the line is $1/(2n+1)$ (it goes up 1 unit in every complete cycle of the pattern of $2n+1$ strokes. But lines with slope $1/(2n+1)$ can only hit lattice points every $2n+1$ units, so there is exactly one touching in each complete cycle.

If you have a series of $2n+1$ strokes, you can cycle that around to $2n+1$ arrangements. For example, the arrangement $//\backslash/\backslash$ can be cycled to four other arrangements: $/\backslash/\backslash/$, $\backslash/\backslash//$, $/\backslash//\backslash$ and $\backslash//\backslash/$. That means the complete set of arrangements can be divided into equivalence classes of size $2n+1$, where two arrangements are equivalent if they are cycled versions of each other.

If we consider the version among these $2n+1$ cycles, the only one that yields a valid mountain range is the one that begins at the low point of the $2n+1$ arrangement. Thus, to get a count of valid mountain ranges with n up-strokes and n down-strokes, we need to divide our count of $2n+1$ stroke arrangements by $2n+1$:

$$C_n = \frac{1}{2n+1}\binom{2n+1}{n} = \frac{1}{2n+1}\cdot\frac{(2n+1)!}{n!(n+1)!} = \frac{1}{n+1}\cdot\frac{(2n)!}{n!\,n!} = \frac{1}{n+1}\binom{2n}{n}.$$

Finally, note that when the line is drawn that touches the bottom edge of the range of mountains with one more "up" than "down", the first steps after the touching points are two "ups", since an "up-down" would immediately dip below the line. It should be clear that if one of the two initial "up" moves is removed, the resulting series will stay above a horizontal line.

5 Generating Function Solution

Using the formulas 1 through 5 in Section 2, we can obtain an explicit formula for the Catalan numbers, C_n using the technique known as generating functions.

We begin by defining a function $f(z)$ that contains all of the Catalan numbers:

$$f(z) = C_0 + C_1 z + C_2 z^2 + C_3 z^3 + \cdots = \sum_{i=0}^{\infty} C_i z^i.$$

If we multiply $f(z)$ by itself to obtain $[f(z)]^2$, the first few terms look like this:

$$[f(z)]^2 = C_0C_0 + (C_1C_0 + C_0C_1)z + (C_2C_0 + C_1C_1 + C_0C_2)z^2 + \cdots.$$

The coefficients for the powers of z are the same as those for the Catalan numbers obtained in equations 1 through 5:

$$[f(z)]^2 = C_1 + C_2 z + C_3 z^2 + C_4 z^3 + \cdots. \tag{6}$$

We can convert Equation 6 back to $f(z)$ if we multiply it by z and add C_0, so we obtain:

$$f(z) = C_0 + z[f(z)]^2. \tag{7}$$

Equation 7 is just a quadratic equation in $f(z)$ which we can solve using the quadratic formula. In a more familiar form, we can rewrite it as: $zf^2 - f + C_0 = 0$. This is the same as the quadratic

equation: $af^2 + bf + c = 0$, where $a = z$, $b = -1$, and $c = C_0$. Plug into the quadratic formula and we obtain:

$$f(z) = \frac{1 - \sqrt{1 - 4z}}{2z}. \tag{8}$$

Notice that we have used the $-$ sign in place of the usual $\pm$ sign in the quadratic formula. We know that $f(0) = C_0 = 1$, so if we replaced the $\pm$ symbol with $+$, as $z \to 0$, $f(z) \to \infty$.

To expand $f(z)$ we will just use the binomial formula on

$$\sqrt{1 - 4z} = (1 - 4z)^{1/2}.$$

If you are not familiar with the use of the binomial formula with fractional exponents, don't worry—it is exactly the same, except that it never terminates.

Let's look at the binomial formula for an integer exponent and just do the same calculation for a fraction. If n is an integer, the binomial formula gives:

$$(a + b)^n = a^n + \frac{n}{1}a^{n-1}b^1 + \frac{n(n-1)}{2 \cdot 1}a^{n-2}b^2 + \frac{n(n-1)(n-2)}{3 \cdot 2 \cdot 1}a^{n-3}b^3 + \cdots.$$

If n is an integer, eventually the numerator is going to have a term of the form $(n - n)$, so that term and all those beyond it will be zero. If n is not an integer, and it is $1/2$ in our example, the numerators will pass zero and continue. Here are the first few terms of the expansion of $(1-4z)^{1/2}$:

$$\begin{aligned}(1 - 4z)^{1/2} \quad = \quad & 1 - \frac{\left(\frac{1}{2}\right)}{1}4z + \frac{\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)}{2 \cdot 1}(4z)^2 - \frac{\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)}{3 \cdot 2 \cdot 1}(4z)^3 + \\ & \frac{\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)\left(-\frac{5}{2}\right)}{4 \cdot 3 \cdot 2 \cdot 1}(4z)^4 - \frac{\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)\left(-\frac{5}{2}\right)\left(-\frac{7}{2}\right)}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}(4z)^5 + \cdots\end{aligned}$$

We can get rid of many powers of 2 and combine things to obtain:

$$(1 - 4z)^{1/2} = 1 - \frac{1}{1!}2z - \frac{1}{2!}4z^2 - \frac{3 \cdot 1}{3!}8z^3 - \frac{5 \cdot 3 \cdot 1}{4!}16z^4 - \frac{7 \cdot 5 \cdot 3 \cdot 1}{5!}32z^5 - \cdots \tag{9}$$

From Equations 9 and 8:

$$f(z) = 1 + \frac{1}{2!}2z + \frac{3 \cdot 1}{3!}4z^2 + \frac{5 \cdot 3 \cdot 1}{4!}8z^3 + \frac{7 \cdot 5 \cdot 3 \cdot 1}{5!}16z^4 + \cdots \tag{10}$$

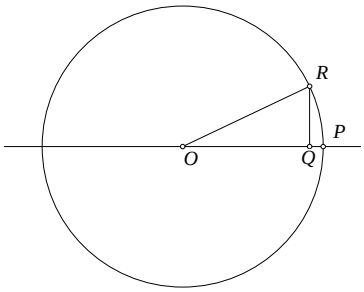
The terms that look like $7 \cdot 5 \cdot 3 \cdot 1$ are a bit troublesome. They are like factorials, except they are missing the even numbers. But notice that $2^2 \cdot 2! = 4 \cdot 2$, that $2^3 \cdot 3! = 6 \cdot 4 \cdot 2$, that $2^4 \cdot 4! = 8 \cdot 6 \cdot 4 \cdot 2$, et cetera. Thus $(7 \cdot 5 \cdot 3 \cdot 1) \cdot 2^4 4! = 8!$. If we apply this idea to Equation 10 we can obtain:

$$f(z) = 1 + \frac{1}{2}\left(\frac{2!}{1!1!}\right)z + \frac{1}{3}\left(\frac{4!}{2!2!}\right)z^2 + \frac{1}{4}\left(\frac{6!}{3!3!}\right)z^3 + \frac{1}{5}\left(\frac{8!}{4!4!}\right)z^4 + \cdots = \sum_{i=0}^{\infty}\frac{1}{i+1}\binom{2i}{i}z^i.$$

From this we can conclude that the i^{th} Catalan number is given by the formula

$$C_i = \frac{1}{i+1}\binom{2i}{i}.$$

5.1 A Strange Geometric Result



Consider the figure above. Assume that the circle has radius r, so $OR = r$ and that the length of RP is 1. What is the length of QP? We know $QP = r - OQ$ and $OQ = \sqrt{r^2 - 1}$ so:

$$QP = r - \sqrt{r^2 - 1}.$$

Let $r = 5$ and we obtain:

$$QP = 0.101020514433643803\ldots$$

Notice that the catalan numbers appear in the decimal expansion; we can see $1, 1, 2, 5, 14$ and "almost" 42. What's going on?

What's even stranger is if we let $r = 5,000,000$. We obtain:

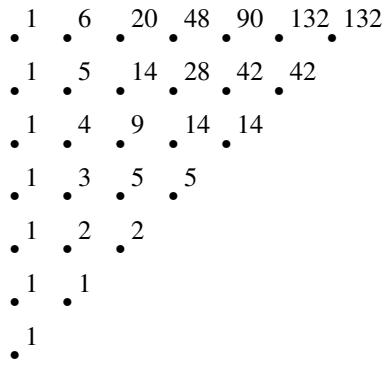
$$\begin{aligned} QP \;=\; & .00000010000000000001000000000000200000000000050000000000001400000 \\ & 00000004200000000000132000000000042900000000001430000000000048620000 \\ & 00000016796000000000587860000000208012000000007429000000000267444000 \\ & 00000096948450000003535767000000129644790000004776387000000176726319 \\ & 00000656412042000024466267020000914825636400034305961365001289904147 \\ & 32404861946401452183673530721526953355091600663747951750370022422166 \\ & 51406149865020924494463603922746434064877050321361304122512394404 2\ldots \end{aligned}$$

where we find the first 25 Catalan numbers surrounded by various numbers of zeroes.

Hint: Look at the generating function for the Catalan numbers.

6 Catalan's Triangle

In this section we will consider a triangle somewhat akin to Pascal's triangle that will provide a nice method to generate the Catalan numbers.



Ignoring the dots for a moment, the rule is simple: As we build up, each row contains one more number than the previous. The first row consists of a single 1. Once a row is complete construct the next row up beginning over the left-most element of the row below and the number placed there is the sum of the number directly below it and the number directly to the left. If there is no number in the slot below or in the slot to the left, just use zero. To make sure you understand the rule, construct the eighth row and make sure that you obtain: $1, 7, 27, 75, 165, 297, 429, 429$.

Notice that the numbers running up the diagonal of this triangle, $1, 1, 2, 5, 14, 42, 132$, are the Catalan numbers. Why is this? You may wish to experiment a little before reading on.

In Section 1.3 we saw that the number of diagonal-avoiding paths is counted by the Catalan numbers. For any point in the triangle above, and consider the problem of counting the number of paths from that point to the bottom-most point where the only allowable moves are one step to the left or one step down, where you are constrained to remain on the lattice points of the triangle.

If we begin at the lowest one, we are already there, so there is only one path; namely, the empty path: don't do anything and you're done. The two 1's in the next row make sense, too, since there's only one path to the bottom from each. (One of the paths is two steps long and one is only one step long, but we are not counting the number of steps, but the number of paths, and there is only one path from each of the points.)

Now start with any other point in the triangle. If that point is on the left of the row, the only possible path is straight down, so there's only one path. If the point is on the right-most end of the row, the only move you can make is to the left, so there are the same number of paths from there to the bottom as there are from the point immediately to your left. For any other point, it is possible to make the initial move to the left or down, in which case the number of paths to the bottom is the sum of the number of paths to the bottom from the point below you and from the point immediately to your left. Notice that this is exactly the same rule we used to generate the numbers in the grid, so every number in the triangle represents the number of paths to the bottom from the point immediately below and to the left of that number.

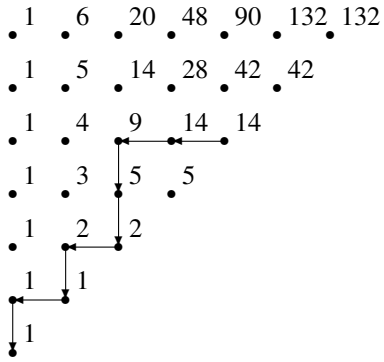
The paths that start on the diagonal are the paths we counted when we were generating the Catalan numbers, so the diagonal numbers are the Catalan numbers.

6.1 Enumerating all Diagonal-Avoiding Paths

A very interesting observation that was pointed out to me by Patrick Labarque is the following. Suppose we'd like to number all the diagonal-avoiding paths, giving the empty path number 0, The

unique path on the triangle with three dots is numbered 1, the two paths on the next larger triangle with six dots get numbers 2 and 3, and so on.

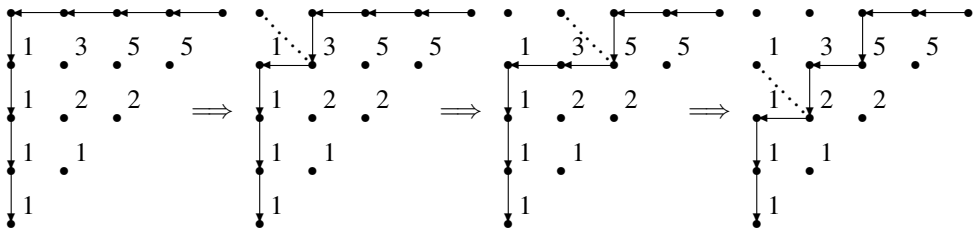
The enumeration can be done by drawing the path as in the sample illustration below:



The number to assign to each path is the sum of the numbers under and to the right of the path; in this case, $1 + 1 + 2 + 5 + 5 = 14$.

It is not too hard to prove that every path from a diagonal point to the bottom-most point has a unique sum of the grid numbers below it and that there are no omitted numbers, assuming that the null path is assigned number 0.

First notice a few things. If we add all the numbers in a particular row, their sum is the Catalan number on the right end of the row above. For example, in the fourth row up, we have $1+3+5+5 = 14$ which is the right-most number in the fifth row up. This makes sense, since if we consider paths starting on the row above, we will move left at least once, but perhaps all the way to the left before taking our first step down. When we take the first downward step, the number below is the number of ways to complete a path where that is the first step down. The sum of all of them, or in other words, the sum of the numbers in the row, is the total number of paths, which is the Catalan number.



To show why the enumeration works, let's assume that all the paths are correctly enumerated for the "mountain range paths" for diagonals of length $0, 1, 2$ and 3. The empty path is assigned the number 0, the unique path that takes one step on the diagonal is assigned 1, the next two are assigned 2 and 3, and the next five are assigned $4, 5, 6, 7$ and 8.

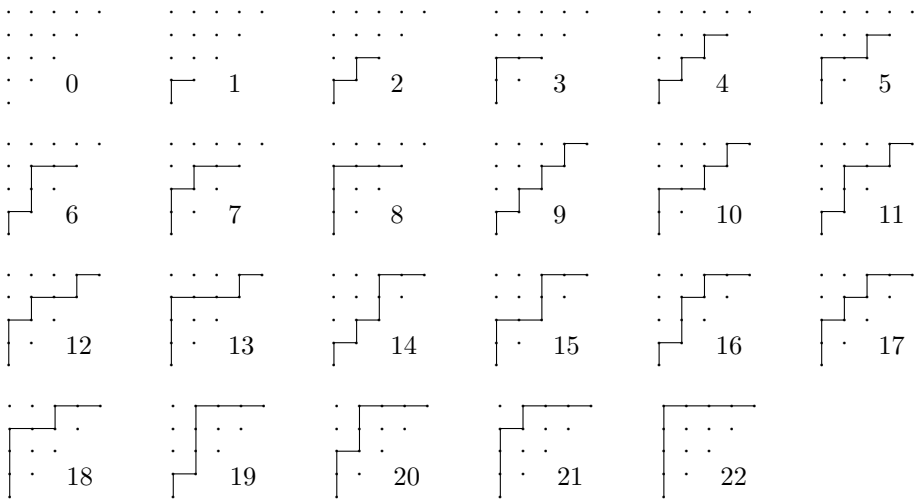
We now would like to make sure that the next fourteen paths are assigned to the numbers $9, 10, \ldots, 22$. If we begin by looking at the path that encloses all the numbers as in the figure above, the top row sums to 14, and that added to the numbers below that sum to 8 yields 22.

We can obtain any other path by doing successive "foldings" of the original path, removing one number at a time and always folding from upper left to lower right. In the figure above, three steps of unfolding are illustrated, and the dotted paths show the motion of each fold.

If any folding occurs, the first one has to be from the upper left corner. This produces a path

with a 1 removed from below it, and it eliminates exactly one path, so this path will be assigned $22 - 1 = 21$. In the illustration, the next fold eliminates three possible paths. Remember that the 3 that was exposed refers to the number of paths from the dot to its lower left to the bottom. When we do that fold, since we've now skipped three paths, we need to subtract 3 from 21, yielding path number 18, et cetera. You can also see that when the path is completely folded down to the minimum path covering the four diagonal steps, the only remaining numbers are $1 + 1 + 2 + 5 = 9$: the first available number after the $0, 1, \ldots, 8$ that were used to enumerate all the shorter paths.

Following is a list of all the paths from number 0 to number 22. The first one (or zeroth one, if you prefer), of course is the empty path. It is a worthwhile exercise to check that the sum of the numbers under at least a few of these paths is equal to the path number.



7 More Examples Without Proof

Here are some more counting problems whose answer is "the Catalan numbers". You can use these as exercises.

7.1 Permutations avoiding 123

A permutation of n numbers consists of a rearrangement of those n numbers. Without any constraints, there are $n!$ permutations. To completely define a permutation, all that is required is an n-tuple of the numbers $\{1, 2, \ldots, n\}$ with the following interpretation: The n-tuple $(p_1, p_2, \ldots, p_n)$ is the permutation that takes 1 to p_1, 2 to p_2, ..., n to p_n. All the p_i in such an n-tuple are distinct. For example, here is a list of all the permutations of the set $\{1, 2, 3\}$:

$$(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1). \tag{11}$$

We will say that a permutation "avoids 123" if in the n-tuple as described above, it is impossible to find three numbers i, j and k such that $i < j < k$ and $p_i < p_j < p_k$. In other words, the n-tuple contains no subsequence of length three that is increasing. For example $(4, 1, 2, 3)$ because you can find 1, 2 and 3 in order. The permutation $(2, 1, 3, 4)$ fails because of the subsequence 2, 3 and 4

(and also the subsequence $1, 3$ and 4). As an example of one that works, the permutation $(4, 1, 3, 2)$ avoids 123.

By definition, all the permutations of 0, 1 or 2 elements avoid 123, since we can't find three different numbers in the n-tuples. There are 1, 1 and 2 such permutations. For $n = 3$, there is only one permutation from the list in List 11 that fails: $(1, 2, 3)$, leaving 5 permutations that avoid 123.

For $n = 4$, there are 24 permutations, 14 of which avoid 123 and 10 of which that do not. Here is a list of the 14:

$$(1,4,3,2),(2,1,4,3),(2,4,1,3),(2,4,3,1),(3,1,4,2),$$
$$(3,2,1,4),(3,2,4,1),(3,4,1,2),(3,4,2,1),(4,1,3,2),$$
$$(4,2,1,3),(4,2,3,1),(4,3,1,2),(4,3,2,1).$$

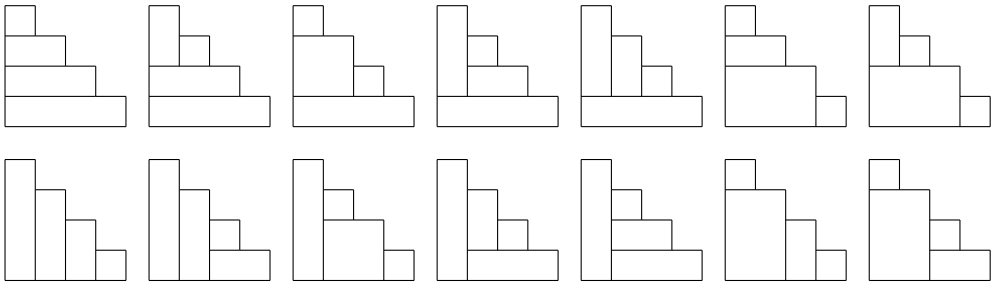
Here is a list of the 10 that do not avoid 123:

$$(1,2,3,4),(1,2,4,3),(1,3,2,4),(1,3,4,2),(1,4,2,3),$$
$$(2,1,3,4),(2,3,1,4),(2,3,4,1),(3,1,2,4),(4,1,2,3).$$

The number of permutations of n elements that avoid 123 is C_n.

7.2 Tiling with Rectangles

Given a "triangular" region composed of n blocks on a side, in how many different ways can the region be tiled with exactly n rectangles? The illustration below shows that for $n = 4$ there are exactly $C_4 = 14$ tilings.



8 Some Interesting Matrices

An interesting property of the Catalan numbers is that if you form either of the following Hankel matrices, their determinants are as follows. These results (and others) are nicely proved using arguments related to counting paths in the following paper by Mays and Wojciechowski:

`http://www.math.wvu.edu/ jerzy/research/21catalan.pdf`

It is also true that the only sequence $C_0, C_1, C_2, \ldots$ that satisfies the following equations is the Catalan numbers.

$$Det\begin{pmatrix} C_0 & C_1 & C_2 & \dots & C_n \\ C_1 & C_2 & C_3 & \dots & C_{n+1} \\ C_2 & C_3 & C_4 & \dots & C_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_n & C_{n+1} & C_{n+2} & \dots & C_{2n} \end{pmatrix} = 1.$$

$$Det\begin{pmatrix} C_1 & C_2 & C_3 & \dots & C_n \\ C_2 & C_3 & C_4 & \dots & C_{n+1} \\ C_3 & C_4 & C_5 & \dots & C_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_n & C_{n+1} & C_{n+2} & \dots & C_{2n-1} \end{pmatrix} = 1.$$

$$Det\begin{pmatrix} C_2 & C_3 & C_4 & \dots & C_n \\ C_3 & C_4 & C_5 & \dots & C_{n+1} \\ C_4 & C_5 & C_6 & \dots & C_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_n & C_{n+1} & C_{n+2} & \dots & C_{2n-2} \end{pmatrix} = n.$$

It is also very interesting to note that the determinants of the following Hankel matrices (and their obvious generalizations) are very tiny:

$$Det\begin{pmatrix} 1/1 & 1/1 & 1/2 & 1/5 & 1/14 \\ 1/1 & 1/2 & 1/5 & 1/14 & 0 \\ 1/2 & 1/5 & 1/14 & 0 & 0 \\ 1/5 & 1/14 & 0 & 0 & 0 \\ 1/14 & 0 & 0 & 0 & 0 \end{pmatrix} = \frac{1}{537824}.$$

$$Det\begin{pmatrix} 1/1 & 1/1 & 1/2 & 1/5 & 1/14 \\ 1/1 & 1/2 & 1/5 & 1/14 & 1/42 \\ 1/2 & 1/5 & 1/14 & 1/42 & 1/132 \\ 1/5 & 1/14 & 1/42 & 1/132 & 1/429 \\ 1/14 & 1/42 & 1/132 & 1/429 & 1/1430 \end{pmatrix} = -\frac{1}{122489812645200000}.$$

Dynamic Programming Questions

Question 1 : EASY

Tribonacci Sequence

The Tribonacci sequence Tn is defined as follows:

T0 = 0, T1 = 1, T2 = 1, and Tn+3 = Tn + Tn+1 + Tn+2 for n >= 0.

Given n, return the value of Tn. [Go to Qs]

Question 2 : MEDIUM

Maximum profit after buying and selling stocks with transaction fees

You are given an array of prices where prices[i] is the price of a given stock on the ith day, and an integer fee representing a transaction fee.

Find the maximum profit you can achieve. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction.

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

The transaction fee is only charged once for each stock purchase and sale. [Go to Qs]

Question 3 : HARD

Longest Increasing Path in Matrix

Given an m x n integers matrix, return the length of the longest increasing path in the matrix.

From each cell, you can either move in four directions: left, right, up, or down. You may not move diagonally or move outside the boundary (i.e., wrap-around is not allowed). [Go to Qs]

Question 4 : MEDIUM

Generate Parentheses

Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses. [Go to Qs]

Question 5 : MEDIUM

House Thief

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police. [Go to Qs]

Question 6 : MEDIUM

Longest Palindromic Subsequence

Given a string s, find the longest palindromic subsequence's length in s.

A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements. [Go to Qs]

Question 7 : MEDIUM

Equal Subset Sum Difference

Given an integer array nums, return true if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or false otherwise. [Go to Qs]

Question 8 : HARD

Mountain Array(Longest Bitonic Subsequence)

You may recall that an array arr is a mountain array if and only if:

- arr.length >= 3
- There exists some index i (0-indexed) with 0 < i < arr.length - 1 such that:
    - arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
    - arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

Given an integer array nums, return the minimum number of elements to remove to make nums a mountain array. [Go to Qs]

Question 9 : HARD

Box Stacking

Given n cuboids where the dimensions of the ith cuboid is cuboids[i] = [widthi, lengthi, heighti] (0-indexed). Choose a subset of cuboids and place them on each other.

You can place cuboid i on cuboid j if widthi <= widthj and lengthi <= lengthj and heighti <= heightj. You can rearrange any cuboid's dimensions by rotating it to put it on another cuboid.

Return the maximum height of the stacked cuboids. [Go to Qs]

Question 10 : MEDIUM

Palindrome Partitioning

Given a string s, partition s such that every substring of the partition is a palindrome. Return all possible palindrome partitioning of s. [Go to Qs]