Commands

21 September 2021 11:23 AM

```
from functools import cmp_to_key
.sort( key = cmp_to_key(compare_function) )
```

2. Building a cache

```
def memo(f):
    cache = {}
    @wraps(f)
    def wrap(*args):
        if args not in cache:
            cache[args] = f(*args)
        return cache[args]
    return wrap
```

Use with @memo

Basic Printer

16 September 2021 9:17 PM

N + N-1 + N-2 + ... + 1 = N*(N+1) / 2 = O(N^2)
From <https://www.interviewbit.com/problems/nestedcmpl2/>

What does it mean when we say that an algorithm X is asymptotically more efficient than Y?
X will always be a better choice for large inputs

From <https://www.interviewbit.com/problems/choose4/>

Maths

16 September 2021 9:23 PM

```
int a = 0, i = N;
while (i > 0)
{
        a += i;
        i /= 2;
}
```
$O(log\ N)$

From <https://www.interviewbit.com/problems/whilecmpl/>

```
    int count = 0;
    for (int i = N; i > 0; i /= 2) {
       for (int j = 0; j < i; j++) {
          count += 1;
       }
    }
```
$O(N)$

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

```
  int gcd(int n, int m) {
        if (n%m ==0) return m;
        if (n < m) swap(n, m);
        while (m > 0) {
           n = n%m;
           swap(n, m);
        }
        return n;
  }
```
$\theta(logN)$

Array Maths

16 September 2021 9:31 PM

1. Pick from both sides:
   Pick B elements from some combination of left and right side of the array.
   Solution - Make prefix and suffix array. Compute max sum possible. O(N)

2. Min Step in Infinite Grid

   Cost to move from point (x, y) to (x1, y1) when 8 movements are possible.
   Cost = max(abs(x1-x) , abs(y1-y))

3. Minimum Lights to activate

   Start with spot=0,
   Find rightmost light j which is less than B distance from 0. [spot-B, spot+B-1].
   Update the spot = j+B. As the found light j can cover next B areas.

   Update the counter for each found spot. And fail if even single case fails.

4. Max Sum Triplet

   $\max\left(A_i + A_j + A_k \right)$ such that $0 <= i < j < k < len(A)$ and $A_i < A_j < A_k$

   Solution:
   a. Prepare a prefix matrix. Prefix[i] will have maximum number from (0, i-1) but smaller than A[i]. Use bisect.insort_left
   b. Prepare a suffix array. Suffix[i] will contain the maximum element from (i+1, len(A))
   c. Compute max(A[i]+prefix[i]+suffix[i])

5. Max Sum Contigous Array

   Contigours subarray of any length with maximum sum.
   Solution:
   a. Use variable max_so_far and max_end_here.
   b. From i = 0 to len(A)-1
      i. Add A[i] to max_end_here
      ii. If max_end_here is less than max_so_far:
         1) A[i] is part of contigous array, update max_so_far to contain A[i]
      iii. If max_end_here become negative i.e. Adding A[i] makes the solution infeasible:
         1) Reset index to start again from i. And max_end_here = 0

6. Add one to number

   String of number , add 1. Solution: Add 1, store carry, use in next iteration.

7. Max Absolute Difference

$$\max\Big(\ f(i,j)\Big) = \max(\ |A_i - A_j| + |i-j|)$$

$$\left|A_i - A_j\right| + |i-j| \ = \begin{cases} (A_i + i)\ -\left(A_j + j\right) & A_i > A_j \ and \ i > j \\ -(A_i + i) + \left(A_j + j\right) & A_i < A_j \ and \ i < j \\ (A_i - i)\ -\left(A_j - j\right) & A_i > A_j \ and \ i < j \\ -(A_i - i) + \left(A_j - j\right) & A_i < A_j \ and \ i > j \end{cases}$$

Case 1 and 2 are same, Case 3 and 4 are same.
$\text{Max1} = A_i + 1$
$\text{Min1} = A_i + 1$
$\text{Max2} = A_i - 1$
$\text{Min2} = A_i - 1$
$\text{Ans} = \max(\ \text{max1} - \text{min1}, \text{max2} - \text{min2}\ )$

8. Partitions:

   No of ways to split arrays into 3 parts such that sum of all partitions are same.
   Solution:
   a. Splitsum = sum(A) // 3
   b. Prepare a suffix array such that suffix[i] contains number of indexes where sum == splitsum
   c. Do a foward pass adding elements, if sum == splitsum : add suffix[i] to total.

9. Maximum Area of Triangle

   Maximum area of traingle parallel to y axis with different verticies.
   Solution:
   For each column, find the leftmost and rightmost index of 'r' and 'g' and 'b' nodes.
   And the most topmost different colour than colour of each node (i, j)

   For each column:
   For each pair of colours in ('r', 'g', 'b'):
   Max(0.5 * maxdiff(c1, c2) * max(left[c3], right[c3]))

10. Flip

    Choose two indices l, r to flip array to maximize the numbers of 1's.
    Solution:
    Find index l, r such that max(count(0) - count(1))
    Update the array to make '0' -> 1 and '1' -> -1
    Apply "Max Sum Contigous Array".
    Range with max sum means max (number of '0' - number of '1')

Value Ranges

16 September 2021 11:48 PM

1. Min Max

   Max, min = A[0], A[1]
   For x in A:
   If x > max: max = x
   Elif x < min: min = x

2. Merge Intervals

   Insert and merge the intervals
   a. Insert new interval
   b. Sort based on start value
   c. Out = [i[0]]
   d. For x in i[1:]:
      i. If x.start < out[-1].end:
         1) Out[-1].end = max(out[-1].end, x.end)
      ii. Else:
         1) Out.append(x)

Simulation Array

17 September 2021 12:05 AM

1. Perfect Peak of an Array

   Find A_i such that it is greater than all element on left and smaller than all element on right.
   Solution:
   Build Prefix and Suffix to store maxtill and mintill i.
   Iterate and find which $prefix_i < A_i < suffix_i$

2. Kth row of a pascal triangle
   Start with [1]
   Build newA = A[i-1] + A[i] for i in [1, len(A)]

3. Spiral Order Matrix
   startRow, endRow, startCol, endCol = 0, A, 0, A
   Out = 0[A][A]
   While sr < er and sc < ec:
   ```
   For j in range(sc, ec):
       Out[sr][j] = counter++
   Sr += 1
   For i in range(sr, er):
       Out[i][ec-1] = counter++
   Ec -= 1
   If sc < ec:
       For j in range(ec-1, sc, -1):
           Out[er-1][j] = counter++
       Er -= 1
   If sr < er:
       For i in range(er-1,  sr, -1):
           Out[i][sc] = counter++
       Sc += 1
   ```

4. Anti Diagonals

   Iterate $0 \leq i < 2*size - 1$
   And for each i $\begin{cases} 0 \leq j \leq i \quad for\ i < size \\ size - 1 \leq j < i - size\ for\ i \geq size \end{cases}$

Bucketing

17 September 2021 12:29 AM

1. Triplets with sum in a range(A, B)

    a. Filter out the element greater than B.
    b. Sort the array.
    c. Index, l, r = 0, 1, len(arr)-1
    d. While l < r:
        i. Tsum = A_i + A_l + A_r
        ii. If tsum in range: return 1
        iii. If tsum less than A: increase index, l
        iv. If tsum is more than B: decrease r

2. Balance Array

    Count ways to remove A_i to make array balanced.
    Make suffix array. Add even element, subtract odd element.
    Make forward pass, check if presum == suffix[i+1].
        Then add even element to presum, subtract odd element to presum

3. Duplicate in Array
    Sum(A) - (Sum of first N Natural Numbers)

4. Maximum Consective Gap

    a. Find max and min of Array
    b. Initialise the buckets of max and min elements.
    c. Make a delta = max - min // (len(A)
    d. Store the values as per the bucket
    e. Iterate the buckets in order and find max_gap.

Arrangement

17 September 2021 1:49 AM

1. Sort Array with squares.
   Notice the sorted given array. Find O(N) solution

2. Largest Number
   Arrange positive number to form largest number.
   Solution:
   a. Sort the numbers using custom comparator.
      Cmpfn(a, b):
         L, r = str(a) + str(b) , str(b) + str(a)
         If int(l) < int(r) : return 1
         Return -1
   b. Join numbers and return

3. Rotate Matrix

   Swap(A_{ij}, A_{ji}) for all $0 \leq i < R \; and \; i \leq j < C$.
   Swap(A_{ij}, A_{ik}) for all $0 \leq i < C$ and (j=0, k=C-1 ; j < k; j++, k--)

4. Next Permutation

   Solution:
   a. Find a index from right where $A_i \geq A_{i+1}$
   b. If all element are descending : return sorted(A)
   c. Find cindex where strictly greater than A[index] on right
   d. Swap A[index], A[cindex]
   E. Sort the right side (Already reverse sorted)

5. Find Permutation

   Given string of I and D and len(numbers). Find perm such that I and D are satsified.
   Min, max = 1, B
   N times: add minv for I and maxv for D.

Sorting

17 September 2021 2:24 AM

1. Noble Integers

   Find p in array A such that num($A_i > p$) == p
   Traverse in sorted order, store the number of elements greater than A[i]

2. Wave Array

   Sort into wave like array
   a. Sort into ascending
   b. Switch two elements at a time.

3. Hotel Bookings Possible

   Sort datelists
   For each element in datelists:
   If start_time is greater than time t : Remove top element. Add to booklist
   Elseif start_time is less than time t : if empty slot available push it.

4. Max Distance

   Find $\max(j - i)$ with $A_j \geq A_i$.
   Make leftMin prefix an rightMax suffix array.
   Tranverse and keep updating (j-i)

5. Maximum Unsorted Subarray
   Minimum subarray to sort to make whole array sorted

   Assume that Al, …, Ar is the minimum-unsorted-subarray which is to be sorted.
   then min(Al, …, Ar) >= max(A0, …, Al-1)
   and max(Al, …, Ar) <= min(Ar+1, …, AN-1)

   a. Compute MAX prefix and MIN suffix
   b. For i in range(0, len(prefix)
      i. If prefix[i] != suffix[i]:
         1) If flag == 0 : start, flag = i, 1
         2) Else: end = i
   c. Return [start,end]

Space Recycle

17 September 2021 2:50 AM

1. Set Matrix Zero

   Set entire row and column 0 if there exist a 0.
   Store row and col info. Update matrix

2. First Missing Integer

   Find first missing positive integers from unsorted array in range(1, N)
   Solution:
   a. Move all negative element before j and positive number after j. Make negative numbers positive.
   b. For each abs(element) after j. If in range. Make the element negative.
   c. Find first positive element . Return index.

3. Maximum Sum Square Submtrix

   Matrix with maximum sum
   Compress Column sum.
   Then perform rowsum.

Missing/ Repeated Number

17 September 2021 3:03 AM

1. Repeat and Missing NumberArray

   Make all element negative if found. If already negative, this is a repeat number.
   Check which element is not negative, that element is missing number.

2. N/3 Repeat Number

   Find two most occuring element. The count their frequency and find N/3 repeat number.

```
for(int i=0;i<n;i++)
{
    if(first == A[i])           count1++;
    else if(second == A[i])     count2++;
    else if(count1 == 0)    {   count1++;   first = A[i];   }
    else if(count2 == 0)    {   count2++;   second = A[i];  }
    else                    {   count1--;   count2--;       }
}
```

ADHOC

17 September 2021 3:08 AM

1. Total Moves for Bishop

   Diagonal distance between (x, y) and (x1, y1) = min(abs(x-x1), abs(y-y1))
   Ans = min(8-A, 8-B) + min(8-A, B-1) + min(A-1, 8-B) + min(A-1, B-1)

2. Prime Sum

   Brute Force check prime

3. Sum of Pairwise Hamming Distance

   Work with respect to the 32 bits. Find count of 1 and 0 in each row.
   Sum = count0 * count1 * 2 for each bit .
   AND with 2**i or 1<<i to get ith bit

4. Step by Step

   Given a number N find out how many steps required to reach n. At ith step take i steps forward or backward.
   Solution:
   a. Find n such that $n^2 + n - 2 * A \geq 0$ Take ceil for all values.
   b. Calulate sum till n terms. Return n if sum is equal to target
   c. If diff is odd. Increase n by 1 , add n to sum.
   d. Check again if diff is odd, increase n by 1. Return n

5. Power of Two Integers

   Check if n can be expressed as A^P.
   Solution:

```
for p in range(2,33):
    if round( A ** (1/p) ) ** p == A:
        return 1
return 0
```

Digit Op

17 September 2021 1:00 PM

1. Palindrome Integer

   A == A[::-1]

2. Reverse Integer

   Reverse the number. If the number is greater than $2^{31} - 1$
   Return 0 else the reverse numbers

3. Next Smallest Palindrome

   a. Make String Mutable
   b. Find size, mid.
   c. Traverse from mid to size :
      i. If equal continue,
      ii. If left side is less than right side then incr is necessary.
      iii. If right side is less than than left side then incr is not necessary
   d. Take left half of the array.
   e. If incr is true:
      i. Carry =1
      ii. Traverse from i = mid-1, 0:
         1) If carry is 0: break
         2) If A[i] == 9: set 0
         3) Else: A[i], carry = A[i] + carry, carry -1
      iii. If carry: return '1' + n-1 '0's + '1'
   f. If size % 2 == 0: A = A + A'
   g. Else: A = A + A[:-1][::-1]
   h. Return buidstring(A)

Number Theory

17 September 2021 6:00 PM

1. GCD

   Gcd(A, B) = A if B == 0
   Gcd(B, A%B) else

2. Nth Fibonacci

   M = [[1, 1], [1, 0]]
   F = [[1, 1], [1, 0]]
   Do matrix multiplication N times
   F^N = then F[0][0] is Nth fibonacci number.

3. Trailing zeros in Factorial

   Count = 0
   While A >= 5:
   A = A // 5
   Count += A

4. Sorted Permutation Rank

   For each character C at index i,
   Rank += (Count of characters less than C) * (N-1!)

5. Largest Coprime Divisor(A, B)

   X divides A and X and B are coprime

   Repeat:
   - Set a, b = A, B
   - While B1 > 0:
     - Set a, b = b, a%b
   - If a = 1: return A
   - A = A // a

6. Sorted Permutation Rank with repeats

   No of permutation with char C as first character:
   N-1 ! // p1 ! * p2 ! * p3! ..
   Use inverse modulus division.

   Rank(A):
   - Rank = 1
   - For each character in A:
     - Count elemenets less than A
     - Count of repeatition of eleemnts after c
     - Calculate (minCount * Fact(total - i - 1)// repeat facts

Number Encoding

19 September 2021 12:12 PM

1. Next Similar Number

    a. Travel from right to left and find i where $C_i < C_{i+1}$.
    b. Find smallest greater element than above element i.
    c. Swap above digits.
    d. Sort all digits after i.

2. Rearrange Array

    a. A[i] = A[i] + (A[A[i]] % n) * n
    b. A[i] = A[i] // n

3. Number of length N and value less than K

    Set A , How many number of len B are possible with value less than C

    Case1 : If B > len(C) : d = 0
    Case2: If B < len(C) : if 0 in A: d^ B else (d-1) * d^(B-1)
    Case3: If B == len(C):

        Digit[]
        First(i) = numbers formed by first i digits
        Lower(i) = number of elements in A less than i. Similar to prefixSum

        Dp[i] =

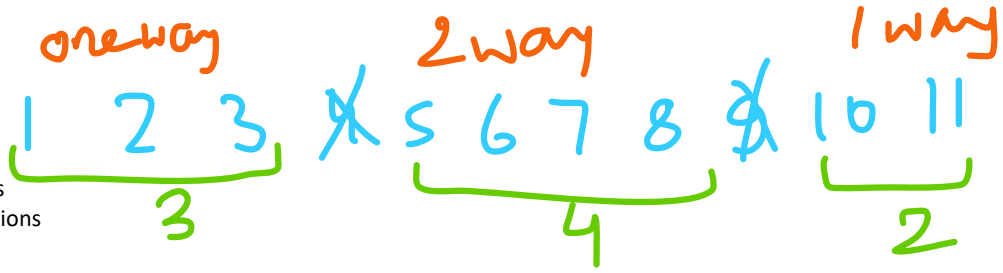
    i) For all numbers F(i-1) is less than F(i-1) of C, dp[i] += dp[i-1]*d
    j) For all number F(i-1) == F(i-1) of C, dp[i] += lower[digit[i]]

Combinatorics

19 September 2021 3:01 PM

1. City Tour

   a. Calculate total togehter eleents
   b. Totalperm * twoways // reductions



$$\frac{n!}{n_1!\, n_2! \cdots n_5!} \times \left(2^{n_2-1} \cdot 2^{n_3-1} \cdots\right)$$

$$= \frac{9!}{3!\,4!\,2!} \times 2^{4-1}$$

2. Grid Unique Paths

   Top row and left col = 1
   Else: M[i][j] = M[i-1][j] + M[i][j-1]

Simple Binary Search

19 September 2021 5:03 PM

Binary Search

```
def b_search(self, A):
    l, r = 0, len(A)-1
    while(l <= r):
        mid = (l+r)//2
        if( A[mid] > A[mid-1] ):                l = mid + 1
        elif ( A[mid] < A[mid-1] ):             r = mid - 1
         else:                                   return mid
    return -1
```

1. Bitonic Array

   First Increasing then decreasing sequence.
   a. Use binary search to find point of switch. Apply bsearch in incresing and inverse binary search in decreasing part

   To find point, go to mid point: if mid element is less than mid-1, set index=mid shift left r else shift right l.

2. Smaller or equal elements

   Find the point where A[i] <= B

```
def greater_search( A, ele ):
    l, r, index = 0, r, len(A)
    while(l <= r):
        mid = (l+r)//2
        if( A[mid] <= ele ):     l = mid + 1
        else:                    index, r = mid, mid - 1
    return index

def smaller_search( A, ele ):
    l, r, index = 0, r, -1
    while(l <= r):
        mid = (l+r)//2
        if( A[mid] <  ele ):     l, index = mid, mid + 1
        else:                    r = mid - 1
    return index
```

3. Woodcutting made easy

   Binary Search : At mid point, find the amount of wood generated. Else repeat binary search process.

4. Matrix Search

   Search for proper row, then search for proper column

5. Search for range

   Apply greater search and smaller search to get immediate larger and smaller element.

6. Sorted Insert Position

   Apply greater binary search with equality to index side.

Search Answer

19 September 2021 7:42 PM

1. Matrix Median

   Find the max and min element by comparing the max and min of each row.
   The median is R * C + 1 //2
   Apply binary search.
   For each mid, calculate the number of element less than mid. Use binary search method.

2. Square Root of Integer.

   Sqrt(A):
   a. Lo, hi, root = 0, A, -1
   b. While lo <= hi:
      i. Mid, sq = (lo + hi)//2 , mid*mid
      ii.

If sq > A:	Hi = mid-1
Else	Root, lo = mid, mid + 1

   c. Return root

3. Allocate Books

   Allocate books to B students so that pages are minimized.
   a. Lo, hi, root = max(A), sum(A), -1
   b. While lo <= hi:
      i. Mid = (lo + hi)//2
      ii. Sq = Check(A, B, mid)
      iii.

If sq > A:	Hi = mid-1
Else	Root, lo = mid, mid + 1

   c. Return root

   The check function will compute how many students will get books if mid pages are atleast alloted to each student.

4. Painter Partition Problem

   Same as above, just the valid function will recieve mid which will denote how much time one painter can spend.
   Keep assigning board to painters till timetaken < mid.

Search Step Simulation

19 September 2021 8:03 PM

1. Power Function

   $x^n \% d$

```
def pow(self, x, n, d):
    res = 1 % d  # Cover case d == 1
    while n > 0:
        if n & 1:   # Odd?
            res = (res * x) % d
        x = x**2 % d
        n >>= 1
    return res
```

2. Simple Queries

   a. Sieve of Eratosthenes

      Product of all divisor of i.

```
p = [1] * (max(A) + 1)
for i in range(1, len(p)):
    for j in range(i, len(p), i):
        p[j] = (p[j] * i) % 1000000007
```

   a. For each element , find how many subarray has x as max.
      Find no of smaller element in left and number of smaller element in right

   b. Store prod of divisor with their frequency. Sort the values.
   c. Store cummulative sum of frequency. Apply binary search for each query.

3.

Sort Modification

19 September 2021 8:35 PM

1. Median Of Array

   Switch Arrays if len(A) > len(B)

```
lo, hi, combined = 0, m, m+n
```

```
while lo <= hi:
    partX = (lo + hi) / 2
    partY = ((combined + 1) / 2) - partX
    LeftX, rightX = self.get_max(A, partX), self.get_min(A, partX)
    LeftY, rightY = self.get_max(B, partY), self.get_min(B, partY)
    if LeftX <= rightY and LeftY <= rightX:
        if combined % 2 == 0:
            return (max(LeftX, LeftY) + min(rightX, rightY)) / 2.0
        else:
            return max(LeftX, LeftY)
    if LeftX > rightY:
        hi = partX - 1
    else:
        lo = partX + 1
return -1
```

2. Rotated Sorted Array Search

```
def search(self, A, B):
    l, r = 0, len(A) - 1
    while( l <= r):
        mid = (l+r)//2
        if(A[mid] == B):                                    return mid
        if( A[mid] <= A[r] ):
            if( B > A[mid] and B <= A[r] ):                 l = mid + 1
            else:                                           r = mid - 1
        else:
            if( A[l] <= B and B < A[mid] ):                 r = mid -1
            else:                                           l = mid + 1
    return -1
```

Search

19 September 2021 8:51 PM

KMP StrStr()

```
def create_lps(pat):
    lps = [0] * len(pat)
    i = 1
    j = 0
    while(i < len(pat)):
        if pat[i] == pat[j]:
            j += 1
            lps[i] = j
            i += 1
        else:
            if j == 0:
                lps[i] = 0
                i += 1
            else:
                j = lps[j-1]
    return lps
class Solution:
    def strStr(self, A, B):
        lps = create_lps(B)
        i = 0
        j = 0
        while( i < len(A) ):
            if A[i] == B[j]:
                i += 1
                j += 1
            if j == len(B):
                return i - j
            elif i < len(A) and A[i] != B[j]:
                if j != 0 :
                    j = lps[j-1]
                else:
                    i += 1
        return -1
```

2. Stringholics

Minimum time after rotated produce same string

Compute LPS of repeat array

```
def computeLPSArray(self, pat):
    i=(pat+pat).find(pat,1,-1)
    return len(pat) if i == -1 else i

def solve(self, A):
    ans = 1
    for s in A:
        res = self.computeLPSArray(s)
        for i in range(1, 2*res+1):
            t = ( i * (i+1) // 2 ) % res
            if t == 0:
                ans = ( ans * i )// gcd(ans, i)
                break
    return ans % 1000000007
```

Tricks

19 September 2021 9:15 PM

1. Min characters to add to make palindromic

   Lps of str + '*' + str
   Return len(str) - lps[-1]

2. Convert to Palindrome

   Remove one character to make the string palindromme
   Traverse l, r
   If A_l ! $= A_r$: Check Palindrome without l or r element. If exist return 1 else 0
   Else: l++, r--

3. Minimum Appends for Palindrome

   Minimum insert at end to make it palindrome
   Lps =createlps(A.reverse() + '*' + A)
   Return len(A) - lps[-1]

4. Minimum Parenthese

   Minimum parenthesis to add to make it valid.
   Push '(', Pop '(' for ')'.
   If '(' does not exist for ')' Increase counter by 1
   Return counter + len(stack)

5. Longest Palindromic Substring

   For each i, scan toward left and right till matching sequence exist. Update the max values if necessary

String Maths

20 September 2021 7:23 AM

1. Roman To Int

   Add value of symbol:
   If prev value is less than current value. Decrease twice the value

2. Binary Addition

Half adder	Full Adder
Sum = a ^ b	Sum = a ^ b ^ c
Carry = a and b	Carry = (a and b) or (b and c) or (a and c)

   Keep adding s1 and s2 untill one of them is empty
   Empty the remaining strng
   Add the carry

3. Power of 2

   If A and (A-1) is zero : return 1

4. Multiply Strings

```
def multiply(self, A, B):
    result = [0] * (len(A) + len(B))
    i_n1, i_n2 = 0, 0
    for i in range(len(A) - 1, -1, -1):
        carry, n1, i_n2 = 0, ord(A[i]) - 48, 0
        for j in range(len(B) - 1, -1, -1):
            n2 = ord(B[j]) - 48
            sumv = n1 * n2 + result[i_n1 + i_n2] + carry
            carry = sumv // 10
            result[i_n1 + i_n2] = sumv % 10
            i_n2 += 1
        if(carry > 0):
            result[i_n1 + i_n2] += carry
        i_n1 += 1


    i = len(result) - 1
    while( i >= 0 and result[i] == 0):    # Ignore leading zeros
        i -= 1

    if( i == -1 ):                        # Check if result does not exist
        return "0"

    s = ""                                # Build the string
    while( i>= 0):
        s += chr(result[i] + 48)
        i -= 1
    return s
```

Bit Play

20 September 2021 7:55 AM

1. Number of 1 bits, Trailing Zeros

   A&1 and A >> 1

2. Reverse Integers

```
int x = 0, t;
for(int i = 0; i < 32; i++)
{
    t = A&1;
    A = A >> 1;
    x = (x << 1 ) | t;
}
return x;
```

3. Divide integers using bit operation

```
q, t = 0, 0
for i in range(31, -1, -1):
    if( t + ( B << i ) <= A ):
        t += B << i
        q |= 1 << i
```

4. Different Bits Sum Pairwise = Max hamming distance

   For each bit sum += count0 * count1 * 2

Bit Tricks

20 September 2021 8:08 AM

1. Min XOR values

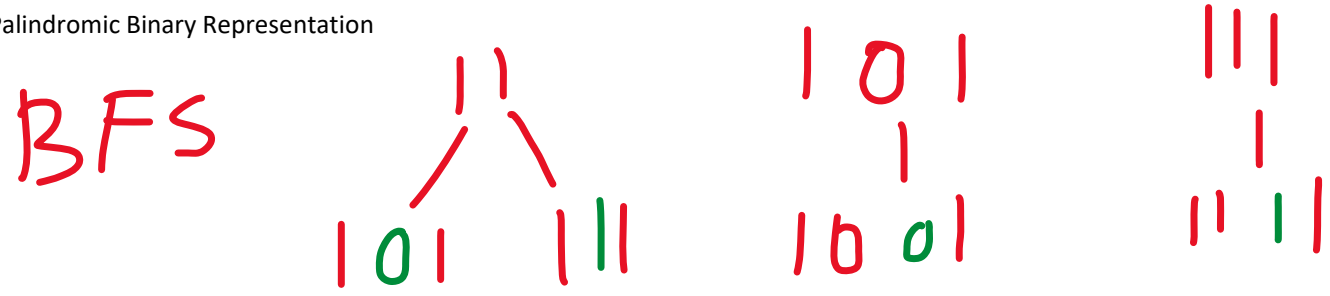
   Minimum xor value is between consecutive values. Sort and find out.

2. Count total set bits

   Total set bits for perfect power of 2's = $(x * 2^{x-1}) + (N - 2^x + 1)$
   Where N is the number and x is the power with 2^x <= N

```
def solve(self, A):

    if( A <= 1 ):
        return A
    x = findmaxpow(A)
    cnt1 = x *  2**(x-1)             # Count of set bits before power of 2
    cnt2 = A - 2**x + 1              # Count of set bits after power of 2
    prevcnt = self.solve( A - 2**x ) # Count of set bits after power of 2 subsearch
    return ( cnt1 + cnt2 + prevcnt ) % 1000000007
```

3. Palindromic Binary Representation



   Run n times, with start element as '11'. Keep pop queue n times and return the number.

4. Xoring subarrays

   Elements occuring only odd number of times will count in final result.
   Value at ith index will occur (i+1) * (N-i) times.
   If N is even : Answer is zero.
   Else : Answer is xor of even indexed elemments.

```
def solve(self, A):
    ans, n = 0, len(A)
    for i in range(n):
        # count the number of subarrays containing A[i]
        # there are i+1 choices for starting index and
        # n-i choices for ending index
        num = (i + 1) * (n - i)

        # if a number occurs in an even number of subarrays,
        # it will be XORed an even number of times and
        # hence will not be reflected into the final answer
        # else it will be reflected just once.
        if num & 1:     ans ^= A[i]
```

```
        return ans
```

5. Single Number with other having 3 occurence

```
    def singleNumber(self, A):
        res = 0
        for i in range(31, -1, -1):
            count = 0
            for x in A:             count += (x >> i) & 1
            if( count % 3 == 1 ):   res = (res << 1) + 1
            else:                   res = (res << 1) + 0
        return res
```

Sorting

20 September 2021 9:45 AM

1. Pair with given difference

   Make a set. Check if B+x or x-B exist in set.

2. 3 Sum

   For each element, apply two pointer.
   Two pointer, take left index and right index. Keep shifting index as per requiremment

3. Counting Trianlges

   Sort(A)
   For each edge, apply two pointer.
   $A_i < A_j < A_k$
   Then check only for $A_i + A_j > A_k$.

   If possible, then A_j and all values from i to j-1 will form a traingle.
   Add j-i to count.

4. DIffK

   Same as 1st question

Tricks

20 September 2021 10:06 AM

1 . Max Ones after Modifiction

Two pointer, left and right. Move right by one is zero count <= B.
Move left by one if zerocount > B.
Update bestl, bestr

2. Counting SubArrays

Same as above. Move right by one if sum + A[ir] < B. Update count = ir - il as all subarrays with ir and rest of previous element have sum less than B
Move left by one if sum + A[il] >= B

3. Subarray wth distinct Integers

Count of atmost B values :

```
il, ir = 0, 0
count = 0
while ir < len(A):
    Add A[ir] into the data
    while len(frq) > B :
        Remove A[il] from data
        il += 1
    count += ir - il + 1
    ir += 1
return count
```

```
Count of exactly B values:
     countAtmost(A, B) - countAtmost(A, B-1)
```

4. Max continous series of 1s

Same as 1

5. Array 3 Pointers

Find i, j, k such that
$\max(\left|A_i - B_j\right|, \left|B_j - C_k\right|, \left|C_k - A_i\right|)$ is minimized.

```
def minimize(self, A, B, C):
    i, j, k     = 0, 0, 0
    store_max   = sys.maxsize
    while( i < len(A) and j < len(B) and k < len(C) ):
        minv = min(A[i], B[j], C[k])
        maxv = max(A[i], B[j], C[k])
        store_max = min( maxv - minv, store_max )
        if store_max == 0:      break
        if A[i] == minv:        i += 1
        elif B[j] == minv:      j += 1
        else:                   k += 1
    return store_max
```

Keep changing the minimum value to next index.

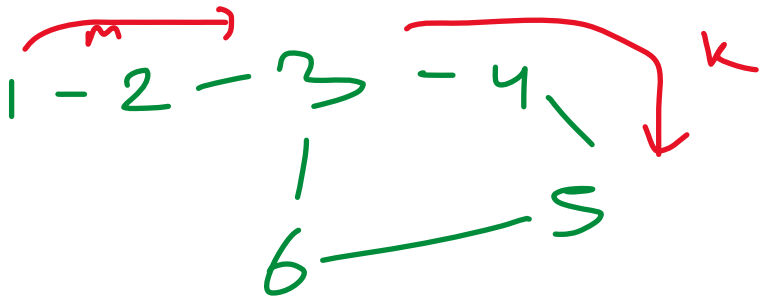
6. Container with most water

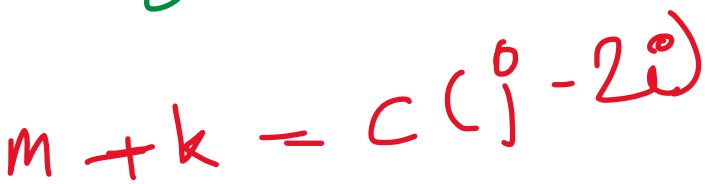
   (i, A[i]) denotes a coordinate. Find max water holding capactiy by a container with x axis and two wall.

   Keep inward moving pointers. L = 0, R = MAX
   Shift lower bar inward. Area = R-L * min(A[l], A[r])

Cycle List Detection

20 September 2021 11:56 AM






```
ListNode *slow = A, *fast = A;
slow = slow->next;
fast = fast->next->next;
while( fast && fast->next )
{
    if ( slow == fast )
        break;
    slow = slow->next;
    fast = fast->next->next;
}
if( slow != fast )
    return NULL;
slow = A;
while( slow != fast )
{
    slow = slow->next;
    fast = fast->next;
}
return slow;
```

Simple Problems

20 September 2021 1:43 PM

1. Balanced Paranthesisi

   Add '(' to stack
   Pop '(' for ')'

2. Simple Directory Path

   Do nothing for '.'
   Pop one for '..'
   Push everything else

3. Redundant Braces

   When ')' is found.
   Set flag = 1, pop element till '(' is found and set flag = 0 if any operation is found.
   If flag is 0, braces are redundant.

4. Min Stack

   Easy Solution

Push	Add x to stack. Add x to minstack if x is less than or equl to top of minstack
Pop	Return last element of stack, If last element is same as top of minstack, then remove top of minstack also
getMin	Last element of Minstack

   Complex Solution:

Push	If x is less than minele: Add $2x - minele$ to stack and make x the new minele. If x is greater or equal: Simply add x to stack
Pop	If top element is less than minele : $minele = 2 * minele - top$
Top	If top element is less than minele: return minele Else: return top element

   If lower element arises, add 2*x to minele and update minele
   Therefore if current element in stack is greater than minele, then the current element is not the minele
   If current element become less than minele, the current element is the minele. Removing it will require
   To update minele to previous minele.

CleverStack

20 September 2021 2:13 PM

```
specials = []
stack = []
for i in range(len(A)):
    while len(stack) and stack[-1] >= A[i] :
        stack.pop()
    if len(stack):
        specials.append(stack[-1])
    else:
        specials.append(-1)
    stack.append( A[i] )
```

Cleverstack is way to find the strictly greater or lower element in left or right of it.

For a minimum cleverstack in left.
Traverse from i = 0 --- > len(A):
    While A[i] <= top : keep removing the top
    For ith element, minimum element on the left is the top of stack if it exists.
    After that, add A[i] to stack.

1. MAXPROD

   Max of (leftSpecialValue * RightSpecialValue)
   leftSpecialValue(i) is rightmost j in the left with greater element
   rightSpecialValue is leftmost j in the right with greater element.

   Run cleverstack twice, store leftspecialvalue for each index, then calculate rightspecialvalue. Return max

2. Nearest Smaller Element

   Simple cleverstack problem with popping of larger and equal elements.

3. Largest Rectangle in histogram

   For each bar i.
    If stack is empty or A[i] >= stack_top_value:
        Add index i to stack
    Else:
        Top = s.pop()
        If element in stack:
            // Area is height of top(popped element) * (difference between current and previous minimum)
            Area = A[top] * (i - s[-1] - 1)
        Else:
            // If top is the minumum of all, area = top * (current element - 0)
            Area = A[top] * i

    // Clear the stack
    While stack:
        Area = top * (i - s[-1] - 1) or top * i

```
    Keep updating max area.

--- Simple Solution :
```

Use simple cleverstack. Keep removing the smaller valued element from stack.
At every point in time, cleverstack contains the index for which height is minimum in the left.

```
height.append(0)
stack = [-1]
ans = 0
for i in range(len(height)):
    // While curr height is smaller than previous minumums, pop the
 indeex. Calculate the area. Finally add current height
   while height[i] < height[stack[-1]]:
       h = height[stack.pop()]
       w = i - stack[-1] - 1
       ans = max(ans, h*w)
    stack.append(i)
height.pop()
return ans
```

Queue

20 September 2021 4:15 PM

1. First non repeating in steam of characters.

   Use ordered map or map + queue

2. Sliding Window Maximum

   CleverQueue.
   Remove from queue back untill a higher value exist. Then append value. Operate with indexes
   For each index i, remove all indexes <= i-B. Add top of queue.

3. Evauate Expresion

   Push to stack, numbers, pop last 2 and operate

4. Rain Water Trapped

   Easy Solution :
   Keep a left max and right max.
   At each step, either shift left or right whichever is less.
   Update leftMax and rightmax
   Add water += leftMax - A[l] or rightMax - A[r]

Maths

20 September 2021 5:30 PM

1. Maximal String

   If $A_i < A_j$:
   Swap them, recurse with one less swap

2. Kth permutation sequence

   At every point we have to find k/(n-1)!
   K = k % (n-1)!
   N = n - 1

3. Gray code:

   Return [i ^ (i // 2) for i in range(2**A)]

   Or
   Start = [0, 1]
   Add 0 to add values , then add 1 to all values
   Join above two sets

Subsets

20 September 2021 5:46 PM

1. Subsets

   Add Ai to subset, Recurse with it. Remove the element

2. Combination Sum

   Add the element , increase sum. Recurse with it.
   Remove the element, decrease sum

3. Combination sum.

   While recursing increase index by 1 to avoid repeated use of same element.

4. Subsets 2 (Contains duplicates)

   Same as 1

Brute Force Builder

20 September 2021 5:53 PM

1. Letter phone

   Build dictionary . Recurse normally

2. Palindrome Partitoning

   If palindrome, recurse with right values.

3. Generate all parenthesis 2.

   If left is less than N : Try to append (. Recurse
   If right is less than N: Try to append). Recurse

4. Permutations

   For each i in (l, r):
   Swap A_l, A_r
   Permute with l+1
   Swap again l,r

5. Nqueens

   Use rowmask, leftdmask, rightdmask
   Recurse for each column starting with 0:
   For i in range(columns):
   Set row, ld, rd masks
   Recurse to next row
   Reset masks
   Return true or false as required.

6. Sudoku

   Row, Column, Block Mask
   Preporcess masks
   Recurse in row order format:
   For i in range(i, row,col):
   Add i to row, col and mask it
   If possible(i, row, col):
   Recurse with next empty box

   Reset i to row, col and unmask it

Search

20 September 2021 6:07 PM

1. Larget Continous Sequence zero sum

   Calculate prefix sum
   For each element i:
   Keep adding prefix[i]
   Update if prefix[i] is zero or negative of sum exist.

2. Longest Subarray length

   Largest range with 1 more 1 than 0
   Change 0 to -1
   Apply above method with target sum as 1

3. 4 SUM

   Use two loops and one two pointer

4. DiffK

   Easy add $B + A_i$ and $A[i] - B$

Keys and Maths

20 September 2021 6:19 PM

1. Linked List Copy

   Create nodes and add these nodes to map
   Retravel nodes and add links wherever necessary

2. Rearrange to form palindrome

   Only one character can exist as 1 others should be even

3. Points on Straight Line

   Add slope and intercept m, c to map

Incremental Hash

20 September 2021 6:36 PM

1. Two out of Three

   Set 1 for elements in A
   Or with 2 for elemenet in B
   Or with 4 for elements in C

   If key is 1, 2 or 4 : Single Occurence.
   Else : more than 2 value exist

2. Subarray with B odd numbers

   Create prefix sum with odd counts. Use to hash

3. Window String

   Prepare target hash, source hash.
   Keep adding element till counter == m
   Then keep increasing l till counter < m

   Update max length windows size

Map

20 September 2021 7:09 PM

1. Distinct Numbers in A windows

   Add an element, Fill lenght, Remove left elemnt

2. LRU Cache

INIT	Lru Array Map Dictionary Capacity
GET	If key does not exist : return -1 Update lru, return value
SET	Add value and update lru. If size > capacity : remove least used element

Heaps

20 September 2021 8:07 PM

1. Ways to form max heap

$$F(n) = {}^{N-1}C_{l} \times F(l) \times F(r)$$

For i in range 2, A+1:
  Calculate number of left keys
  Apply F(i)

To calculate no of elements in left
Getleft(N):
    Left = N // 2
    Right = N - 1 - left
    While (left + 1) & left != 0 and right + 1 & right != 0
        Left, Right = +1, -1

Atleast one of the sides should have power of 2 elments.

2. N Pair Max Combinations

Sort A, B
Push A[-1] + B[-1] with key as (N-1, N-1)
At every iteratoin, pop an element and push to out.
For popped element i, j. Push i-1, j and i, j-1 in heap if does not exist.

3. Profit Maximization

Add all element to heap.
Pop element with highest cost, add to cost. Push again with 1 less cost.

4. Merge K sorted Arrays

Push first element of each row. Pop minimum along with row and col info. Add popped value to out, and next value to heap.

5. Magicians and Chocolates

Heapify with max cost. Pop maximum and push back value // 2

6.

Easy

20 September 2021 8:32 PM

1. Highest Product of 3 values

   Find 3 maximum and 2 minimum values
   Return max(max1 * max2 * max3 , max1 * min1 * min2)

2. Bulbs

   Minimum switch flips to turn on all bulbs
   Traverse from left to right
   $If\ count\ \%\ 2\ ==\ 1:\ count\ +=\ (\ A[i]\ ==\ 1\ )$
   $If\ count\ \%\ 2\ ==\ 0:\ count\ +=\ (\ A[i]\ ==\ 0\ )$

   If count is even:
   If current bulb state is 0 : mean bulb will be actually off
   Else: bulb was on

   If count is odd:
   If current bulb state is 1: bulb will be actually off
   Else: bulb will be on

3. Disjoint Intervals

   Sort the intervals by last value.
   Start with first value.
   Keep moving forward. If start is more than prev end, update end and count

4. Largest Permutation

   Make atmost B swaps to make the array largest.

   Algorithm:

   a. Make a map with index of all elements
   b. Run B times:
      i. Find i where Max - i != A[i]
      ii. Find the position of (Max - i)
      iii. Swap the elements and update the hashmap

Medium

20 September 2021 8:47 PM

1. Meeting Rooms

   Assign Meeting Room as per schedule
   Sort by start time
   Create a min heap to store end time.

   Traverse through all meetings:
   If meeting start time is more than end time of active meeting.
   Remove active meeting
   Add the current meeting to active meeting.

   Active meeting always contains parralley ongoing meetings

2. Distribute Candy

   Give each child more candies than their neighbor if heir rating is higher
   Create a prefix and suffix array.

```
def candy(self, A):
        left, right = [1] * len(A), [1] * len(A)
        for i in range(1, len(A)):
            inv = len(A) - i - 1
            if A[i] > A[i-1]:        left[i] = left[i-1] + 1
            if A[inv] > A[inv + 1]:  right[inv] = right[inv+1] + 1
        candies = []
        for i in range(len(A)):   candies.append( max(left[i], right[i])
)
        return sum(candies)
```

3. Seats

   Create the seats array.
   Find the median.

   For each element in seats position array:
   Start = Seats[i]
   End = PositionInOriginalArray - MedianIndex + i
   Seats += end - start

```
def seats(self, A):
    seats = [i for i in range(len(A)) if A[i] == 'x']
    med = len(seats) // 2
    moves = 0
    for i in range(len(seats)):
        start, end  = seats[i], seats[med] - med + i
        moves += abs( end - start )
    return moves % 10000003
```

4. Assign Mice to Holes

Heapify A and B
While A is not empty:
X = max(X, abs(heapop(A) - heappop(B))

5. Majority Element

Apply similar to N/3 Repeat Numbers.
Cancel out occurences of elements.

```
 ele, count = A[0], 1
for x in A[1:]:
    if x != ele:    count -= 1
    else:           count += 1
    if count == 0:  ele, count = x, 1
return ele
```

6. Gas Stations

```
def canCompleteCircuit(self, gas, cost):
        if sum(gas) - sum(cost) < 0: return -1
        tank = 0        # gas available in car till now
        start = 0       # Consider first gas station as starting point
        for i in range(len(gas)):
            tank += gas[i] - cost[i]
            if tank < 0:        # the car has deficit of petrol
                start = i+1     # change the starting point
                tank = 0        # make the current gas to 0,
                                #
                            as we will be starting again from next statio
                            n
        return start
```

BST Traversal

20 September 2021 9:30 PM

1. Valid BST from Preorder

   Idea : Use stack to always store the largers elements.

   Algorithm:
   For each x in A:
   Check if x is less than root. If yes return 0
   Remove smaller element from stack and make it root.
   Add current element to x

```
def solve(self, A):
        stack, root = [], -sys.maxsize
        for x in A:
            if x < root:                    return 0
            while stack and x > stack[-1]:  root = stack.pop()
            stack.append(x)
        return 1
```

   Root always contains the value left maximum. Use stack to track the root.
   While traversing, the mid node will be in stack, while travelling left root value will be the unchanged. Only when travelling right, root will be updated to contain the popped element.

2. Kth smallest element in Tree

   In order traversal, update a counter instead of print statement

3. 2 Sum Binary Tree

   PreOrder Traversal, Add Sum - root.val to set instead of print

4. BST Iterator

Init	Stack = [] Apply Fill_Left() to root
hasNext	Check if elements exist in stack
Next	Pop the top eleemnt from stack Apply Fill_Left() to top.right Return top.value
Fill_Left	While root: Add root to stack Root = root.left

5. Recover Binary Search Tree

   Do a inorder traversal
   Keep check of prev, first, last

Instead of print, Check if current val is less than previous value. If yes, it means a smaller element is after larger element.
Therefore add the prev index to first and current place to last. Also, last can be later element, so last should be updated to next nodes also.

Inorder	

```
traverse(A.left)
if prev and A.val < prev:
    if not first:    first = prev
    last = A.val
prev = A.val
traverse(A.right)
```

Trie

21 September 2021 11:21 AM

```
class TrieNode:
    def __init__(self):
        self.child = [None] * 26
        self.end = False

class Trie:
    def __init__(self):
        self.root = TrieNode()
    def insert(self, key):
        temp = self.root
        for x in key:
            if temp.child[ ord(x) - 97 ] == None:
                temp.child[ ord(x) - 97 ] = TrieNode()
            temp = temp.child[ ord(x) - 97 ]
        temp.end = True
    def find(self, key):
        temp = self.root
        for x in key:
            if temp.child[  ord(x) - 97 ] == None:
                return 0
            temp = temp.child[ ord(x) - 97 ]
        return int(temp.end)
```

1. Hotel Reviews

   Sort with respect to good words
   Make trie of good words
   Sort with respect to count in trie

2. Shortest Unique Prefix

   Easy, make a dictionary, store the prefix count + 1 for each word found.
   Run again if count is 1, prefix is that substring for that word

   Same for trie. Build the trie. Add aditional information about child count for each childs.

   Whenever visiting the child, update it count by 1 in insert function.

3. Xor between two arrays

   To find element with maximum xor, given binary string.
   For each element, search the element in trie after that,
   Insert the inverse of the element in the trie.

```
    def insert(self, key):
        temp = self.root
        for i in range(31, -1, -1):
            bit = (key >> i) & 1
            if temp.child[bit] == None:
                temp.child[bit] = TrieNode()
```

```
            temp = temp.child[bit]

    def find_max(self, key):
        temp, res = self.root, 0
        for i in range(31, -1, -1):
            index = int((key >> i) & 1 == 0)
            res = res * 2
            if temp.child[index]:
                res += 1
                temp = temp.child[index]
            else:
                temp = temp.child[ 1 ^ index ]
        return res
```

Simple Tree Operations

21 September 2021 11:50 AM

1. Path to given node.

   Traverse and return path rcrusively

2. Remove hald nodes

   Remove nodes with single child.
   Pass the information upward. Let the node be as it is if leftvalue and rightvalue == 1

```
def check(self, A):
        if A == None:            return None
        A.left = self.check(A.left)
        A.right = self.check(A.right)
        if A.left == None and A.right != None:            return A.right
        if A.right == None and A.left != None:            return A.left
        return A
```

3. Balanced Binary Tree

   Return max of left and right tree height. If at any node, left height and right height diff is more than 1 ,
   Return False

4. Maximum Edge Removal

   Function depth return removal_count and total nodes.
   For each edge in node x:
   Add node counter and remove counter to total
   If nodes in that edge are in even number:
   Increase remove_counter
   Return total_remove_counter and total_nodes + 1 (for current node)

   If no edge exist: remove 0, 1

5.

2 Trees

21 September 2021 12:11 PM

1. Merge Two Binary Tree

   Travel the nodes together. Add the values of both the values
   Cases:
   If both left are None : Travel a.left, b.left
   If a.left = None : Make a.left = b.left

   Same for right side

2. Symmetric Binary Tree

   Travel in opposite direction.
   If both are none return 1, if one is none return 0.
   If either values are unequal or left or right return 0 return 0 else 1

3. Identical Binary tree

   Travel in same direction.

```
def equal(self, A, B):
        if A == None and B == None:           return 1
        if A == None or B == None:            return 0
        return int (
                  A.val == B.val
                  and self.equal(A.left, B.left)
                  and self.equal(A.right, B.right) )
```

Traversal

21 September 2021 12:19 PM

1. Vertical Order Traversal

   Keep a map with level value.
   Queue = [(start, 0)]
   Apply level order traversal,
   Explore nodes,
   Add node, level-1 for left node
   Add node, level+1 for right node

   Store the map in sorted order.

2. Diagonal Order Traversal

   Keep a map with level value.
   Queue = [(start, 0)]
   Apply level order traversal,
   Explore nodes,
   Add node, level+1 for left node
   Add node, level for right node

   Preorder traversal with adding into map with key as slope the value

Level Order Traversal

21 September 2021 12:30 PM

1. Right View of Binary Tree

   Level Order Traversal, add last value in each queue in the answer.

2. Cousins In Binary Tree

   Level Order Traversal,
   If B in left or right:
   Break
   Explore all other nodes in that level.
   If flag becomes one, the nodes in the queue are siblings.

3. Reverse Level Order

   Easy as Level Order Traversal

4. Zig Zag Traversal

   Easy as level order Traversal

5. Populate Next Right Pointers Tree

   Easy as level order Traversal

Root To Leaf

21 September 2021 12:49 PM

1. Burn A Tree

   Function Run(Takes A Tree, Node B) : Return f (State to check if B is found) and d cost

```
def run(self, A, B):
    if A == None:            return False, 0
    if A.val == B:           return True, 0
    f1, d1 = self.run( A.left, B )
    f2, d2 = self.run( A.right, B )
    if f1 == False and f2 == False:     return False, max(d1, d2) + 1
    self.out = max( self.out, d2 + d1 + 1)
    return True, f1*d1 + f2*d2 + 1
```

2. Sum Root To leaf

   Recursively pass numbers till now. At leaf add the number

3. Path Sum

   Same as 2

4. Min Depth of Binary Tree

   Return 1 if both left and right are None
   Check for left path and right path. Return minimum

5. Roots to leaf paths with sum

   Same as 2 track path.

Tree Construction

21 September 2021 1:27 PM

1. Inorder Traversal of Cartesian Tree

   At each iteration.
   Make root with max in Array
   Make left subtree with left values
   Make right subtree with right value

```
def buildTree(self, A):
    if not A:            return None
    i = A.index(max(A))
    root = TreeNode(A[i])
    root.left = self.buildTree(A[:i])
    root.right = self.buildTree(A[i+1:])
    return root
```

2. Sorted Array to Balanced BST

   Same as above. Make mid as root node to generate balanced bst.

3. Construct Binary tree from inorder and preorder.

   Set preIndex = 0
   Shift index everytime you make a node.
   Root index i = inOrder.index(preOrder[preIndex])
   Make root with index i

```
def builder(self, inorder):
    if not inorder:
        return None
    i = inorder.index( self.preorder[self.preindex] )
    self.preindex += 1
    root = TreeNode(inorder[i])
    root.left = self.builder(inorder[:i])
    root.right = self.builder(inorder[i+1:])
    return root
```

4. Binary Tree From Inorder an PostOrder

   Same as above, travel backwards
   postIndex = len(A) - 1
   Root index i = inOrder.index(postOrder[postIndex])
   Decrease preIndex

Inplace Change

21 September 2021 1:35 PM

1. Invert The Binary Tree

   Easy, Travel , swap left and right move left and right

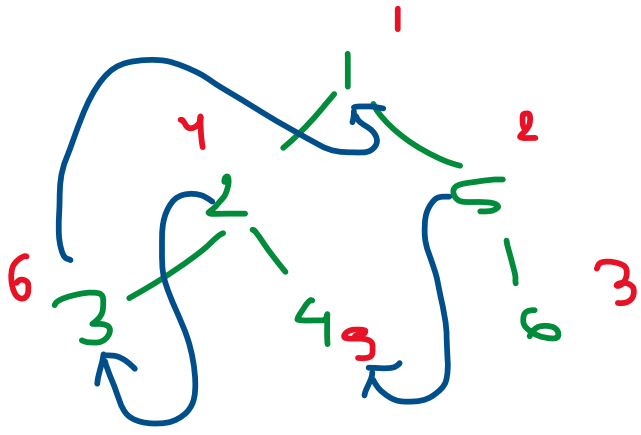
2. Lowest Common Ancestor

   Easy: Get Path for A and B. Find lowest common.
   Complex:

```
def findLCA(root, n1, n2):
    if root is None:
        return [0, 0], None
    lcheck, lnode   = findLCA(root.left, n1, n2)
    rcheck, rnode   = findLCA(root.right, n1, n2)
    comb            = [lcheck[0] or rcheck[0], lcheck[1] or rcheck[1]]
    if lcheck == [1, 1]:      # If both present in left subtree
        return [1, 1], lnode
    if rcheck == [1, 1]:      # If both present in right subtree
        return [1, 1], rnode
    if comb == [1, 1]:        # If each present in each subtree, current root is answer
        return [1, 1], root

    if n1 == n2 and root.val == n1:   # If n1 = n2 == current root
        return [1, 1], root
    if root.val == n1:        # If root is n1 , search n2 in subtrees
        if lcheck[1] == 1 or rcheck[1] == 1:
            return [1, 1], root
        return [1, 0], None

    if root.val == n2:        # If root is n2, search n1 in subtrees
        if lcheck[0] == 1 or rcheck[0] == 1:
            return [1, 1], root
        return [0, 1], None

    return comb, None         # return found elements.
```

3. Linked List Tree

   Keep moving right, while returning return current node as prev.
   This prev can be used to set as next node for rightest node in the left.




```
def flatten(A,prev=None):
    if not A:        return prev
    prev    =   flatten(A.right, prev)
    prev    =   flatten(A.left, prev)
    A.right, A.left, prev =   prev, None, A
    return prev
```

4. Order of People Heights

   Easy Solution:
   Take (height, index) as tuple into a list. Reverse Sort with respect to heights.
   Traverse the list, add the heights to their respective index.
   In the sorted list, all larger elements are already filled, so index will tell the current position to insert.

```
def order(self, A, B):
    l, out = [ (A[i], B[i]) for i in range(len(A))], []
    l = sorted(l, reverse=True)
    for val, inf in l:         out.insert(inf, val)
    return out
```

   Segment Tree : Can be used to store sum or max

   Initiation : For n nodes, we require n array of size 2*n - 1
   The left child = 2*p
   The right child = 2*p + 1

   Build Tree :

Add all element from nth position to right

For i in range(N):	Tree[n+i] = A[i]
For i in range(N-1, -1, -1):	Tree[i] = Tree[2*i] + Tree[2*i + 1]

Update Tree :

Set tree[n+ index] = value
Set i = index + n
While i > 1:
    Tree[i//2] = Tree[i] + Tree[i+1]
    Update i = i // 2

Query Tree:

Sum = 0
L, R = L + N, R + N
While L < R:

If L & 1:	Sum += Tree[L] L += 1
If R & 1:	R -= 1 Sum += Tree[R]

    L, R = L//2, R//2
Return sum

Segment Tree Solution :

s

2D Strings DP

21 September 2021 3:49 PM

1. Longest Common Subsequence

   $if\ \ A_i == B_j\ :\ \ dp_{i,j} = dp_{i-1,j-1} + 1$
   $else:\ \ dp_{i,j} = \max(\ dp_{i-1,j}\ , dp_{i,j-1}\ )$

2. Longest Palindromic Subsequence

   Longest Common Subsequence with A, A.reverse()

3. Edit Distance

   Minimum move to Make string A equal to B.
   In the main case: if character at i and j index match , try with next item
   If they do not match, try removing character (left) , try adding character (up), try update(leftup)

   $if\ i\ and\ j == 0:\ \ dp_{i,j} = 0$
   $else\ if\ i == 0:\ \ dp_{i,j} = dp_{i,j-1} + 1$
   $else\ if\ j == 0:\ \ dp_{i,j} = dp_{i-1,j} + 1$
   Else
   $If\ A_i == B_j: dp_{i,j} = dp_{i-1,j-1}$
   $Else:\ \ dp_{i,j} = 1 + \min(\ left, up, leftup)$

4. Repeating Sub Sequence

   Longest common subsequence on one diagonal matrix (upper diagonal matrix for i < j)
   The dp[R-1][C] will contain the value.

5. Distinct Subsequences

   Idea:
   First Column = 0 : As empty string cant have any subsequence
   First Row = 1 : As empty string is subsequence at all points.
   If Ai matches Bj : Count without Ai dp[i-1][j] + Count without Ai and Bj dp[i-1][j-1]
   Else : Count without Ai dp[i-1][j]

6. Scramble String

   ```
   isScramble(A, B)
       If A == B: return 1
       If len(A) != len(B) : return 0
       For i in range( len(A) - 1 ):
           If  leftElement(A, i) == leftElement(B, i):
               Check isScramble( A[:i+1], B[:i+1] ) and isScramble( A[i+1:], B[i+1:] )
           If leftElement(A, i) == rightElement(B, len(B) - i)
               Check isScramble( A[:i+1], B[len(B) - i:] ) and isScramble( A[i+1:], B[:len(B) - i ] )
   ```

7. Regular Expression Match

   For '*' in pattern: Check all possible combination with and without taking the next elements
   For equal character or '?' in pattern: Check with next string and pattern

If pattern.len - count(*) > string.len: Then there exist extra elements in pattern so false
Dp = [1] * [0] * len(string)
For c in pattern:
    If c == '*':
        Dp[i+1] = dp[i+1] or dp[i] for i in [0, len(string)]
    Else:
        Dp[i+1] = dp[i] and (c == string[i] or c == '?') for i in [len(string)-1 0]
    Dp[0] = dp[0] and c == '*'
Return 1 if dp[-1] else 0

Keep single array of size len(string) + 1.
For every char in pattern:
    Update the whole array.
    If c == *: make all value next to it true
    Else: Make all values which matches pattern[0] true whose previous value was true

8. Regular Expression 2

   Easier, Apply recursive implementation

9. Interleaving Strings

```
def rec(self, A, B, C):
        a, b, c = len(A), len(B), len(C)
        if a == 0 and b == 0 and c == 0:    return 1
        if a + b != c: return 0
        flag = 0
        if a > 0 and A[0] == C[0]:    flag = flag or self.rec(A[1:], B, C[1:])
        if b > 0 and B[0] == C[0]:    flag = flag or self.rec(A, B[1:], C[1:])
        return flag
```

Simple Array DP

21 September 2021 7:15 PM

1. Length of Longest Subsequence

   Longest subsequence first increasing then decreasing

```
def longestSubsequenceLength(self, A):
        inc, dec = [1] * len(A), [1] * len(A)
        for i in range(1, len(A)):
            for j in range(i):
                if( A[i] > A[j] and inc[i] < inc[j] + 1 ):
                    inc[i] = inc[j] + 1

            revi = len(A) - i - 1
            for j in range(len(A)-1, revi, -1):
                if( A[revi] > A[j] and dec[revi] < dec[j] + 1 ):
                    dec[revi] = dec[j] + 1
        maxi = 0
        for i in range(len(A)):       maxi = max( maxi, (inc[i] + dec[i]
- 1))
        return maxi
```

2. Smallest Subsequence with given primes

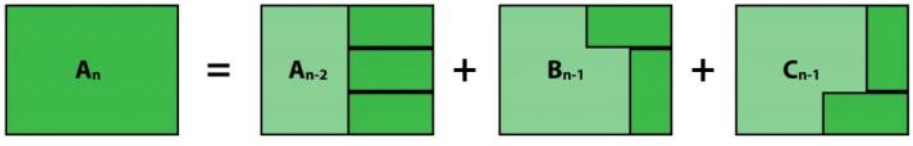
   Return array of size D denoting first D integers having A, B, C or combination of these as their prime factors.
   Out = [1]
   Ai, Bi, Ci = 0, 0, 0
   While len(out) <= D:
   Minele = min(out[ai] * A, out[bi] * B, out[ci] * C)
   Out.append(minele)
   If out[ai] * A == minele: ai += 1
   If out[bi] * B == minele: bi += 1
   If out[ci] * C == minele: ci += 1
   Return out

3. Largest Area of Rectangle with Permutations

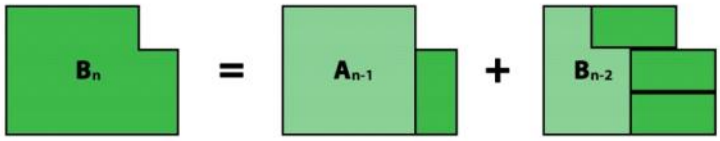
   a. Calculate consecutive 1's in every column
   b. Reverse sort each row. (Reverse Sorting will make every value in reverse order). Left of each value will always be greater
   c. Traverse each row and check for max area. Area = Column Number * Histogram[i][j] At each point column number and consecutive number of 1's in that point is the area. Since the row is reverse sorted, previous row must have equal or higher number of 1's in the left

4. Tilling with Dominoes

   3xA board with 2x1 dominoes.



$A_n = A_{n-2} + 2\,B_{n-1}$ *and* $A_0 = 1, A_1 = 0$



$B_n = A_{n-1} + B_{n-2}$ *and* $B_0 = 0, B_1 = 1$

5. Paint Houses

   Dp[i][j] = min(dp[i-1][a] + A[i][j] , dp[i-1][b] + A[i][j])
   Where a and b are index other than j from (0, 1, 2)

6. Ways to Decode

   Return valid(A[1:]) + valid(A[2:])

7. Stairs
   Can only take 1 or 2 steps. For each n, count the ways to reach n-2 and n-1 and add them.

   A = [1, 1]
   Then fib: A.append(A[-1] + A[-2])

8. LIS

   At each step, increase the count by 1 with previous max element which is less than current element

   Dp = [1] * len(A)
   For i in range(1, A.len):
   For j in range(0, i):
   If Ai > Aj and dpi < dpj + 1:
   dpi = dpj + 1
   Return max(dp)

9. Intersecting Chords in A Circle

   Catalan Number

   At any time, divide the remaining points into two sets, no point in S1 joins S2.
   $Ways(N) = \sum_{i}^{N-1} Ways(i) * Ways(N - 1 - i)$
   Ways(0) = 1, Ways(1) = 1

10.

Derived DP

21 September 2021 9:33 PM

1. Chain of Pairs

   Check by Ignoring current element.
   If maxLimit < current start: Try to move forward by adding current element.
   Return maxPossible

2. Max Sum Without Adjacent Elements

   At each index i,
   $dp[i] = \max(dp[i-1], \max(A[i], B[i]) + dp[i-2])$
   Find max element at ith position if we consider previous index or previours to previous index

3. Merge Elements

   Dp[i][j] contain the cost to merge all elements from index i to j.
   Set dp[i][j] and merge cost = INT_MAX, sum(A[i:j+1])
   For index in range(i+1, j+1):
   Dp[i][j] = min(prev, merge_cost + merge(i, index-1, A) + merge(index, j, A)

Knapsack

21 September 2021 9:47 PM

1. Flip Array

   Make an dp array of size sum
   For n in A:
   For j in [sum//2, n]:
   Dp[j] = min(dp[j], dp[j-i] + 1)
   # This creates a dp array which stores all possible sum in the dp. Dp also contains
   # the minimum number of elements to make the sumi
   # Now we need to check the sum nearest to the middle.

2. Tushar Birthday Party

   Algorithm:
   a. Find max_eating_capacity.
   b. For cap in [1, max_cap]:
      i. For dish in [1, dishes]:
         1) If capacity >= Dishwt[dish]:
            a) Dp[cap][dish] = min(dp[cap][dish-1] , dp[cap - dishwt][dish] + Cost[dish]
         2) Else:
            a) Dp[cap][dish] = dp[cap][dish-1]

   For each capacity, find the minium number of dish which can fill it. Above algorithm will fill
   The minimum dishes for each capacity.

   Then find the number of dishes required for each friend.

3. 0-1 Knapsack

```
def knapSack(W, wt, val, n):
    dp = [[0 for x in range(W+1)] for x in range(n+1)]
    # Build table dp[][] in bottom up manner
    for i in range(n+1):
        for w in range(W+1):
            if i==0 or w==0:
                dp[i][w] = 0
            elif wt[i-1] <= w:
                dp[i][w] = max(val[i-1] + dp[i-1][w-wt[i-1]],  dp[i-1][w])
            else:
                dp[i][w] = dp[i-1][w]
```

   Profit of zero capacity or no element = 0 (Row and column)
   If weight of ith element is more than capacity,
   Then max(profit with one less carrying capacity, cost of ith element + profit with carrying capacity
   Decreased by wt of ith element)
   For each capacty i, check if adding object w can increase the profit.

4. Equal Average Partition

Divide the array in two parts such that both parts are equal

Since s1/ n1 = s2 / n2 = s / n
-> s1 = s * n1 / n

Find n1 such that s1 is integer => s* n1 % n == 0
For those n1, apply subset sum with (0, sumCount, elementCount).
If subset exist. Make sorted disjoint sets. Return it.

Aditional Problems

21 September 2021 11:09 PM

1. Best Time to buy and sell stocks 2.

   Buy one sell one.
   As soon as you get a higher stock than purchased one. Add it to profit, then update the purchase to current stock

DP OPTIMIZED BACKTRACK

1. Word Break 2

   Given a string and a dictionary, find possible sentences in the string.

   Solution:
   Build the trie dictionary

```
def rec(self, A, N, res):
        for i in range(1, N+1):
            if self.trie.search(A[:i]):
                if i == N:          self.ans.append(res + A[:i])
                self.rec(A[i:], N-i, res + A[:i] + ' ')
```

   For every matching word in string A, recurse with string after matching word.

2. Word Break 1

   Check if string can be segmented to words in dictionary. Same as above. Dont build the string

3. Palindrome Partitioning

```
def minCut(self, s):
    if s == s[::-1]:       return 0
    else:
        x = sys.maxsize
        for i in range(1, len(s)):
            if s[:i] == s[:i][::-1]:
                x = min(x, 1 + self.minCut(s[i:]))
        return x
```

   Test for every possible segmenet if it is a palindrome.

4. Count Permutations of BST

   a. Build Combination (N)
      Create 2D matrix Using $nCr(n,r) = nCr(n-1,r-1) + nCr(n-1,r)$
   b. DP = [A][B] A is nodes, B is height
   c. DP[0][0], DP[1][0] = 1, 1
   d. For node in [2:A+1]
      i. For ht in [1:B+1]
         1) Total = 0
         2) For i in [0: node]
            a) L, r = i, nodes - i - 1
            b) Curr = 0
            c) For j in [0, ht-1]:
               i) Curr += DP[L][j] + DP[r][ht-1]
               ii) Curr += DP[R][j] + DP[L][ht-1]
            d) Curr += DP[L][ht-1] + DP[r][ht-1]
            e) Total += curr * nCr(L + r, r)
         3) DP[nodes][ht] = total
   e. Return DP[A][B]

$$F(N) = \sum_{x=0}^{N-1} {}^{N-1}C_x \, F(x)\, F(N-1-x)$$

5. Unique Binary Trees

   Find structurally unique BST that ca store 1...A

```
def numTrees(self, A):
    dp = [1, 1] + [0] * A
    for i in range(2, A+1):
        for j in range(0, i):
            dp[i] += dp[j] * dp[i-j-1]
    return dp[A]
```

$$F(N) = \sum_{x=0}^{N-1} f(x)\, f(N-1-x)$$

   UBT(0), UBT(1) = 1, 1

Greedy Or DP

21 September 2021 11:48 PM

1. Tushar Birthday Bombs
   Maximize total number of kicks with total sum of kicks strenght <= Bearing Limit.

   Calculte maximum hits. Keep the elements in left less than limit.
   Keep checking if using the element gives same number of hits as maximum hits. Then replace the element.

   Algorithm():
- min_strength , min_index, max_hits = min(B), index(min), A // min_st
- Store the hit strength less than limit upto the min_index

- I, ans = []
- While i < len(S):
  - Max_hit , Curr_hit = A // min_strength, 1 + (limit - S[i]) // min_strenght
  - If limit >= S[i] and max_hit == cur_hit :

    Add the index to ans, decrease limit = limit - S[i]
- Return ans

2. Jump Game Array

   For index i in [0: len(A) - 1]
   If i > max_jump : return 0
   Max_ump = max(max_jump, i + A{i])
   Return 1

3. Min Jump Array

   Find the min index on left which brings to i.

Tricky

22 September 2021 9:23 AM

1. Longest AP

   Check in left, (max count where difference A[i] - A[j]) + 1 wherever exist.

```
sol = defaultdict(int)
n = len(a)
for i in range(n):
    for j in range(i):
        d = a[i] - a[j]
        sol[(i, d)] = max(sol[(i, d)], 1 + sol[(j, d)])
return max(sol.values(), default=0) + 1
```

   Store the difference with index value, so next time when checking of difference with that element, if difference match, increase the count by 1

2. N digits numbers with digit sum S

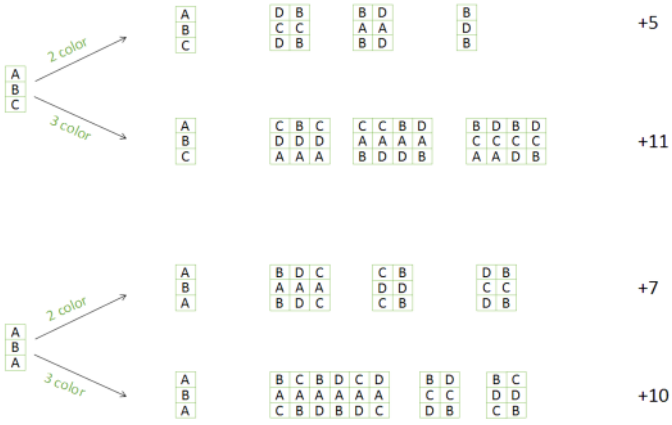
   Recurse start with (1, 9)
   If Number Limit becomes 0, return if Sum is Zero.
   If Sum becomes 0, return 1 : ie possible
   Else, check with all (0, 9) digits

3. Shortest Common Superstring

   Bit Masking with DP
   Idea : Pick order for strings in S and construct superstring : Naive
   Pick highest overlap strings and merge them : Greedy

   Untill the list is left with one element:
   Find the pair with highest overlap,
   Merge the strings in the list

4. Ways to Color 3xN board




```
def solve(self, A):
    c3 = 24
    c2 = 12
    for i in range(A-1):
        c3, c2 = ( 11*c3 + 10*c2 ) % 1000000007, ( 5*c3 + 7*c2 )
    return (c3+c2) % 1000000007
```

5. Kth manhattan Distance Neighborhood

For each $M_{i,j}$ find $M_{p,q}$ such that $|i-p| + |j-q| \leq K$
Run K times:
Update $M_{i,j}$ with max neighbor.

6. Best Time to buy and Sell Stocks atmost B times.

dp[i][j] = maximum profit from most i transactions using prices[0..j]
A transaction is defined as one buy + sell.
On Day j :
a. Do nothing, proft is same : dp[i][j] = dp[i][j-1]
b. Sell the stock j, To sell stock j, must buy it on
~t : [0, ..., j-1] max(prices[j] - prices[t] + dp[i-1][t-1]) buying on day t and selling on day j

Time complexity of this approach is O(n^2 * k).

In order to reduce it to O(n*k), we must find
~t : [0, ..., j-1] max(prices[j] - prices[t] + dp[i-1][t-1]) in constant time

If you see carefully,
~t : [0, ..., j-1] max(prices[j] - prices[t] + dp[i-1][t-1]) is same as
prices[j] + ~t : [0, ..., j-1] max(dp[i-1][t-1] - prices[t])

Track the max till T_th column.

```
def solve(self, price, B):
    B = min(len(price), B)
    profit = [ [0] * (len(price) + 1) for i in range(B + 1)]
    for i in range(1, B + 1):
        maxdiff = -sys.maxsize
        for j in range(1, len(price) ):
            maxdiff = max( maxdiff, profit[i-1][j-1] - price[j-1] )
            profit[i][j] = max( profit[i][j-1], price[j] + maxdiff)
    return profit[B][len(price)-1]
```

7. Coins in a Line

Execute Smarty,
If turn is yours, you try to maximize the coins,
Else minimize the coins.

```
def pick(self, l, r, turn):
    if l == r:          return 0
    t1 = self.pick(l+1, r, 1 ^ turn)
    t2 = self.pick(l, r-1, 1 ^ turn)
    if turn == 1:           return max(t1 + self.A[l], t2 + self.A[r])
    return min(t1, t2)
```

8. Evaluate Expression to True

Add all characters in one array, all symbols in one array.
Travel with gap of 2 character at once to 3 upto all character at once.
T[i][j], F[i][j] will contain true ways and false ways from i -- j

For each gap and with each character.

```
tt, ff, tf, ft = T[i][k] * T[k+1][j], F[i][k] * F[k+1][j], T[i][k] * F[k+1][j], F[i][k] * T[k+1][j]
                    if ope[k] == '|':
                        T[i][j] +=  tf + ft + tt
                        F[i][j] +=  ff
                    if ope[k] == '&':
                        T[i][j] +=  tt
                        F[i][j] +=  tf + ft + ff
                    if ope[k] == '^':
                        T[i][j] +=  tf + ft
                        F[i][j] +=  tt + ff
```

9. Egg Drop Problem

Drop egg from floor x:
   a. Egg break : Check from remaining eggs and floor x - 1
   b. Egg survives : Check from all eggs and floor total - x

Take maximum of above two cases : as we need to minimize the problem

Iterative Solution:

```
Dp[i][j]  to store minimum number of trials for i eggs and j floors.
Dp[i][1], dp[i][0] = 1, 0
Dp[1][j] = j

For index i, j:
    For x -> [1, j+1]:
        Res = 1 + max( dp[i-1][x-1] , dp[i][j-x] )
    Dp[i][j] = min Res
```

Easiest Solution:

```
L, R = 1, B
Apply  binary search:
    If  sumBinaryCoeff( mid, A)  < B:
        L = Mid + 1
    Else:
        R = Mid
Return L
```

$$SumBinaryCoeff(\ N, R\ ) = \sum_{i=1}^{i=R} nCr(\ N, i\ )$$

Max Floor(x trials, n eggs) = MaxFloor(x-1 trials, n-1 eggs) + MaxFloor(x-1 trials, n eggs) + 1
MaxFloor(0, n) = 0
MaxFloor(x, 0) = 0

We need to find x such that
MaxFloor(x trials, n eggs) >= K floors
Above relation solves to :

$$\text{Maxfloor} = \sum_{i=1}^{N} nCr(\ x, i\ )$$

10. Best Time to buy and sell stocks 3

Only two transactions are allowed.

```
Dp  = 0 * [stocks + 1] [ 2 + 1]
For i in range(1, 3):
    Maxdiff = INT_MIN
    For j in range(1, stocks):
        Maxdiff = max( maxdiff, dp[ i - 1 ][ j - 1 ] - stocks[ j - 1 ])
        Dp[ i ] [ j ]  = max( dp[ i ][ j - 1] ,  stocks[j] + maxdiff )

Return dp[2][ stocks]
```

11. Longest Valid Parenthesis

Dp[i] = longest set of parenthesis ending at i

If s[i] == '(':	dp[i] = 0

If s[i] == ')':	If s[i-1] == '(':
	  Dp[i] = dp[i-2] + 2
	Elif s[i-1] == ')' and s[i-dp[i-1]-1] == '(' :
	  Dp[i] = dp[i-1] + 2 + dp[i-dp[i-1]=2]

1) Create an empty stack and push -1 to it.
   The first element of the stack is used
   to provide a base for the next valid string.
2) Initialize result as 0.

3) Add index of '(' to stack.
2) If str[i] == ')',

    We need to remove its equivalent '(' for stack if it exists.

   a) Pop an item from the stack (Most of the
      time an opening bracket)

    If there is still an element in stack, it is the left valid limit uptill now.

    I - Stack[-1] is the current limit/

   b) If the stack is not empty, then find the
      length of current valid substring by taking
      the difference between the current index and
      top of the stack. If current length is more
      than the result, then update the result.

    If stack is empty, means the string in the left is invalid totally,

    Add index of current ')' element to mark the start of valid index as the left is not
    valid anymore.

   c) If the stack is empty, push the current index
      as a base for the next valid substring.

3) Return result.

Another Approach

Tree DP

22 September 2021 1:04 PM

1. Max Edge Queries

   Find maximum weighted edge in a simple path from u to v.

   Binary Lifting to pre compute the maximum weighted edge from every other node at a distance of 2^i

   INIT:
   - Make adjaceny list
   - Dp[i][j] store parent of j at 2^i
   - Max[i][j] store the maximum edge from node j to parent of this node at 2^i

   DFS:
   - Update level of each node
   - Update parent of each node in dp
   - Update weight of each node in mx

   Compute Ancestor
   - For j in range(1, maxLevel):
     - For i in range(1, N+1):
       - T = dp[i][j-1]
       - If t != -1:
         - Dp[i][j] = dp[t][j-1]
         - mx[i][j] = max(mx[i][j] , mx[t][j-1])

   Get LCA(u, v)
   - If level_u < level_v : swap u, v
   - Diff = level_u - level_v
   - Move to parent :
     - If diff and 1 << i is true;

   - For remaining level:
     - Move to parent together, and check if they match.
     - If both become equal, return it as parent

   Find_Max_Edge(u, v):
   - Lca = get_lca(u, v)
   - Travel in path from lca to u and lca to v to return max edge

2. Max Sum Path in binary Tree
   # Do a postorder traversal.
   At the end of traversal of node, you will get path from both subtrees, return maximum path cost.
   At every step update max value to contain all possible combinations c, l+c, r+c, l+r+c.

```
def traverse(self, R):
    if R == None: return 0
    l, c, r = self.traverse(R.left), R.val, self.traverse(R.right)
    self.ans = max( self.ans,   c,  l + c,   c + r,  l + c + r)
```

```
    return max(c,             l + c,            c + r)
```

Matrix DP

22 September 2021 1:50 PM

1. Kingdom War

   Since the element in left is less or equal to right and element in up is less or equal to down.

   $$dp[i,j] = A[i,j] + dp[i+1,j] + dp[i,j+1] - dp[i+1,j+1]$$

2. Maximum Path in Triangle

   Keep updating the paths according to the maximum above.

3. Maximum Size square SubMatrix

   $$dp[i,j] = \min(dp[i-1,j], dp[i,j-1], dp[i-1,j-1]) + 1$$

4. Increasing Path in Matrix
   Update the cost if possible in the matrix in row order.

```
if i < len(A) - 1  and A[i][j] < A[i+1][j]:
       dp[i+1][j] = max(dp[i+1][j], dp[i][j] + 1)
if j < len(A[0]) - 1 and A[i][j] < A[i][j+1]:
       dp[i][j+1] = max(dp[i][j+1], dp[i][j] + 1)
```

5. Minimum Difference Subsets

   Use knapsack to store the possible sums.
   Find the sum closest to sum//2. return sum - 2 * sumIndex

6. Subset Sum Problem

   Recurse with adding the element or without adding the element. If sum becomes 0, subset sum is possible.
   Top down:
   Knapsack type, Try storing the sums.
   If subset sum exist return 1.

7. Unique Paths in Grid

   If A[i][j] == 1 : leave as 0
   Else : dp[i][j] = dp[i-1][j] + dp[i][j-1]

8. Dungeon Princess

   Matrix Path, with life health dropper and potion to increase health

   Traverse matrix in reverse order.
   Set the bottom right value as 1 or (x if -x is there in the point)

   For every point on boundary:
   Dp[i][j] = max(1, nextBoundaryElement - current element)

   For every other elements:
   Dp[i][j] = max(1 , min(health below, health right) - current health)

9. Min Sum Path in Matrix

   Easy Problem. Sum of left and top elemnt

10. Min Sum Path in Triangle

    Easy as above.

11. Max Rectangle in Binary Matrix

    Make histogram at each row.
    Find maxhistogram area at each row.

    Use CleverStack to solve maxArea for histogram.

    CleverStack:
    Histogram Area:

    Making Left and Right Smaller index --
        While there is smaller element in stack than current element:
            RightSmaller [Make index of stack element] = current index
        LeftSmaller[currentIndex] = Index of stack top element

    Direct CleverStack:
        For each element in HISTORGRAM:
            If current element is greater or equal to top element:
                Add index to stack
            Else Current Element is smaller top element:
                Top_index = stack.pop()
                Curr_Area = A[top_index] * (current_index - stack[-1] - 1)
                Or Curr_Area = A[top_index] * current_index if stack is empty

        Empty the stack:
            Curr_area = A[top_index] * (max_i - stack[-1] - 1)
            Or Curr_Area = A[top_index] * max_i

        Here stack store immediate lower element index in the left.

12. Rod Cutting

    Recursively,
    try all points and divide and conquer left and right part separately.
    Return leftcost, left seq, rightmin, rightseq

13. Queen Attack

    For each queen:
    Travel in each direction:
        Keep increasing the count untill a border arrives or other queen is found.

Suffix/ Prefix DP

22 September 2021 4:18 PM

1. Sub Matrix with sum Zero

   For each start_row [0 : ROWS]
   M = [0] * N
   For i in [start_row : ROWS]
   For j in [0 : N]
   M[j] += A[i][j]
   Ans += countSubsetSumZero(M)
   Return ans

   Rows = [0, 1, 2, 3]
   Start Rows to End Rows =
   (0, 0), (0, 1), (0, 2), (0, 3)
   (1, 1), (1, 2), (1, 3)
   (2, 2), (2, 3)
   (3, 3)

   All possible combinations of rectangle. Count SubsetSumZero for each row.

2. Coins Sum Infinite

   Same as count sums existence (Knapsack).

   Try two ways, [Rec(A, Value - A[index], index) + Rec(A, Value, index-1)]
   If value become zero return 1,
   If value < 0 or index == -1 return 0

   Iterative Solution:

```
def coinchange2(self, A, B):
        dp = [0] * (B + 1)
        dp[0] = 1
        for coin in A:
            for i in range(1, B + 1):
                if coin <= i:
                    dp[i] = dp[i] + dp[i - coin]
        return dp[-1] % 1000007
```

3. Max Product Subarray

   Set min, max, res = A[0], A[0], A[0]
   For i in [1: len(A)] :
   Set min, max = max(A[i], max * A[i], min * A[i]), min(A[i], max * A[i], min * A[i])
   Res = max(res, max)
   Return res

4. Best Time to buy and Sell stocks.

   Update min_so_far at every index j.
   Update res = max(res, price[j] - min_so_far)

5. Arrange 2

   Try to put horses one by one in the stable. And recurse with next hourses and one less stable. Update cost if less than current cost.

Graph Traversal

22 September 2021 4:56 PM

1. Path in Directed Graph

   Do dfs or bfs. If node explored return 1 else 0

2. Water Flow

   Validity of BFS is if neighbor less than or equal to current: Allow the transfer

   Do BFS for boundaries for blue lake.
   Separetely Do BFS for boundaries for red lake
   Count the intersection

3. Stepping Numbers

   Do BFS, neighbor can be (current * 10 + current % 10 +- 1) if possible

4. Capture Regions on Board

   Capture all regions covered by 'x'.
   Change all '0' --> '_'.
   Start with all boundary and do bfs. Make the '_' to '0'.

   The nodes which are untouched or '_' are surrounded by 'x'. In the original matrix, mark the Untouched nodes to 'x'.

5. Word Search Board

   Do DFS to check if letter exist. Neighbor is if it is in range and character matches.

DFS

22 September 2021 5:19 PM

1. Path with good nodes

   DFS with goodcount.
   If lead and goodcount < limit:
   Increase resultCounter

2. Largest Distance between nodes of a tree

   a. DFS with start node. Capture the node farthest from root keeping a height counter.
   b. DFS with node recieved from above. The maximum height reached will be max dist between two nodes of tree

3. Cycles in Directed Graph

   DFS : if a visited + unclosed node is found, cycle is true.

4. Delete Edge.

   Maximize the product of (sum of weights of nodes in tree 1 , sum of wts of nodes in tree 2)

   Build the sum from leaf to root.
   Store the sum from branches and current node value as NODE WEIGHT.
   For each node in B:
   leftCost = min(NODEWT[edge[0]] , NODEWT[edge[1]])
   currVal = leftCost * (TotalCost - leftCost)
   Update maxVal

5. Two Teams

   Check if all people can be separeted into two teams.
   Do Graph Coloring.
   Apply colour 0 to root, for every edge switch colour. If the neighbor is found with same colour
   Return 0

BFS

22 September 2021 6:03 PM

1. Valid Path

   Prepare graph with circle conditions. Do graph search.

2. Snake Ladder

   Make a dictionary to update snakes and ladder.
   Do BFS while queue not empty:
   Add 0 to queue:
   Add next 6 neigbors into queue.
   Update with dictionary if snake or ladder is encountered.
   Repeat till 100 is not reached.

3. Region in Binary Matrix

   Multiple BFS with count of explored node by each starting point.
   Return maximum count of explored nodes.

4. Level Order

   Queue based Graph Traversal

5. Smallest Multiple with 0 and 1

   Start with queue = ['1']
   BFS, add '0' and '1' to end of each element of queue:
   The first to be found as multiple is the result. Do the modulus and set parent accordingly.

6. Min Cost Path

   Algorithm:
   1. Start with cell (0,0) push it into the doubly-ended queue and also make it distance as 0.
   2. Pop the front node from the queue.
   3. You can go to four directions from the popped node so if you go to a direction that is written on the cell then mark dist[neighbour]=dist[current]+1 else dist[neighbour]=dist[curr]
   4. If direction in which you move is not same as the direction written on the cell so this implies you have to incur a cost of 1 so push it to the back of queue else push it to the front of the queue.
   5. Repeat steps 2-4

   Push the forward nodes in front. Push the reverse going nodes in the back.

7. Permutation Swaps:

   Prepare the graph with good pairs.
   Create a matrix for each element's store source positions and destination position.
   Do DFS to check if path exist for all source to destination.

Graph Connectivity

22 September 2021 6:31 PM

1. Commutable Islands

   Minimal Spanning Tree

```
def find(self, i):
    if self.parent[i] == i:  return i
    return self.find( self.parent[i])

def union(self, x, y):
    xr, yr = self.find(x), self.find(y)
    if self.rank[xr] < self.rank[yr]:            self.parent[xr] = yr
    elif self.rank[yr] < self.rank[xr]:          self.parent[yr] = xr
    else:            self.parent[yr], self.rank[xr] = xr, self.rank[xr] +
1
```

```
Kruskal Or Prims Algorithm
Kruskal Union Find:
 a. Sort edge with respect to weight.
 b. Take an edge, Check if it does not make a cycle. Add to output.
    Increase edge count by 1. Union(u, v)
```

2. Possibility of finishing all courses given pre requisites

   Detect Cycle in list. If cycle exist it it not possible

3. Cycle in a undirected graph

   Cycle with more than two node.
   Provide a variable as parent.
   While exploring the neighbors, if the nbor is visited and parent != node :
   Return cycle exists.

4. Black Shapes

   Count connected componenets

Extra Problems

22 September 2021 6:52 PM

1. Convert Sorted List to BST

   Count the number of nodes.

   Work with inorder tranversal.
   Keep travelling left with half nodes,
   After returing from left side,
   Make the node with current head list.
   Keep travelling right with other half of the list.

```
def sortedListToBSTRecur(self, n) :
    if (n <= 0) :
        return None
    left = self.sortedListToBSTRecur( int(n/2) )
    root = TreeNode( self.head.val)
    root.left = left
    self.head = self.head.next
    root.right = self.sortedListToBSTRecur( n - int(n/2) - 1)
    return root

def sortedListToBST(self, A):
    self.head = A
    n, temp = 0, A
    while temp != None:            n, temp = n + 1, temp.next
    return self.sortedListToBSTRecur(n)
```

2. Clone Graph

   Hard Problem
   Traversal and create a copy of nodes and add it to dictionary
   Retraverse and make nodes as in original graph

Shortest Path

22 September 2021 7:00 PM

1. Sum of Fibonacci Number

   Build fibonacci number till A.
   Do coin change.

2. Knight on Chess Board

   Do a breath first search, if destination is reached count steps to reach there.

3. Useful Extra Edges

   Recursively, use the edge only once. Keep flag to store this information.
   Do dfs from src to dest as well as dest to src.

4. Word Ladder

   Shortest Transformation of Word A to B using dictionary C
   Store all characters at each position using the dictionary.
   Do BFS.
   For each position:
   Try each character other than itself and if word is valid:
   Add it to queue