## 1) Total no of paths in matrix if right and down move allowed

https://leetcode.com/problems/unique-paths/submissions/
https://leetcode.com/problems/unique-paths/discuss/22958/Math-solution-O(1)-space
https://leetcode.com/problems/unique-paths-ii/(with some obstacles)
Dp[i][j] = dp[i-1][j] + dp[i][j-1]

Minimum path sum in matrix

a) https://leetcode.com/problems/minimum-path-sum/
b) dp[i][j] = a[i][j] + min(dp[i-1][j], dp[i][j-1])

## 2) Min stair cost

https://leetcode.com/problems/min-cost-climbing-stairs/
Each stair has given cost find min cost to reach at top

```
Input: cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]
Output: 6
```

dp[i] = cost[i] + min(dp[i-1],dp[i-2])

## 3) Is subsequence

https://leetcode.com/problems/is-subsequence/

a) Using dp
   i) dp[i][j] = dp[i-1][j-1] if s[i] = t[j]
      = false else
      If one cell is true make entire row true

| s\t | h | a | x | b | c |
|---|---|---|---|---|---|
| a | F | T | T | T | T |
| b | F | F | F | T | T |
| c | F | F | F | F | T |

b) Using two pointer
   i) Scan s by i and inc j when s[i] == t[j]
      if j == t.length() found string

## 4) Robbing house

a) Each house has some value indicated by array element adjacent house can not be robbed find max amount that can be robbed

b) dp[ i ] = max( dp[ i - 1 ], value[ i ] + dp[ i - 2 ] )
   Don't rob current house Rob current house

c) Or using inc exc property

d) 2, 1, 1, 2 => (2 + 2 = 4)
   1, 2, 3 ,4 => (2 + 4 = 6)

e) Circular Robbing house
   i) https://leetcode.com/problems/house-robber-ii/submissions/
   ii) return max(robhouse(nums, 0, nums.size()-2),robhouse(nums, 1,nums.size()-1));

5) Longest valid Parentheses

https://leetcode.com/problems/longest-valid-parentheses/solution/

| ( | ) | ( | ( | ) | ) |
|---|---|---|---|---|---|
| 0 | 2 | 0 | 0 | 2 | 6 |

If s[ i ] == ‘ ( ‘
Dp[ i ] = 0
Else
\\s[ i ] = = ‘ ) ‘
If s[ i - 1 ] == ‘ ( ‘
S[ i ] = s[ i - 2 ] + 2
Else s[ i - 1 ] == ‘ ) ‘
if( s[ i - dp[ i - 1 ] - 1 ] == ' ( ')
dp[i] = dp[i-1] + 2 + dp[i-dp[i-1]-2];
else
dp[i] = 0;

6) Total no of unique binary search tree

https://leetcode.com/problems/unique-binary-search-trees/discuss/31666/DP-Solution-in-6-lines-with-explanation.-F(i-n)-G(i-1)-*-G(n-i)

Series : 1 2 5 14 42 132

G(n) = F(1, n) + F(2, n) + ... + F(n, n). // possible roots

G(n) = G(0) * G(n-1) + G(1) * G(n-2) + … + G(n-1) * G(0)

G(n) = E( G(x-1) * G(n-x) ) where 1<= x <= n

Eg : 1 2 3 4 5 6 if 3 i root left side 2 right side 3 element 1,2,3,4,5,6

```
G[0] = G[1] = 1;


  for(int i=2; i<=n; ++i) {
    for(int j=1; j<=i; ++j) {
      G[i] += G[j-1] * G[i-j];
    }
  }
  return G[n];
```

7) No of ones in binary representation of number
   a) f[i] = f[i >> 1] + (i & 1);
   b) Explaination.
   c) Take number X for example, 10011001.
   d) Divide it in 2 parts:
   e) <1>the last digit ( 1 or 0, which is " i&1 ", equivalent to " i%2 " )

f) <2>the other digits ( the number of 1, which is " f[i >> 1] ", equivalent to " f[i/2] " )

## 8) Edit distance problem

a) Min no of operation to convert word1 to word2
b) dp[i][j] = dp[i-1][j-1] if word1[i-1] = word2[j-1]
c) min( dp[i-1][j], dp[i][j-1],dp[i-1][j-1] ) + 1 else

https://leetcode.com/problems/edit-distance/discuss/25846/C%2B%2B-O(n)-space-DP

In above link 3 approaches to solve dp using O(n2) O(2n) and O(n)

## 9) 0/1 Knapsack Problem

a) https://practice.geeksforgeeks.org/problems/0-1-knapsack-problem/0
b) Here limited supply of item is given so

```
if (i==0 || w==0)
    K[i][w] = 0;
else if (wt[i-1] <= w)
      K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w]);
else
      K[i][w] = K[i-1][w];
```

## 10) Coin change problem

a) https://practice.geeksforgeeks.org/problems/coin-change/0

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 0 | 1 | 0 | 0 | 0 | 0 |
| 1 1 | 1 | 1 | 1 | 1 | 1 |
| 2 2 | 1 | 1 | 2 | 2 | 3 |
| 3 3 | 1 | 1 | 2 | 3 | 4 |

```
if(a[i-1] > j)
    dp[i][j] = dp[i-1][j];
else
    dp[i][j] = dp[i-1][j] + dp[i][j-a[i-1]];
```

b) https://leetcode.com/problems/coin-change/submissions/

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 0 | 0 | INF | INF | INF | INF |
| 1 1 | 0 | 1 | 2 | 3 | 4 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 2 | 0 | 1 | 1 | 2 | 2 |
| 3 | 3 | 0 | 1 | 2 | 1 | 2 |

if(a[i-1] > j)
  dp[i][j] = dp[i-1][j];
else
  dp[i][j] = min( dp[i-1][j] , dp[i][j-a[i-1]] + 1);  take INT = 99999 to avoid overflow

c) https://leetcode.com/problems/combination-sum-iv/
Find all combinations of numbers having sum = target
Use backtracking approach and covert it to top down and bottom up approach

## 11) Longest Inc subsequence nlogn

a) LCS

https://leetcode.com/problems/longest-increasing-subsequence/discuss/74848/9-lines-C%2B%2B-code-with-O(NlogN)-complexity

```
vector<int> res;
for(int i=0; i<nums.size(); i++) {
    int ind = lower_bound(res.begin(), res.end(), nums[i]) - res.begin();
    if(ind == res.size()) res.push_back(nums[i]);
    else res[ind] = nums[i];
}
return res.size();
```

b) Stack Book

https://cses.fi/problemset/task/1073/

```
  for(int i=0;i<n;i++){
    int index = upper_bound(dp.begin(), dp.end(), a[i]) - dp.begin();
    if(index == dp.size())
        dp.push_back(a[i]);
    else
        dp[index] = a[i];
}
cout << dp.size() <<en
```

c) Maximum nested doll possible

a. https://leetcode.com/problems/russian-doll-envelopes/
b. Sort by width asc and when equal des by height
c. Apply lis on height
d. Reason : (6, 1), (6,2), (6, 3) here if we don't sort by height will get ans = 3 so.

d) Maximum no of longest Increasing Subsequence

e. https://leetcode.com/problems/number-of-longest-increasing-subsequence/

```
if (nums[i] > nums[j]) {
   if (len[j]+1 > len[i]) {
        len[i] = len[j]+1;
        cnt[i] = cnt[j];
   }
   else if (len[j]+1 == len[i])
      cnt[i] += cnt[j];
```

## 12) Palindrome

a) Longest Palindromic subsequence

i) https://leetcode.com/problems/longest-palindromic-subsequence/submissions/

```
if(s[i] == s[j])
    dp[i][j] = dp[i+1][j-1] + 2;
else
    dp[i][j] = max(dp[i][j-1],dp[i+1][j]);
```

ii)

(1) A X B B A B

| | | | | | | |
|---|---|---|---|---|---|---|
| A | 1 | 1 | 1 | 2 | 4 | 4 |
| B | 0 | 1 | 1 | 2 | 2 | 3 |
| X | 0 | 0 | 1 | 2 | 2 | 3 |
| B | 0 | 0 | 0 | 1 | 1 | 3 |
| A | 0 | 0 | 0 | 0 | 1 | 1 |
| B | 0 | 0 | 0 | 0 | 0 | 1 |

b) Longest Palindromic Substring :

i) For each character traverse both side and find odd len palindrome

ii) Similarly traverse both side and find even len palindrome

iii) https://leetcode.com/problems/longest-palindromic-substring/

iv) If s[i] == s[j]

```
    dp[i][j] =dp[i+1][j-1]
Else
    dp[i][j] = false
```

| | A | B | A | A | B | A |
|---|---|---|---|---|---|---|
| A | 1 | 0 | 1 | 0 | 0 | 1 |
| B | | 1 | 0 | 0 | 1 | 0 |
| A | | | 1 | 1 | 0 | 0 |
| A | | | | 1 | 0 | 1 |
| B | | | | | 1 | 0 |
| A | | | | | | 1 |

c) Count palindromes

i) https://practice.geeksforgeeks.org/problems/count-palindrome-sub-strings-of-a-string/0

```
s = '@' + s + '#';
int n = s.size();
for(int mid = 1; mid <= n-1; mid++){
   i = mid-1, j = mid+1;

   while(s[i] == s[j])
      count++, i--, j++;

   i = mid-1, j = mid;
   while(s[i] == s[j])
      count++, i--, j++;
```

```
}
cout << count << endl;
```

d) Min insertion to make string palindrome
   i) https://www.geeksforgeeks.org/minimum-insertions-to-form-a-palindrome-dp-28/
   ii) table[l][h] = (str[l] == str[h]) ? table[l + 1][h - 1] : (min(table[l][h - 1], table[l + 1][h]) + 1);
   iii) Find longest palindromic subsequence. Ans = Total length-length of LPS

e) .https://www.geeksforgeeks.org/minimum-number-deletions-make-string-palindrome/
   i) table[l][h] = (str[l] == str[h]) ? table[l + 1][h - 1] : (min(table[l][h - 1], table[l + 1][h]) + 1);
   ii) Find longest palindromic subsequence. Ans = Total length-length of LPS

f) https://www.geeksforgeeks.org/find-if-string-is-k-palindrome-or-not/
   Find longest palindromic subsequence. Check length-length of LPS <= K

g) Count Pal substring in range i to j
   i) https://www.geeksforgeeks.org/count-of-palindromic-substrings-in-an-index-range/
   ii) https://www.geeksforgeeks.org/count-palindrome-sub-strings-string/

## 13) Longest Repeating Subsequence

a) https://www.geeksforgeeks.org/longest-repeating-subsequence/

b) Using dp

```
if (str[i-1] == str[j-1] && i != j)
    dp[i][j] =  1 + dp[i-1][j-1];
 else
    dp[i][j] = max(dp[i][j-1], dp[i-1][j]);
```

| | | a | b | b | a |
|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 |
| a | 0 | 0 | 1 | 1 | 1 |
| b | 0 | 1 | 1 | 1 | 1 |
| c | 0 | 1 | 1 | 1 | 2 |
| d | 0 | 1 | 1 | 2 | 2 |

c) Without using dp O(n) for check existence of repeating subsequence
   i) https://www.interviewbit.com/problems/repeating-subsequence/
   ii) If any char occurs more than 2 times return true
   Now make a string where each char repeats 2 times
   If this string is palindrome than then return 0
   Else return 1

## 14) Number of Distinct subsequences t in s

a) https://leetcode.com/problems/distinct-subsequences/

b)
```
if(t[i-1] == s[j-1])
    dp[i][j] = dp[i-1][j-1] + dp[i][j-1];
 else
    dp[i][j] = dp[i][j-1];
```

c)

| | | B | A | B | G | B | A | G |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| B | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 3 |
| A | 0 | 0 | 1 | 1 | 1 | 1 | 4 | 4 |
| G | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 5 |

## 15) Longest inc dec sequence

a) https://www.interviewbit.com/problems/length-of-longest-subsequence/

b)

i) `inc[i]` stores Longest Increasing subsequence ending with `A[i]`

ii) `dec[i]` stores Longest Decreasing subsequence ending with `A[i]`

iii) Now we need to find the maximum value of `(inc[i] + dec[i] - 1)`

## 16) Largest Rectangle with 1's , swapping of columns allowed

a) https://www.interviewbit.com/problems/largest-area-of-rectangle-with-permutations/

## 17) Decode Ways

a) https://leetcode.com/problems/decode-ways/submissions/

b)

```
if(x < 27)
    dp[i] = dp[i+1] + dp[i+2];
else
    dp[i] = dp[i+1];
```

## 18) Interleaving Strings

a) https://www.interviewbit.com/problems/interleaving-strings/

b)

```
if(i >= 0 && A[i] == C[k])
    a =  isInterleaving(A,B,C,i-1,j,k-1);

if(j >= 0 && B[j] == C[k])
    b = isInterleaving(A,B,C,i,j-1,k-1);

Return a || b
```

## 19) Wild card character

a) Use map <pair,bool> of dp to avoid memory limit errror

b) https://www.interviewbit.com/problems/regular-expression-match/ (both top down and bottom up)

c)

```
if(text[i-1] == pat[j-1] || pat[j-1] == '?')
        dp[i][j] = dp[i-1][j-1];
else if(pat[j-1] == '*')
        dp[i][j] = dp[i-1][j] || dp[i][j-1];
else
        dp[i][j] = false;
```

## 20) Regular Expression

a) https://www.interviewbit.com/problems/regular-expression-ii/ ( both top down and bottom up)

b) Top down approach

```
if(text[i-1] == pat[j-1] || pat[j-1] == '.')

        dp[i][j] = dp[i-1][j-1];

else if(pat[j-1] == '*')

{
```

```
if(pat[j-2] == text[i-1] || pat[j-2] == '.')
    dp[i][j] = dp[i][j-2] || dp[i-1][j-2] || dp[i-1][j];
Else
    dp[i][j] = dp[i][j-2];
}
```

## 21) Scramble String

a) https://www.interviewbit.com/problems/scramble-string/

b)

```
if(Scramble(A.substr(0,i),B.substr(0,i)) && Scramble(A.substr(i),B.substr(i)) ||
    Scramble(A.substr(0,i),B.substr(n-i)) && Scramble(A.substr(i), B.substr(0,n-i)))
    return mp[A+B] = true;
```

## 22) Longest Arithmetic Sequence

a) https://leetcode.com/problems/longest-arithmetic-sequence/submissions/

b) A,B,C are in AP then A = 2 B - C

i) dp[i][j] no of items in arithmetic series ends with a[i], a[j]

ii) int target = 2*a[i] - a[j];

iii) if(index.find(target) == index.end())

iv) dp[i][j] = 2;

v) else

vi) dp[i][j] = dp[index[target]][i] + 1;

## 23) Min palindrome partitioning

a) https://www.interviewbit.com/problems/palindrome-partitioning-ii/ (bottam up) O(n^3)

i)
```
if(ispal[i][j])
    dp[i][j] = 0;
else{
    for(int k=i;k<j;k++){
        dp[i][j] = min(dp[i][j], dp[i][k] + dp[k+1][j] +1);
    }
}
```

b) abcb ⇒ a | bcb , ab | cb, abc | b

c) https://leetcode.com/submissions/detail/314502291/ (top down)

i)
```
for(int i=start;i<end;i++){
    ans = min(ans, 1+findMincut(A,start,i) + findMincut(A,i+1,end));
}
```

d) https://leetcode.com/problems/palindrome-partitioning-ii/submissions/ O(n^2)

## 24) Word Break Problem

a) https://leetcode.com/submissions/detail/314851563/ (top down)

i)
```
for(int i=1;i<=maxlen;i++)

    if(dict.find(s.substr(start,i)) != dict.end() && canBreak(s,start+i))

        return true;
```

b) https://leetcode.com/submissions/detail/314912714/ (bottom up) O(n^3)

i)
```
if(dict.find(s.substr(i,len))!=dict.end())

    dp[i][j] = true;

else{

    for(int k=i;k<j;k++){

        dp[i][j] =  dp[i][k] && dp[k+1][j];

        if(dp[i][j]) break;

    }

}
```

c) O(n2) https://leetcode.com/problems/word-break/discuss/43790/Java-implementation-using-DP-in-two-ways

## 25) Word break 2

a) https://www.interviewbit.com/problems/word-break-ii/

b) Dp[i] = string of vector of solution from 0 to i

c) For i in 0 to n-1

i) For j in 0 to i

```
Stg = stg(j , i)

If ( dict contains stg and dp[j-1] > 0)

    For x in dp[j-1]

        Dp[i] += stg + ‘ ‘ + x
```

## 26) Dungeon Princesses

a) https://www.interviewbit.com/problems/dungeon-princess/

## 27) Buy and Sell

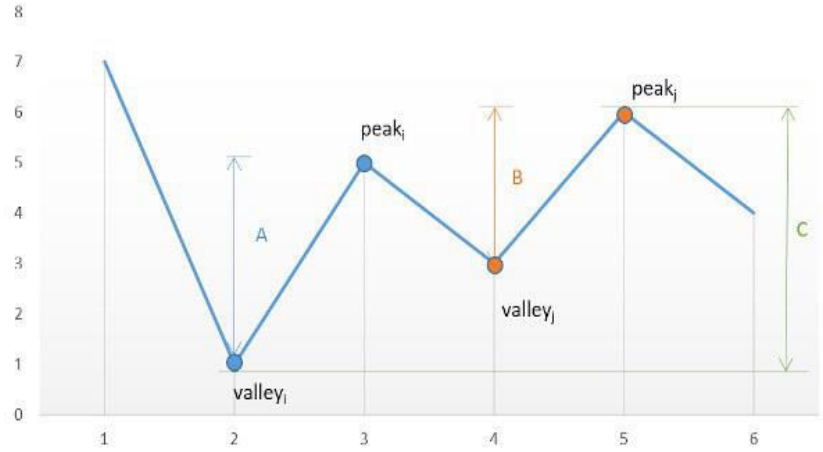
a) Any no of times

https://www.interviewbit.com/problems/best-time-to-buy-and-sell-stocks-ii/

Option 1: Buy stock at valley[i] then sell at peak[i] makes profit A (peak[i] - valley[i]). Then buy stock at valley [j] and sell at peak[j] makes profit B (peak[j] - valley[j]). So the total profit of this trade option is A + B.

Option 2: Skip the intermediate trades, i.e,, we buy stock at valley[i] then sell at peak[j]. In this case, the total profit will be C (peak[j]-valley[i]).

Based on the graph shown below, A + B > C (if not, peak[i] and valley[j] won't exist). So in order to maximize the profit, we can buy stock at valleys and then sell stock at peaks.




b)
c) At Most two
   i) https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/
d) Exact k transactions
   i) https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/
   ii) Either dont do a transaction on jth day or do do transaction with maximum profit
   iii) If k >= n/2 than maxprofit (a)

   Else

   dp[i][j] = max(dp[i][j-1]); // i transactions j days

   max(prices[j] - prices[m] + dp[i-1][m]); 0<= m < j

   iv) Optimization using maxvalue refer link
e) https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/discuss/108870/Most-consistent-ways-of-dealing-with-the-series-of-stock-problems
   i) https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown
   ii) https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee

## 28) Maximum Product subarray

a) www.interviewbit.com/problems/max-product-subarray/
b) Dpmax[i] = max subarray ending at index i
c) Dpmin[i] = min subarray ending at index i
d) Dpmax = max(A[i],dpmax*A[i],dpmin*A[i])
e) Dpmax = min(A[i],dpmax*A[i],dpmin*A[i])

## 29) Smallest k no whose prime factors are only Primes p1 p2 and p3

a) https://www.interviewbit.com/problems/smallest-sequence-with-given-primes/

```
dp[0] = 1;
    for(int i=1;i<D+1;i++){
        int mine = min(dp[p1]*A, min(dp[p2]*B, dp[p3]*C));
        if(mine == dp[p1]*A) p1++;
        if(mine == dp[p2]*B) p2++;
        if(mine == dp[p3]*C) p3++;
        dp[i] = mine;
    }
```

30) Flip min no so resultant array is min no negative

a) https://www.interviewbit.com/problems/flip-array/
b) S1 + s2 = sum

   S1 - s2 = 0

   So s2 = sum/2

   Using knapsack find no of items with max target possible