Longest Repeating Subsequence

Str: a e a b c k d b n c

LCS (S1, S1) → S1.len( )

normal

LCS (S1, S1) → LRS

modified

(i) sharing of chars is not allowed.

(ii) LCS

S1 -> a e a k b m b

	a	e	a	k	b	m	b	-
a	2	2	2	1	1	1	1	0
e	2	1	1	1	1	1	1	0
a	2	1	1	1	1	1	1	0
k	1	1	1	1	1	1	1	0
b	1	1	1	1	1	1	1	0
m	1	1	1	1	1	0	0	0
b	1	1	1	1	1	0	0	0
-	0	0	0	0	0	0	0	0

```
if( ch(i) == ch(j) && i != j)
  dp[i][j] = 1 + dp[i+1][j+1];
}
else {
  dp[i][j] = max(dp[i][j+1], dp[i+1][j])
}
```

Arithmetic Slices 1

count of subarrays forming an AP of atleast 3 length.

O(n)

arr : 1 3 5 6 4 8 12 16 20 19 22 26

X	X	1	0	0	0	1	2	3	0	0	0

dp :

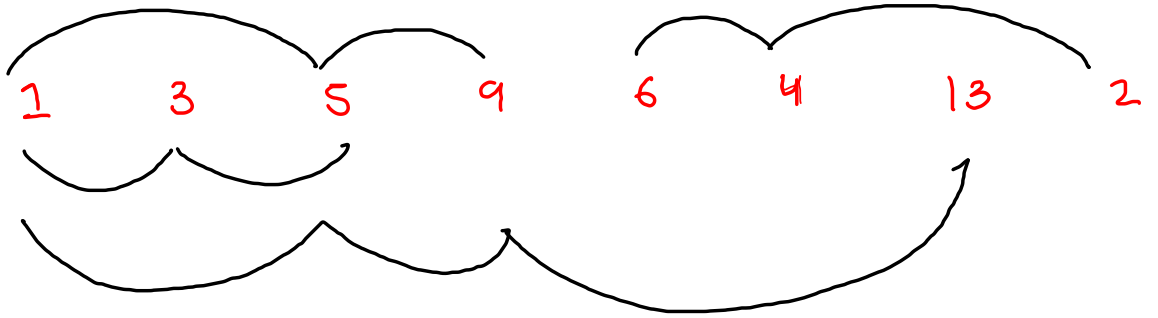
sum of array

dp[i] -> count of subarrays ending at i which forms an ap of len ≥ 3

Arithmetic Slices II - Subsequence

Count all subseq of nums forming an ap of atleast 3.

arr : 1 3 5 9 6 4 13 2



brute force : $n \cdot 2^n$

optimised appro

4	2	3	5	2	6	7	9	8
X	-2 -> 1 (4,2)	1 - 1 (2,3) -1 -> 1 (4,3)	2 - 1 (3,5) 3 - 1 (2,5) 1 - 1 (4,5)	-3 -> 1 (5,2) -1 -> 2 (4,3,2 3,2) 0 -> 1 (2,2) -2 -> 1 (4,2)	4 -> 2 (2,6 2,6) 1 -> 2 (4,5,6 5,6) 3 - 1 (3,6) 2 - 1 (4,6)	1 -> 3 (4,5,6,7 5,6,7 6,7) 5 - 1 (2,7) 2,7 2 - 2 (3,5,7 5,7) 4 - 1 (3,7) 3 - 1 (4,7)		

cd vs count of cup's ending at i of length ≥ 2

ans:

1 (4, 3, 2) 1 (3, 5, 7)

1 (4, 5, 6)

2 [4, 5, 6, 7
5, 6, 7]

oans = 0 + 0 + 1 + 1 + 0 + 2 + 1

(4, 3, 2) (4, 5, 6) (4, 5, 6, 7 / 5, 6, 7) (3, 5, 7)

4	2	3	5	2	6	7
X	-2 -> 1 (4, 2)	1 -> 1 (2, 3) -1 -> 1 (4, 3)	2 -> 1 (3, 5) 3 -> 1 (2, 5) 1 -> 1 (4, 5)	-3 -> 1 (5, 2) -1 -> 2 (4, 3, 2 / 3, 2) 0 -> 1 (2, 2) -2 -> 1 (4, 2)	4 -> 2 (2, 6 / 2, 6) 1 -> 2 (4, 5, 6 / 5, 6) 3 -> 1 (3, 6) 2 -> 1 (4, 6)	1 -> 3 (4, 5, 6, 7 / 5, 6, 7 / 6, 7) 5 -> 2 (2, 7 / 2, 7) 2 -> 2 (3, 5, 7 / 5, 7) 4 -> 1 (3, 7) 3 -> 1 (4, 7)

```java
int oans = 0;
for(int i=1; i < nums.length;i++) {
    for(int j=i-1; j >= 0;j--) {
        long d = (long)nums[i] - (long)nums[j];

        int vj = dp[j].getOrDefault(d,0);
        int vi = dp[i].getOrDefault(d,0);

        oans += vj;

        dp[i].put(d,vj + vi + 1);
    }
}
```

264. Ugly Number II

An **ugly number** is a positive integer whose prime factors are limited to `2`, `3`, and `5`.

X	1	2	3	4	5	6	8	9	10	12	~~14~~	
0	1	2	3	4	5	6	7	8	9	10	11	12

wrong

2x7

2 → ~~2~~ ~~4~~ ~~6~~ ~~8~~ ~~10~~ ~~12~~ 14 16 . . . .

3 → ~~3~~ ~~6~~ ~~9~~ ~~12~~ 15 18 21 24 . . . . .

5 → ~~5~~ ~~10~~ 15 20 25 30 35 40 . . . .

						3p		2p				
X	1	2	3	4	5	6	8	9	10	12	15	16
0	1	2	3	4	5	6	7	8	9	10	11	12
				5p								

$2 * dp[2p]$

$3 * dp[3p]$

$5 * dp[5p]$

n = 12

```java
for(int i=2; i <= n;i++) {
    int c2 = 2 * dp[p2];
    int c3 = 3 * dp[p3];
    int c5 = 5 * dp[p5];

    int un = Math.min(Math.min(c2,c3),c5);
    dp[i] = un;

    if(un == c2) {
        p2++;
    }
    if(un == c3) {
        p3++;
    }
    if(un == c5) {
        p5++;
    }
}
```

						p3		p2				
X	1	2	3	4	5	6	8	9	10	12	15	16
0	1	2	3	4	5	6	7	8	9	10	11	12
				p5								