Ugly Number

prime factor : 2, 3, 5

1st ugly no. : 1

n = 12

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

2 * dp[p2]

3 * dp[p3]

5 * dp[p5]

Super Ugly Number

A **super ugly number** is a positive integer whose prime factors are in the array `primes`.

Given an integer `n` and an array of integers `primes`, return *the* `n^th` ***super ugly number***.

The `n^th` **super ugly number** is **guaranteed** to fit in a **32-bit** signed integer.

K - variables

↳ array

primes

primes[k] * dp[ptr[k]]

2 * 19

7 * 7

13 * 4

19 * 2

ptr

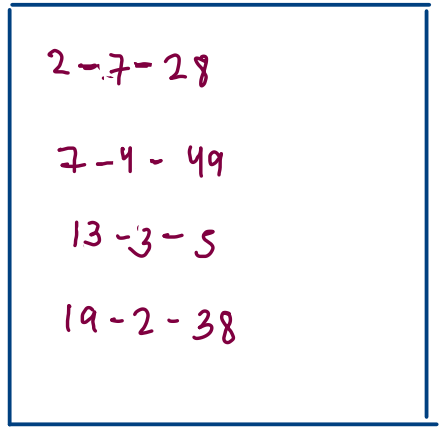
0	1	2	3
9	4	3	2
p2	p7	p13	p19

0	1	2	3	4	5	6	7	8	9	10	11	12
X	1	2	4	7	8	13	14	16	19	26	28	32

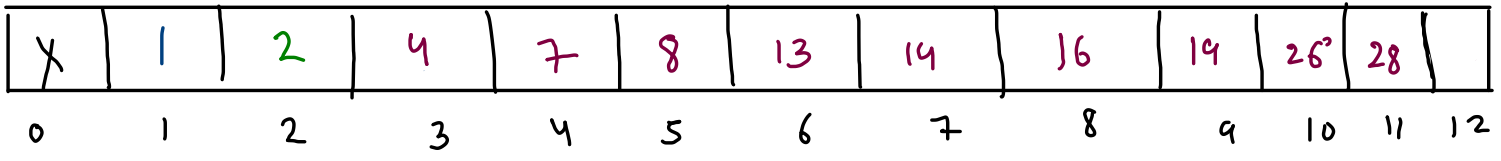
primes

use PQ

min-heap (value is the basis)




```
pair {
  int prime;
  int ptr;
  int value;
}
```

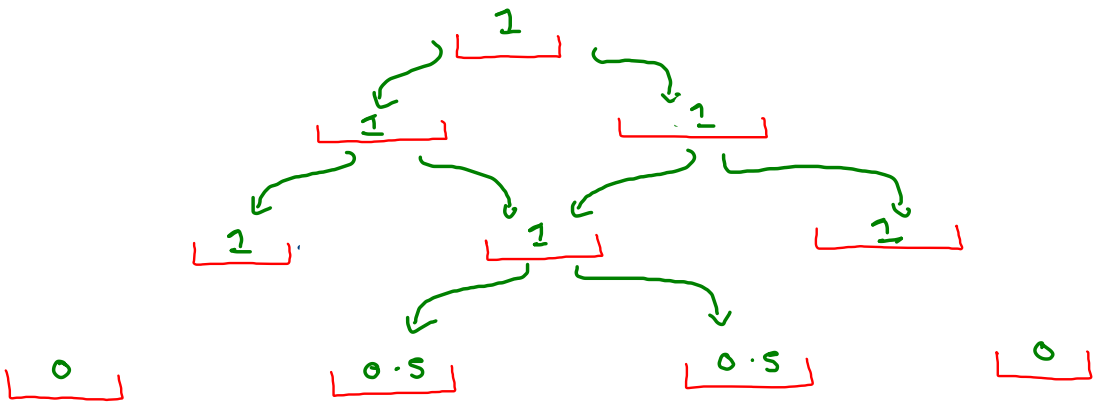


Water Overflow

Medium Accuracy: 36.91% Submissions: 2678 Points: 4

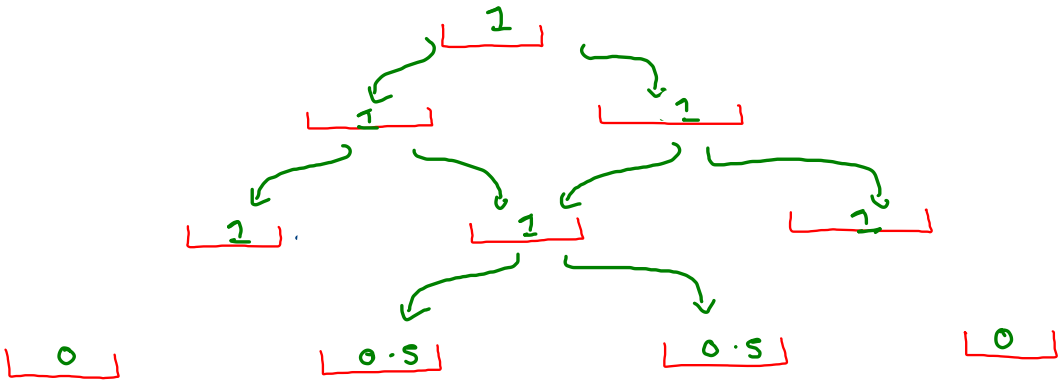
There is a stack of water glasses in a form of pascal triangle and a person wants to pour the water at the topmost glass, but the capacity of each glass is 1 unit. Overflow takes place in such a way that after 1 unit, 1/2 of remaining unit gets into bottom left glass and other half in bottom right glass.Now John pours **K** units of water in the topmost glass and wants to know how much water is there in the **C**th glass of the **R**th row.

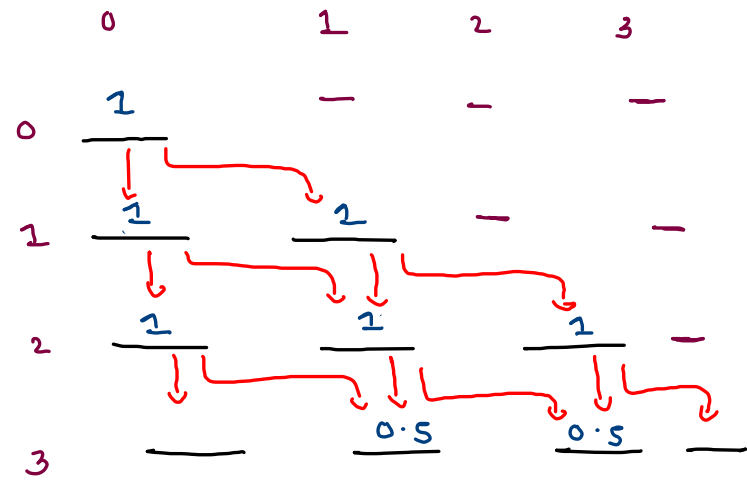
Note: Assume that there are enough glasses in the triangle till no glass overflows.



K = 7

R = 3 C = 2


1
1
1
1
1
1
0
0.5
0.5
0


0
1
2
3
0
1
2
3
1
1
1
1
1
1
1
1
1
0.5
0.5

cap = 4

r = 4, c = 2

```
double[][]dp = new double[R + 1][R + 1];
```

dp[0][0] = 1.0;

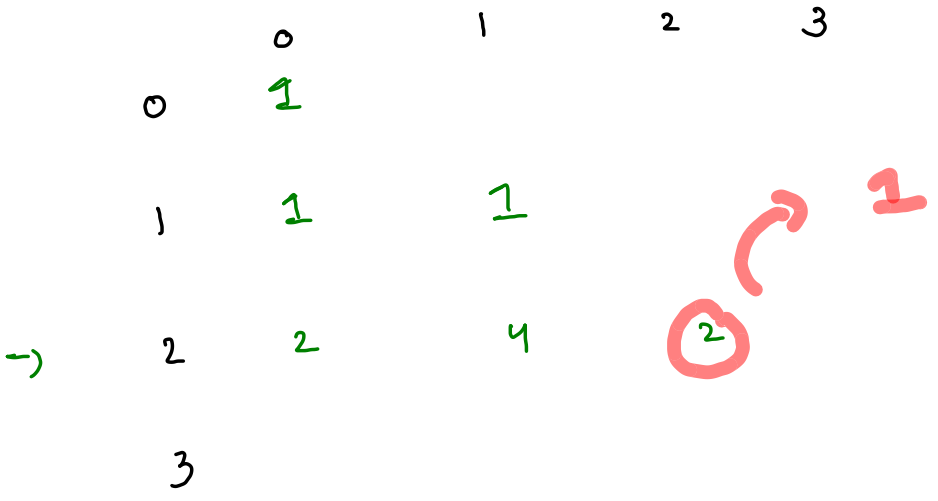
```
for(int i=0; i < R;i++) {
    for(int j=0; j <= i;j++) {
        double ew = dp[i][j] - 1;

        if(ew > 0.0) {
            dp[i][j] = 1.0;

            //equally distribute this extra water in left and right nbr
            dp[i+1][j] +=  ew / 2; //left nbr
            dp[i+1][j+1] += ew / 2; //right nbr
        }
    }
}

if(dp[R-1][C-1] > 1.0) {
    return 1.0;
}
else {
    return dp[R-1][C-1];
}
```

	0	1	2	3	4
0	1	–	–	–	–
1	1	1	–	–	–
2	1	1	1	–	–
3	0.25	1	1	0.25	–
→ 4	0	0.125	0.25	0.125	

```
for(int i=0; i < R;i++) {
    for(int j=0; j <= i;j++) {
        double ew = dp[i][j] - 1;

        if(ew > 0.0) {
            dp[i][j] = 1.0;

            //equally distribute this extra water in left and right nbr
            dp[i+1][j] +=  ew / 2; //left nbr
            dp[i+1][j+1] += ew / 2; //right nbr
        }
    }
}

if(dp[R-1][C-1] > 1.0) {
    return 1.0;
}
else {
    return dp[R-1][C-1];
}
```

r = 2, c = 2

k = 11



Edit Distance

Given two strings `word1` and `word2`, return *the minimum number of operations required to convert* `word1` *to* `word2`.

You have the following three operations permitted on a word:

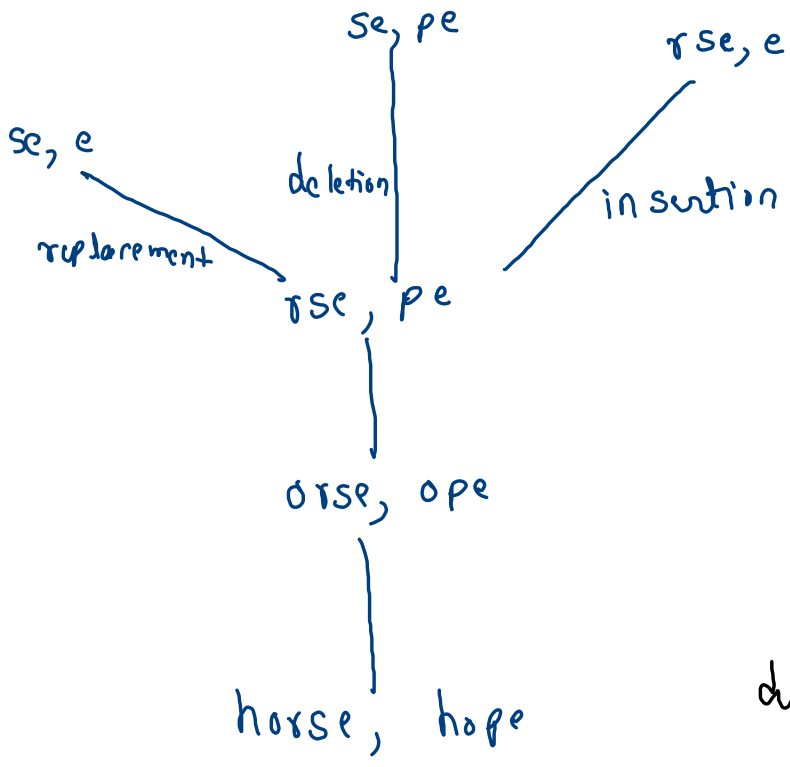
- Insert a character
- Delete a character
- Replace a character

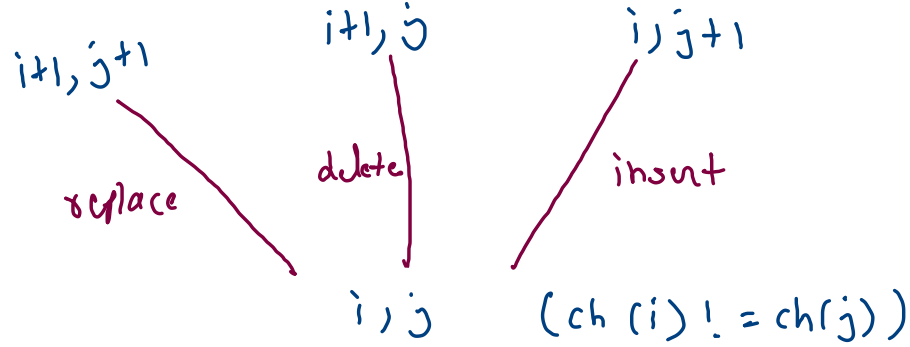
insertion, deletion, replacement

w1 -> amaze

w2 -> wow

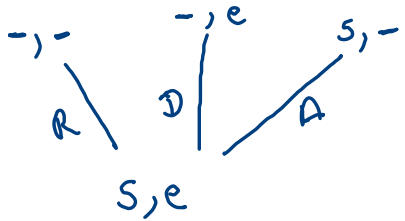
ans: 5



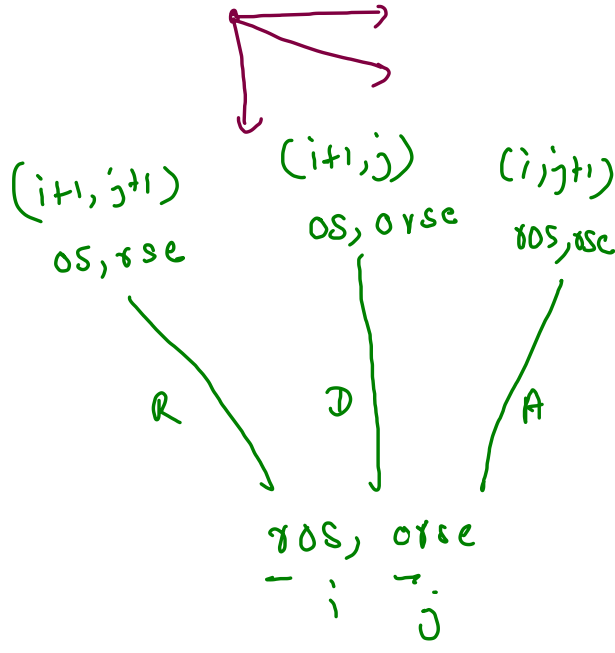


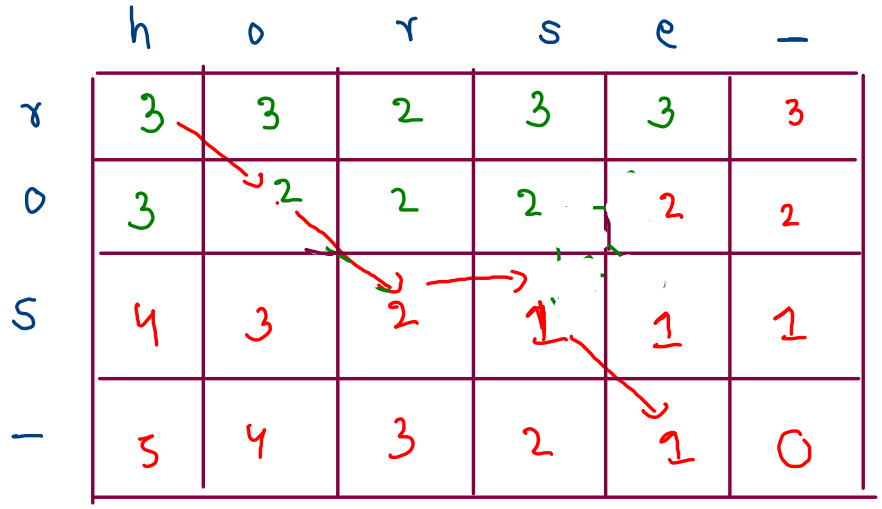
$$dp[i][j] \rightarrow \begin{cases} dp[i+1][j+1] & ; ch(i) == ch(j) \\ 1 + \min \begin{pmatrix} i+1, j+1 & (R) \\ i+1, j & (D) \\ i, j+1 & (A) \end{pmatrix} & ch(i) != ch(j) \end{cases}$$

$$dp[i][j] \rightarrow \begin{cases} dp[i+1][j+1] & ; ch(i) == ch(j) \\ 1 + \min \begin{pmatrix} i+1, j+1 \ (R) \\ i+1, j \ (D) \\ i, j+1 \ (A) \end{pmatrix} & ch(i) != ch(j) \end{cases}$$



	h	o	r	s	e	–
r	3	3	2	3	3	3
o	3	2	2	2	2	2
s	4	3	2	1	1	1
–	5	4	3	2	1	0





ros

↓

horse

Rev. Eng.

replacement (h, r)

Addition (r)

addition (e)