geeksforgeeks.org

Ugly Numbers - GeeksforGeeks

GeeksforGeeks

8-10 minutes

Ugly numbers are numbers whose only prime factors are 2, 3 or 5. The sequence 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, … shows the first 11 ugly numbers. By convention, 1 is included.
Given a number n, the task is to find n’th Ugly number.

Examples:

```
Input  : n = 7
Output : 8

Input  : n = 10
Output : 12

Input  : n = 15
Output : 24

Input  : n = 150
Output : 5832
```

Method 1 (Simple)
Loop for all positive integers till the ugly number count is smaller than n, if an integer is ugly than increment ugly number count.

To check if a number is ugly, divide the number by greatest divisible powers of 2, 3 and 5, if the number becomes 1 then it is an ugly number otherwise not.

For example, let us see how to check for 300 is ugly or not. Greatest divisible power of 2 is 4, after dividing 300 by 4 we get 75. Greatest divisible power of 3 is 3, after dividing 75 by 3 we get 25. Greatest divisible power of 5 is 25, after dividing 25 by 25 we get 1. Since we get 1 finally, 300 is ugly number.

Below is the implementation of the above approach:

- C++
- C
- Java
- Python3
- C#
- PHP
- Javascript

C++

```
#include <iostream>

using namespace std;
```

```
int maxDivide(int a, int b)
{
    while (a % b == 0)
        a = a / b;
    return a;
}
int isUgly(int no)
{
    no = maxDivide(no, 2);
    no = maxDivide(no, 3);
    no = maxDivide(no, 5);
    return (no == 1) ? 1 : 0;
}
int getNthUglyNo(int n)
{
    int i = 1;
    int count = 1;
    while (n > count)
    {
        i++;
        if (isUgly(i))
            count++;
    }
    return i;
}
int main()
{
    unsigned no = getNthUglyNo(150);
    cout << "150th ugly no. is " << no;
    return 0;
}
```

C

Java

Python3

C#

PHP

Javascript

Output

```
150th ugly no. is 5832
```

This method is not time efficient as it checks for all integers until ugly number count becomes n, but space complexity of this method is O(1)

Method 2 (Use Dynamic Programming)

Here is a time efficient solution with O(n) extra space. The ugly-number sequence is 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, …

because every number can only be divided by 2, 3, 5, one way to look at the sequence is to split the sequence to three groups as below:

(1) 1×2, 2×2, 3×2, 4×2, 5×2, …

(2) 1×3, 2×3, 3×3, 4×3, 5×3, …

(3) 1×5, 2×5, 3×5, 4×5, 5×5, …

We can find that every subsequence is the ugly-sequence itself (1, 2, 3, 4, 5, …) multiply 2, 3, 5. Then we use similar merge method as merge sort, to get every ugly number from the three subsequences. Every step we choose the smallest one, and move one step after.

```
1 Declare an array for ugly numbers:  ugly[n]
2 Initialize first ugly no:  ugly[0] = 1
3 Initialize three array index variables i2, i3, i5 to point to
   1st element of the ugly array:
        i2 = i3 = i5 =0;
4 Initialize 3 choices for the next ugly no:
         next_multiple_of_2 = ugly[i2]*2;
         next_multiple_of_3 = ugly[i3]*3
         next_multiple_of_5 = ugly[i5]*5;
5 Now go in a loop to fill all ugly numbers till 150:
For (i = 1; i < 150; i++ )
{
    /* These small steps are not optimized for good
      readability. Will optimize them in C program */
    next_ugly_no  = Min(next_multiple_of_2,
                        next_multiple_of_3,
                    next_multiple_of_5);

    ugly[i] =  next_ugly_no

    if (next_ugly_no  == next_multiple_of_2)
    {
       i2 = i2 + 1;
       next_multiple_of_2 = ugly[i2]*2;
    }
    if (next_ugly_no  == next_multiple_of_3)
    {
       i3 = i3 + 1;
       next_multiple_of_3 = ugly[i3]*3;
     }
     if (next_ugly_no  == next_multiple_of_5)
     {
       i5 = i5 + 1;
       next_multiple_of_5 = ugly[i5]*5;
     }

}/* end of for loop */
6.return next_ugly_no
```

Example:

Let us see how it works

```
initialize
  ugly[] =  | 1 |
  i2 =  i3 = i5 = 0;

First iteration
```

```
  ugly[1] = Min(ugly[i2]*2, ugly[i3]*3, ugly[i5]*5)
          = Min(2, 3, 5)
          = 2
  ugly[] =  | 1 | 2 |
  i2 = 1,  i3 = i5 = 0  (i2 got incremented )

Second iteration
   ugly[2] = Min(ugly[i2]*2, ugly[i3]*3, ugly[i5]*5)
           = Min(4, 3, 5)
           = 3
   ugly[] =  | 1 | 2 | 3 |
   i2 = 1,  i3 =  1, i5 = 0  (i3 got incremented )

Third iteration
   ugly[3] = Min(ugly[i2]*2, ugly[i3]*3, ugly[i5]*5)
           = Min(4, 6, 5)
           = 4
   ugly[] =  | 1 | 2 | 3 |  4 |
   i2 = 2,  i3 =  1, i5 = 0  (i2 got incremented )

Fourth iteration
   ugly[4] = Min(ugly[i2]*2, ugly[i3]*3, ugly[i5]*5)
            = Min(6, 6, 5)
            = 5
   ugly[] =  | 1 | 2 | 3 |  4 | 5 |
   i2 = 2,  i3 =  1, i5 = 1  (i5 got incremented )

Fifth iteration
   ugly[4] = Min(ugly[i2]*2, ugly[i3]*3, ugly[i5]*5)
            = Min(6, 6, 10)
```

```
        = 6
   ugly[] =  | 1 | 2 | 3 |  4 | 5 | 6 |
   i2 = 3,  i3 =  2, i5 = 1  (i2 and i3 got incremented )
```

Will continue same way till I < 150

- C++
- C
- Java
- Python
- C#
- PHP
- Javascript

C++

```
#include <bits/stdc++.h>

using namespace std;

unsigned getNthUglyNo(unsigned n)

{

    unsigned ugly[n];

    unsigned i2 = 0, i3 = 0, i5 = 0;

    unsigned next_multiple_of_2 = 2;

    unsigned next_multiple_of_3 = 3;

    unsigned next_multiple_of_5 = 5;

    unsigned next_ugly_no = 1;
    ugly[0] = 1;
    for (int i = 1; i < n; i++) {
        next_ugly_no = min(
            next_multiple_of_2,
            min(next_multiple_of_3,
next_multiple_of_5));
        ugly[i] = next_ugly_no;
        if (next_ugly_no == next_multiple_of_2) {
            i2 = i2 + 1;
            next_multiple_of_2 = ugly[i2] * 2;
        }
        if (next_ugly_no == next_multiple_of_3) {
            i3 = i3 + 1;
            next_multiple_of_3 = ugly[i3] * 3;
        }
        if (next_ugly_no == next_multiple_of_5) {
            i5 = i5 + 1;
            next_multiple_of_5 = ugly[i5] * 5;
        }
    }
    return next_ugly_no;
}
int main()
```

```
{
    int n = 150;
    cout << getNthUglyNo(n);
    return 0;
}
```

C

Java

Python

C#

PHP

Javascript

Time Complexity: O(n)
Auxiliary Space: O(n)
Super Ugly Number (Number whose prime factors are in the given set)

Method 3 (Using SET Data Structure in C++, Javascript and TreeSet in JAVA)

In this method, SET data structure to store the minimum of the 3 generated ugly numbers which will the ith Ugly Number stored at the first position of the set. SET Data Structure as a set stores all the element in ascending order

Below is the implementation of the above approach:

- C++
- Java
- Python3
- C#
- Javascript

C++

```
#include <bits/stdc++.h>

using namespace std;

int nthUglyNumber(int n)

{

    if (n == 1 or n == 2 or n == 3 or n == 4 or n ==
5)

        return n;

    set<long long int> s;

    s.insert(1);

    n--;

    while (n) {

        auto it = s.begin();

        long long int x = *it;

        s.erase(it);

        s.insert(x * 2);

        s.insert(x * 3);
        s.insert(x * 5);

        n--;

    }

    return *s.begin();

}

int main()

{

    int n = 150;

    cout << nthUglyNumber(n);

}
```

Java

Python3

C#

Javascript

Time Complexity:- O(N log N)
Auxiliary Space:- O(N)

Method 4(Using Binary Search)

1. This method is suitable if you have a max value for n. The no will be of form x=pow(2,p)*pow(3,q)*pow(5,r).
2. Search from low=1 to high =21474836647. We are expecting nth ugly no to be in this range.
3. So we do a binary search. Now suppose we are at mid now we are

going to find the total number of ugly numbers less than mid and put our conditions accordingly.

Below is the rough CPP code:

- C++
- Java
- Python3
- C#
- Javascript

C++

```
#include <bits/stdc++.h>

using namespace std;

int nthUglyNumber(int n)

{

  int pow[40] = { 1 };

  for (int i = 1; i <= 30; ++i)

    pow[i] = pow[i - 1] * 2;

  int l = 1, r = 2147483647;

  int ans = -1;

  while (l <= r) {

    int mid = l + ((r - l) / 2);

    int cnt = 0;

    for (long long i = 1; i <= mid; i *= 5)
    {
      for (long long j = 1; j * i <= mid; j *= 3)
      {
        cnt += upper_bound(pow, pow + 31,
                           mid / (i * j)) - pow;
      }
    }
    if (cnt < n)
      l = mid + 1;
    else
      r = mid - 1, ans = mid;
  }
  return ans;
}
int main()
{
    int n = 150;
    cout << nthUglyNumber(n);
    return 0;
}
```

Java

Python3

C#

Javascript

Time Complexity: O(log N)

Auxiliary Space: O(1)

Please write comments if you find any bug in the above program or other ways to solve the same problem.