Super Ugly Number (Number whose prime factors are in given set)

Difficulty Level : Medium • Last Updated : 15 Jun, 2022

Super ugly numbers are positive numbers whose all prime factors are in the given prime list. Given a number n, the task is to find the nth Super Ugly number.
It may be assumed that a given set of primes is sorted. Also, the first Super Ugly number is 1 by convention.

Examples:

```
Input  : primes[] = [2, 5]
         n = 5
Output : 8
Super Ugly numbers with given prime factors
are 1, 2, 4, 5, 8, ...
Fifth Super Ugly number is 8

Input  : primes[] = [2, 3, 5]
         n = 50
Output : 243

Input : primes[] = [3, 5, 7, 11, 13]
        n = 9
Output: 21
```

Recommended: Please try your approach on ***{IDE}*** first, before moving on to the solution.

In our previous post, we discussed Ugly Number. This problem is basically an extension of **Ugly Numbers**.
A **simple solution** for this problem is to one by one pick each number starting from 1 and find it's all primes factors, if all prime factors lie in the given set of primes that means the number is Super Ugly. Repeat this process until we get nth Super Ugly Number.

An **efficient solution** for this problem is similar to **Method-2** of Ugly Number. Here is the algorithm:

1. Let **k** be the size of a given array of prime numbers.
2. Declare a set for super ugly numbers.
3. Insert the first ugly number (which is always 1) into the set.
4. Initialize array **multiple_of[k]** of size k with 0. Each element of this array is an iterator for the corresponding prime in primes[k] array.
5. Initialize **nextMultipe[k]** array with primes[k]. This array behaves like next multiple variables of each prime in given primes[k] array i.e; nextMultiple[i] = primes[i] * ugly[++multiple_of[i]].
6. Now loop until there are n elements in set ugly.
   a). Find minimum among current multiples of primes in nextMultiple[] array and insert it in the set of ugly numbers.
   b). Then find this current minimum is multiple of which prime.
   c). Increase iterator by 1 i.e; ++multiple_Of[i], for next multiple of current selected prime and update nextMultiple for it.

Below is the implementation of the above steps.

CPP

```
// C++ program to find n'th Super Ugly number
```

```
#include<bits/stdc++.h>
using namespace std;

// Function to get the nth super ugly number
// primes[]       --> given list of primes f size k
// ugly           --> set which holds all super ugly
//                    numbers from 1 to n
// k              --> Size of prime[]
int superUgly(int n, int primes[], int k)
{
    // nextMultiple holds multiples of given primes
    vector<int> nextMultiple(primes, primes+k);

    // To store iterators of all primes
    int multiple_Of[k];
    memset(multiple_Of, 0, sizeof(multiple_Of));

    // Create a set to store super ugly numbers and
    // store first Super ugly number
    set<int> ugly;
    ugly.insert(1);

    // loop until there are total n Super ugly numbers
    // in set
    while (ugly.size() != n)
    {
        // Find minimum element among all current
        // multiples of given prime
        int next_ugly_no = *min_element(nextMultiple.begin(),
                                        nextMultiple.end());

        // insert this super ugly number in set
        ugly.insert(next_ugly_no);

        // loop to find current minimum is multiple
        // of which prime
        for (int j=0; j<k; j++)
        {
            if (next_ugly_no == nextMultiple[j])
            {
                // increase iterator by one for next multiple
                // of current prime
                multiple_Of[j]++;

                // this loop is similar to find  dp[++index[j]]
```

```
                // it -->  dp[++index[j]]
                set<int>::iterator it = ugly.begin();
                for (int i=1; i<=multiple_Of[j]; i++)
                    it++;

                nextMultiple[j] = primes[j] * (*it);
                break;
            }
        }
    }

    // n'th super ugly number
    set<int>::iterator it = ugly.end();
    it--;
    return *it;
}

/* Driver program to test above functions */
int main()
{
    int primes[] = {2,  5};
    int k = sizeof(primes)/sizeof(primes[0]);
    int n = 5;
    cout << superUgly(n, primes, k);
    return 0;
}
```

Output:

```
 8
```

This article is contributed by **Shashank Mishra (Gullu)**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Another method (using priority_queue)

Here we use min heap priority_queue.

The idea is to push the first ugly no. which is 1 into priority_queue and at

every iteration take the top of priority_queue and push all the multiples of that top into priority_queue.
Assuming a[] = {2, 3, 5},
so at first iteration 1 is top so 1 is popped and 1 * 2, 1 * 3, 1 * 5 is pushed.
At second iteration min is 2, so it is popped and 2 * 2, 2 * 3, 2 * 5 is pushed and so on.

Implementation:

CPP

```
// C++ program for super ugly number
#include<bits/stdc++.h>
using namespace std;
// function will return the nth super ugly number
int ugly(int a[], int size, int n){

    // n cannot be negative hence
    // return -1 if n is 0 or -ve
    if(n <= 0)
        return -1;

    if(n == 1)
        return 1;

    // Declare a min heap priority queue
    priority_queue<int, vector<int>, greater<int>> pq;

    // Push all the array elements to priority queue
    for(int i = 0; i < size; i++){
        pq.push(a[i]);
    }

    // once count = n we return no
    int count = 1, no;

    while(count < n){
        // Get the minimum value from priority_queue
        no = pq.top();
        pq.pop();
```

```
        // If top of pq is no then don't increment count.
        // This to avoid duplicate counting of same no.
        if(no != pq.top())
        {
            count++;

            // Push all the multiples of no. to priority_queue
            for(int i = 0; i < size; i++){
                pq.push(no * a[i]);
            // cnt+=1;
        }
        }
    }
    // Return nth super ugly number
    return no;
}

/* Driver program to test above functions */
int main(){
    int a[3] = {2, 3,5};
    int size = sizeof(a) / sizeof(a[0]);
    cout << ugly(a, size, 10)<<endl;
    return 0;
}
```

Python3

```
# Python3 program for super ugly number

# function will return the nth super ugly number
def ugly(a, size, n):

    # n cannot be negative hence return -1 if n is 0 or -ve
    if(n <= 0):
        return -1
    if(n == 1):
        return 1

    # Declare a min heap priority queue
    pq = []

    # Push all the array elements to priority queue
```

```
    for i  in range(size):
        pq.append(a[i])

    # once count = n we return no
    count = 1
    no = 0
    pq = sorted(pq)

    while(count < n):
        # sorted(pq)
        # Get the minimum value from priority_queue
        no = pq[0]
        del pq[0]


        # If top of pq is no then don't increment count.
        # This to avoid duplicate counting of same no.
        if(no != pq[0]):
            count += 1

            # Push all the multiples of no. to priority_queue
            for i in range(size):
                pq.append(no * a[i])
            #   cnt+=1
        pq = sorted(pq)
    # Return nth super ugly number
    return no

# /* Driver program to test above functions */
if __name__ == '__main__':
    a = [2, 3,5]
    size = len(a)
    print(ugly(a, size, 1000))

    # This code is contributed by mohit kumar 29.
```

Javascript

```
<script>
// Javascript program for super ugly number

    //function will return the nth super ugly number
    function ugly(a,size,n)
    {
```

```
    //n cannot be negative hence return -1 if n is 0 or -ve
if(n <= 0)
    return -1;

if(n == 1)
    return 1;

// Declare a min heap priority queue
let pq=[];;

// Push all the array elements to priority queue
for(let i = 0; i < size; i++){
    pq.push(a[i]);
}

// once count = n we return no
let count = 1, no;
pq.sort(function(a,b){return a-b;}) ;

while(count < n){
    // Get the minimum value from priority_queue
    no = pq.shift();


    // If top of pq is no then don't increment count. This to avoid d
    if(no != pq[0])
    {
        count++;

        //Push all the multiples of no. to priority_queue
        for(let i = 0; i < size; i++){
            pq.push(no * a[i]);
        //    cnt+=1;
    }

    }
    pq.sort(function(a,b){return a-b;}) ;
}
// Return nth super ugly number
return no;
}
```

```
/* Driver program to test above functions */
let a=[2, 3,5];
```

```
    let size = a.length;
    document.write(ugly(a, size, 1000));


    // This code is contributed by unknown2108
</script>
```

Output:

```
51200000
```

Time Complexity: O(n*size*logn)

Auxiliary Space: O(n)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Previous

Jacobsthal and Jacobsthal-Lucas numbers

Next

Ugly Numbers

RECOMMENDED ARTICLES

Page : 1 2 3

01 **Print all numbers whose set of prime factors is a subset of the set of the prime factors of X**
22, Apr 19

02 **Count numbers in a given range whose count of prime factors is a Prime Number**
28, Oct 20

03 **Check whether a given number is an ugly number or not**
20, Jul 20

04 **Maximum number of prime factors a number can have with exactly x factors**
07, May 20

05 **Check if a number exists having exactly N factors and K prime factors**
10, May 20

06 **Count numbers in a given range having prime and non-prime digits at prime and non-prime positions respectively**
15, Jan 21

07 **Count numbers from range whose prime factors are only 2 and 3 using Arrays | Set 2**
31, May 20

08 **Ugly Numbers**
11, Jul 09

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : prajmsidc, mohit kumar 29, unknown2108, keshavgoyal3, hardikkoriintern

Article Tags : prime-factor, series, Dynamic Programming

Practice Tags : Dynamic Programming, series

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

NodeJS