Super Ugly Number

Write a program to find the nth super ugly number.

Super ugly numbers are positive numbers whose all prime factors are in the given prime list `primes` of size `k`. For example, `[1, 2, 4, 7, 8, 13, 14, 16, 19, 26, 28, 32]` is the sequence of the first 12 super ugly numbers given `primes` = `[2, 7, 13, 19]` of size 4.

Note:
(1) `1` is a super ugly number for any given `primes`.
(2) The given numbers in `primes` are in ascending order.
(3) 0 k $\leq$ 100, 0 n $\leq$ 10^6, 0 primes[i]

Credits:
Special thanks to @dietpepsi for adding this problem and creating all test cases.

Solution 1

```
public int nthSuperUglyNumber(int n, int[] primes) {
    int[] ret    = new int[n];
          ret[0] = 1;

    int[] indexes  = new int[primes.length];

    for(int i = 1; i < n; i++){
        ret[i] = Integer.MAX_VALUE;

        for(int j = 0; j < primes.length; j++){
            ret[i] = Math.min(ret[i], primes[j] * ret[indexes[j]]);
        }

        for(int j = 0; j < indexes.length; j++){
            if(ret[i] == primes[j] * ret[indexes[j]]){
                indexes[j]++;
            }
        }
    }

    return ret[n - 1];
}
```

written by larrywang2014 original link here

Solution 2

Keep k pointers and update them in each iteration. Time complexity is O(kn).

```
int nthSuperUglyNumber(int n, vector<int>& primes) {
        vector<int> index(primes.size(), 0), ugly(n, INT_MAX);
        ugly[0]=1;
        for(int i=1; i<n; i++){
            for(int j=0; j<primes.size(); j++) ugly[i]=min(ugly[i],ugly[index[j]]
*primes[j]);
            for(int j=0; j<primes.size(); j++) index[j]+=(ugly[i]==ugly[index[j]]
*primes[j]);
        }
        return ugly[n-1];
}
```

written by zjh08177 original link here

Solution 3

Solution 1 ... ~1570 ms

Using generators and `heapq.merge`. Too bad there's no `itertools.unique`.

```
def nthSuperUglyNumber(self, n, primes):
    uglies = [1]
    def gen(prime):
        for ugly in uglies:
            yield ugly * prime
    merged = heapq.merge(*map(gen, primes))
    while len(uglies) < n:
        ugly = next(merged)
        if ugly != uglies[-1]:
            uglies.append(ugly)
    return uglies[-1]
```

Solution 2 ... ~1400 ms

Same thing done differently and it's a bit faster.

```
def nthSuperUglyNumber(self, n, primes):
    uglies = [1]
    merged = heapq.merge(*map(lambda p: (u*p for u in uglies), primes))
    uniqed = (u for u, _ in itertools.groupby(merged))
    map(uglies.append, itertools.islice(uniqed, n-1))
    return uglies[-1]
```

written by StefanPochmann original link here