ABC 162 解説

kyopro_friends, gazelle, camypaper, tempura0224, ynymxiaolongbao

2020 年 4 月 12 日

For International Readers: English editorial starts on page 7.

A:Lucky 7

N を文字列として受け取り、7 を含むかどうかで判定できます。

```
int main(){
    char s[4];
    scanf("%s",s);
    if(s[0]=='7'||s[1]=='7'||s[2]=='7')puts("Yes");
    else puts("No");
}
```

B:FizzBuzz Sum

実際に各項が数かどうかを判定し、数ならば足すことで答えを求めることができます。

```
int main(){
    int n;
    scanf("%d",&n);
    long long ans=0;
    for(int i=1;i<=n;i++){
        if(i%3!=0 && i%5!=0)ans+=i;
    }
    printf("%lld\n",ans);
}
```

なお、この問題は $O(1)$ で解くこともできます。

```
long long sum(long long n){return n*(n+1)/2;}
int main(){
    int n;
    scanf("%d",&n);
    long long ans;
    ans=sum(n)-sum(n/3)*3-sum(n/5)*5+sum(n/15)*15;
    printf("%lld\n",ans);
}
```

C: sum of gcd of tuples (easy)

$\gcd(a,b,c) = \gcd(\gcd(a,b),c)$ が成立します。

K 以下の 2 つの数の最大公約数は、ユークリッドの互除法を用いることで $O(\log K)$ で求めることが出来ます。したがって、実際に全ての (a,b,c) の組に対して最大公約数を計算することで、この問題は $O(K^3 \log K)$ で解けました。

C 言語でのユークリッドの互除法の実装例は次のとおりです。

再帰版

```c
int gcd(int p, int q){
    if(p % q == 0)return q;
    return gcd(q, p % q);
}
```

非再帰版

```c
int gcd(int p, int q){
    while(q != 0){
        int r = p % q;
        p = q;
        q = r;
    }
    return p;
}
```

D: RGB Triplets

1 つ目の条件を満たす組の数は、S に含まれる R, G, B の数をそれぞれ $r,\ g,\ b$ としたとき rgb です。

このうち 2 つ目の条件を満たさない組がいくつあるかを考えます。$j - i = k - j$ を満たすような組 $(i,\ j,\ k)$ の個数は $O(N^2)$ です。よって、例えば $i,\ j$ を固定するといった方法で、この全てを調べて、それが 1 つ目の条件を満たしているかを確認すればいいです。このアルゴリズムの計算量は、後半の全探索がボトルネックになり $O(N^2)$ です。

E: sum of gcd of tuples (hard)

各数列 $\{A_i\}$ に対して最大公約数を計算していては間に合いません。そこで、$1 \leq X \leq K$ に対して「$\mathrm{gcd}(A_1, ..., A_N) = X$ となる数列 $\{A_i\}$ がいくつあるか？」という問題を考えます。これが解ければ元の問題にも答えることが出来ます。

最大公約数が X の倍数であるための必要十分条件は、$A_1, ..., A_N$ が全て X の倍数であることです。そのような数列は $\lfloor \frac{K}{X} \rfloor^N$ 個あります。

ぴったり X であるための必要十分条件は、「X の倍数であり、かつ、$2X, 3X, ...$ ではない」です。X が大きい方から順に計算していくことによって、$2X, 3X, ...$ の個数を引いて求めることができます。

計算量は $O(K \log K + K \log N)$ です。

F:Select Half

この問題は、もし N が十分小さければ次のような DP で解くことができます。

$DP[i][j] = \{i$ 番目までのうちどの 2 個も連続しない j 個を選んだ時の和の最大値 $\}$

この DP には無駄が多いので、ここから状態数を減らします。連続する要素を選んではいけないので、i 番目までの数のうち選べるのは最大で $\lfloor \frac{i+1}{2} \rfloor$ 個です。同様に、残りの $N-i$ 個から選べるのは最大で $\lfloor \frac{N-i+1}{2} \rfloor$ 個なので、最終的に $\lfloor \frac{N}{2} \rfloor$ 個を選ぶためには、i 番目までに $\lfloor \frac{N}{2} \rfloor - \lfloor \frac{N-i+1}{2} \rfloor$ 個以上選んでいる必要があります。

$\lfloor \frac{N}{2} \rfloor - \lfloor \frac{N-i+1}{2} \rfloor \geq \lfloor \frac{i}{2} \rfloor - 1$ であるので、冒頭の DP で考慮すべき j の値は、各 i について $\lfloor \frac{i}{2} \rfloor - 1$ 以上 $\lfloor \frac{i+1}{2} \rfloor$ 以下の高々 3 通りであることがわかり、状態数及び計算量が $O(N)$ となって解けました。

A:Lucky 7

You can judge by receiving N as a string and check if it contains 7.

```
int main(){
    char s[4];
    scanf("%s",s);
    if(s[0]=='7'||s[1]=='7'||s[2]=='7')puts("Yes");
    else puts("No");
}
```

B:FizzBuzz Sum

You can find the answer by actually checking if each term is a number or not, and adding up if it is a number.

```
int main(){
    int n;
    scanf("%d",&n);
    long long ans=0;
    for(int i=1;i<=n;i++){
        if(i%3!=0 && i%5!=0)ans+=i;
    }
    printf("%lld\n",ans);
}
```

This problem can also be solved in a total of $O(1)$ time.

```
long long sum(long long n){return n*(n+1)/2;}
int main(){
    int n;
    scanf("%d",&n);
    long long ans;
    ans=sum(n)-sum(n/3)*3-sum(n/5)*5+sum(n/15)*15;
    printf("%lld\n",ans);
}
```

C: sum of gcd of tuples (easy)

It holds that $\gcd(a, b, c) = \gcd(\gcd(a, b), c)$.

The greatest common divisor between two numbers less than or equal to K can be found in a total $O(\log K)$ time using the Euclidean Algorithm.

Therefore, by actually calculating the greatest common divisors for all (a, b, c), this problem could be solved in a total of $O(K^3 \log K)$ time.

The following are implementation examples of Euclidean Algorithm in C Language.

Recurrent version

```
int gcd(int p, int q){
    if(p % q == 0)return q;
    return gcd(q, p % q);
}
```

Non-recurrent version

```
int gcd(int p, int q){
    while(q != 0){
        int r = p % q;
        p = q;
        q = r;
    }
    return p;
}
```

D: RGB Triplets

The number of pairs that satisfies the first condition is rgb, where r, g and b denotes the number of appearances of R, G, B in S, respectively. Now let's consider how many of them do not satisfies the second condition. The number of tuples (i, j, k) such that $j - i = k - j$ is $O(N^2)$. Therefore, you can iterate through all the of them by, for example, fixing i and j, and then check if it satisfies the first condition. このアルゴリズムの計算量は、後半の全探索がボトルネックになり $O(N^2)$ です。The time complexity of this algorithm is $O(N^2)$, where the latter brute force is the bottle neck.

E: sum of gcd of tuples (hard)

It won't finish in time if you calculate the greatest common divisor for each sequence $\{A_i\}$. Then, for each $1 \le X \le K$, let us consider the problem "how many sequences $\{A_i\}$ are there such that $\gcd(A_1, ..., A_N) = X$?". If this problem was solved, then the original problem can also be answered.

The necessary and sufficient condition of the greatest common divisor being a multiple of X is that all the $A_1, ..., A_N$ are multiples of X. The number of such sequences is $\lfloor \frac{K}{X} \rfloor^N$.

The necessary and sufficient condition of it being exactly X is that "it is a multiple of X, but not $2X, 3X....$" If they are calculated in the decreasing order of X, it can be found by subtracting the number of those of $2X, 3X, ...$

The time complexity is $O(K \log K + K \log N)$.

F:Select Half

If N is small enough, this problem can be solved by the following DP.

$DP[i][j]$ = {The maximum sum of j elements chosen from the first i elements where no two of them are adjacent}

This DP is inefficient a lot, so we will decrease the number of state from this. Since we cannot choose adjacent elements, we can choose at most $\lfloor\frac{i+1}{2}\rfloor$ elements out of the first i elements. Similarly, we can choose at most $\lfloor\frac{N-i+1}{2}\rfloor$ out of the remaining $N-i$ elements, so in order to choose $\lfloor\frac{N}{2}\rfloor$ elements in the end, at least $\lfloor\frac{N}{2}\rfloor - \lfloor\frac{N-i+1}{2}\rfloor$ elements has to be chosen from the first i elements.

Since $\lfloor\frac{N}{2}\rfloor - \lfloor\frac{N-i+1}{2}\rfloor \geq \lfloor\frac{i}{2}\rfloor - 1$, it appears that at most three kinds of j, which is greater than or equal to $\lfloor\frac{i}{2}\rfloor - 1$ and less than or equal to $\lfloor\frac{i+1}{2}\rfloor$, has to be considered, and now the number of states and the time complexity is $O(N)$, so the problem could be solved.