具体数学初级班第二周参考答案

517 老师

2020 年 5 月 8 日

0.1 A 题

根据欧拉函数的性质 6 可以得到小于 n 与 n 互质的数的和，不互质就是总和减去互质的

另外，此题 n 比价大，没法预处理，需要掌握 $O(\sqrt{n})$ 求欧拉函数

```
#include <bits/stdc++.h>
using namespace std;
const int md = (int)1e9 + 7;
int phi(int n) {
  int ret = n;
  for (int i = 2; i * i <= n; i++) {
    if (n % i == 0) {
      ret = ret / i * (i - 1);
      while (n % i == 0) {
        n /= i;
      }
    }
  }
  if (n > 1) {
    ret = ret / n * (n - 1);
  }
  return ret;
}

int main() {
  int n;
```

```
  while(scanf("%d", &n) == 1) {
    if (n == 0) {
      break;
    }
    long long sum = 1LL * n * (n + 1) / 2 - n;
    sum -= 1LL * n * phi(n) / 2;
    printf("%lld\n", sum % md);
  }
  return 0;
}
```

0.2 B 题

这个题需要一点点的组合数学基础，如果看不懂可以等组合数学课之后再回来做

将一个数拆分成 1，2，3,... ，n 个正整数相加的方案数就相当于，用 $0, 1, 2, ..., n-1$ 块隔板去分割这个数

这里要注意 $1+3+7$ 与 $1+7+3$ 算不同的方案

那么总方案就是 $\binom{n-1}{0} + \binom{n-1}{1} + ... + \binom{n-1}{n-1} = 2^{n-1}$

但是这个题 n 非常大，由于此题模数是个质数，所以根据费马小定理，$a^{\varphi(p)} \% p = 1$，$\varphi(p) = p - 1$

因此 $a^{p-1} \% p = 1$，所以

$$2^{n-1} \% p = 2^{(n-1)\%(p-1)} \% p$$

这也是欧拉函数降幂的一个应用

费马小定理

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

const int N = 100010;
const int md = (int) 1e9 + 7;

char s[N];

int pow_mod(int a, int b, int c) {
```

```
  int ret = 1;
  while (b) {
    if (b & 1) {
      ret = 1LL * ret * a % md;
    }
    b >>= 1;
    a = 1LL * a * a % md;
  }
  return ret;
}


int main() {
  while (scanf("%s", s) == 1) {
    int n = strlen(s);
    long long ret = 0;
    for (int i = 0; i < n; i++) {
      ret = ret * 10 + s[i] - '0';
      ret = ret % (md - 1);
    }
    ret = (ret - 1 + (md - 1)) % (md - 1);
    printf("%d\n", pow_mod(2, ret, md));
  }
  return 0;
}
```


0.3 C 题

直接利用整除分块即可

整除分块

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

```
int main() {
  int t, n, ca = 1;
  scanf("%d", &t);
  while (t--) {
    scanf("%d", &n);
    long long ret = 0;
    for (int i = 1, j; i <= n; i = j + 1) {
      j = n / ( n / i );
      ret += 1LL * (n / i) * (j - i + 1);
    }
    printf("Case %d: ", ca++);
    if (ret & 1) {
      printf("odd\n");
    } else {
      printf("even\n");
    }
  }
  return 0;
}
```


0.4 D 题

此题考查对素数筛法的灵活应用，我们发现 a, b 虽然值比较大，但是他们的差距很小

而且如果某个 x 满足 $a \leq x \leq b$, 而且 x 是个合数，那么 x 一定有一个素因子是小于等于 $\sqrt{x}$ 的

因此我们可以枚举 $\sqrt{b}$ 以内的素因子跳跃到 $[a, b]$ 区间内去筛，保存标记数组可以采用偏移标记的方法

比如用 flag[0] 表示 flag[a], flag[1] 表示 flag[a+1]....，flag[b-a] 表示 flag[b], 那么开一个一百万大小的数组就够了

素数筛法的扩展

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

```
const int N = 1000010;

bool f[N];
int p[N], tot;
bool flag[N];

void init () {
  for (int i = 2; i < N; i++ ) if (!f[i]) {
    p[tot++] = i;
    for (int j = i + i; j < N; j += i) {
      f[j] = true;
    }
  }
}
int main() {
  init();
  int t, ca = 1;
  scanf("%d", &t);
  while (t--) {
    int a, b;
    scanf("%d%d", &a, &b);
    memset(flag, false, sizeof(flag));
    for (int i = 0; i < tot; i++) {
      if (p[i] * p[i] > b) {
        break;
      }
      int g = ceil(1.0 * a / p[i]);
      //j 没开long long会RE
      for (long long j = max(2LL*p[i], 1LL * g * p[i]); j <= b; j += p[i]) {
        flag[j - a] = true;
      }
    }
    int ret = 0;
    for (int i = a; i <= b; i++) {
      ret += (flag[i - a] == false);
    }
```

```
    if (a == 1) {
      ret--;
    }
    printf("Case %d: %d\n", ca++, ret);
  }
  return 0;
}
```

0.5 E 题

求 $\sum\limits_{i=1}^{n}\sum\limits_{j=i}^{n}[lcm(i,j)=n]$

这个式子本质上是在求有多少的不同的 (i,j) 对 $(i \le j)$ 满足 $lcm(i,j)=n$, 我们转换一下，如果对 i,j 之间的大小关系不做限制，答案更加好求, 但是要减去一些多余的

$$2*ans=\sum_{i=1}^{n}\sum_{j=1}^{n}[lcm(i,j)=n]+\sum_{i=1}^{n}[lcm(i,i)=n]$$

$$ans=(\sum_{i=1}^{n}\sum_{j=1}^{n}[lcm(i,j)=n]+1)/2$$

假设我们对 n 分解质因数可以得到 $n=\prod_{i=1}^{k}p_i^{a_i}$, 那么对于每个 p_i 来说，要么 i 里面取到 a_i 次，要么 j 里面取到 a_i 次，一共有 $2*a_i+1$ 种选择，所以答案就是 $((\prod 2*a_i+1)+1)/2$

最小公倍数应用题

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

const int N = 10000010;

bool flag[N];

int tot, p[N / 10];

void init(int n) {
```

```
    for (int i = 2; i <= n; i++) {
        if (!flag[i]) {
            p[tot++] = i;
        }
        for (int j = 0; j < tot; j++) {
            if (i * p[j] > n) {
                break;
            }
            flag[i * p[j]] = true; //p[j]是i * p[j]的最小素因子
            if (i % p[j] == 0) { // 再往后p[j]就不是i * p[j]的最小素因子了
                break;
            }
        }
    }
}

int main() {
  init(10000000);
  int t, ca = 1;
  scanf("%d", &t);
  while (t--) {
    printf("Case %d: ", ca++);
    long long n;
    scanf("%lld", &n);
    long long ret = 1;
    for (int i = 0; i < tot; i++) {
      if (p[i] * p[i] > n) {
        break;
      }
      if (n % p[i] == 0) {
        int cnt = 0;
        while (n % p[i] == 0) {
          n /= p[i];
          cnt++;
        }
        ret = ret * (2LL * cnt + 1);
```

```
      }
    }
    if (n > 1) {
      ret = ret * 3;
    }
    ret = (ret + 1) / 2;
    printf("%lld\n", ret);
  }
  return 0;
}
```

0.6 F 题

考察整除分块以及四次方和公式

平方和公式：$\sum_{k=1}^{n} k^2 = \frac{n*(n+1)*(2n+1)}{6}$

立方和公式：$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}$

四次方和公式：$\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

公式计算

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

int m;
long long get_sum(long long n) {
    long long ret = n % m * (n + 1) % m * (2 * n + 1) % m * (n % m * n % m *
        3 % m + n * 3 % m - 1) ;
    return (ret % m + m ) % m;
}

int main () {
    int t;
    long long n;
    scanf("%d", &t);
    while (t--) {
```

```
        scanf("%lld%d", &n, &m);
        m *= 30;
        long long ret = 0;
        long long j;
        for (long long i = 1; i <= n; i = j + 1) {
            j = n / (n / i);
            long long value = n / i;
            ret += value * ( (get_sum(j) - get_sum(i - 1)) % m + m ) % m;
            ret %= m;
        }
        ret /= 30;
        printf("%lld\n", ret);
    }
    return 0;
}
```

0.7 G 题

这个题主要考察贡献的计算，在信息学比赛中十分常见，即计算某个东西贡献了几次

对于这个题来说我们就是要计算某个 $A[i]$ 最终被算到了几次即可

我们发现每一个 $A[i]$ 被算到的次数都是一样的，所有数的总次数为 $k*n^k$, 所以每个数被算到的次数为 $k*n^{k-1}$

那么答案就是 $k*n^{k-1}*sum$ (sum 表示 A 数组的和)

算贡献

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

int pow_mod(int a, int b, int c) {
  int ret = 1;
  while (b) {
    if (b & 1) {
      ret = 1LL * ret * a % c;
    }
```

```
    b >>= 1;
    a = 1LL * a * a % c;
  }
  return ret;
}

int a[1010];
int main() {

  int t, ca = 1;
  scanf("%d", &t);
  while (t--) {
    printf("Case %d: ", ca++);
    int n,k,md;
    scanf("%d%d%d", &n, &k, &md);
    long long sum = 0;
    for (int i = 0; i < n; i++) {
      scanf("%d", &a[i]);
      sum += a[i];
      sum %= md;
    }
    printf("%d\n", 1LL * k * pow_mod(n, k - 1, md) % md * sum % md );
  }
  return 0;
}
```


0.8 H 题

我们需要求

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} lcm(i,j)$$

令 $f(n)=\sum_{i=1}^{n-1} lcm(i,n)$

那么 $ans=\sum_{i=2}^{n}\sum_{j=1}^{i-1} lcm(j,i)=\sum_{i=2}^{n} f(i)$

所以只要能比较快的求出 $f(n)$，然后求一个前缀和就可以了

$$f(n)=\sum_{i=1}^{n-1}lcm(i,n)=\sum_{i=1}^{n-1}\frac{in}{gcd(i,n)}=n\sum_{g|n,g<n}\sum_{i=1}^{n-1}\frac{i}{g}[gcd(i,n)=g]$$

$$=n\sum_{g|n,g<n}\sum_{i=1}^{\frac{n}{g}}i[gcd(i,\frac{n}{g})=1]=n(\sum_{g|n,g<n}\frac{\frac{n}{g}\varphi(\frac{n}{g})}{2})$$

$g<n$ ，所以 $\frac{n}{g}\neq 1$, 所以可以枚举所有的因子，再枚举所有的倍数来求和

另外注意：对 2^{64} 取模，本质上就是把结果保存成 unsigned long long 类型就行了

最小公倍数应用

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

const int N = 3000010;

unsigned long long f[N];
unsigned long long phi[N];

void init() {
  for (int i = 1; i < N; i++) {
    phi[i] = i;
  }
  for (int i = 1; i < N; i++) {
    for (int j = i + i; j < N; j += i) {
      phi[j] -= phi[i];
    }
  }
  for (int g = 2; g < N; g++) {
    for (int n = g; n < N; n += g) {
      f[n] += 1ULL * phi[g] * g / 2 * n; //小心这里容易爆炸，先除再乘
    }
  }
  for (int i = 2; i < N; i++) {
    f[i] = f[i - 1] + f[i];
  }
}
```

```
int main() {
  init();
  int t, ca = 1;
  scanf("%d", &t);
  while (t--) {
    int n;
    scanf("%d", &n);
    printf("Case %d: %llu\n", ca++, f[n]);
  }
  return 0;
}
```