具体数学初级班第一周参考答案

517 老师

2020 年 5 月 1 日

1 理论部分

1.1 证明题 1

反证法. 假设不唯一，即存在 $a, b > 0$

$a = k_1 * b + r_1$ $(0 \leq r_1 < b)$

$a = k_2 * b + r_2 (0 \leq r_2 < b)$

$k_1 \neq k_2, r_1 \neq r_2$, 假设 $r_1 < r_2$, 那么 $k_1 > k_2$

那么 $(k_1 - k_2) * b + (r_1 - r_2) = 0$

$b = \frac{r_2 - r_1}{k_1 - k_2}$

所以 b 是 $r_2 - r_1$ 的因子，很显然 $r_2 - r_1 < b$, 那么只可能是 $r_1 = r_2$

1.2 证明题 2

反证法. 假设不唯一

$n = p_1 p_2 ... p_x = q_1 q_2 ... q_y$

假设 $p_1 < q_1$

因为 $p_1 | q_1 q_2 ... q_y$

又因为如果质数 $p|ab$，则要么 $p|a$ 和 $p|b$ 中一定有一个成立

所以 $q_1 q_2 ... q_y$ 中一定包含 p_1

产生了矛盾

1.3 证明题 3

假设 k 不是 $\varphi(n)$ 的因子, 并且 $k < \varphi(n)$

假设 $\varphi(n) = m * k + r\ (r < k)$
那么
$a^{\varphi(n)} = a^{mk+r} \equiv 1 \pmod{n}$
因此 $a^{mk} * a^r \equiv 1 \pmod{n}$
因此 $a^r \equiv 1 \pmod{n}$
但是因为 $r < k$, 这与 k 是最小的满足条件的数产生了矛盾

2 编程练习

以下不按难度排列, 提供大致意思翻译，具体输入输出的格式见题面
ABCFHIJ 可以先做
有能力的同学可以同时做其他题

2.1 A 题

这是一个找规律题，我们可以观察到从里往外每一圈的数字个数是 $1, 3, 5, 7...$
于是我们首先可以定位 n 在第几圈，根据等差数列求和公式我们知道前 i 圈一共有 i^2 个数
然后是计算顺时针还是逆时针方向的
最后是横着还是竖着

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

int main() {
  int T;
  scanf("%d", &T);
  long long n;
  int ca = 1;
  while (T--) {
    scanf("%lld", &n);

    //先计算在第几圈
    long long best = sqrt(1.0 * n);
    if (best * best != n) {
```

```
      best++;
    }

    //计算n所在的圈前面一共有多少个数
    long long pre = (best - 1) * (best - 1);

    //计算n之前的数的差距
    long long delta = n - pre;

    printf("Case %d: ", ca++);
    if (best & 1) { //奇数圈，逆时针
      if (delta <= best) {
        printf("%lld %lld\n", best, delta);
      } else {
        printf("%lld %lld\n", best - (delta - best), best);
      }
    } else { //偶数圈，顺时针
      if (delta <= best) {
        printf("%lld %lld\n", delta, best);
      } else {
        printf("%lld %lld\n", best, best - (delta - best));
      }
    }
  }
  return 0;
}
```

2.2 B 题

这是一道比较基础的欧拉函数练习题，预处理欧拉函数后，再预处理前缀和，每次询问 $O(1)$ 搞定

注意此题可能爆 long long，于是代码里使用了 unsigned long long

欧拉函数

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

```
using namespace std;

const int N = 5000010;

bool flag[N];
int p[N / 10], tot;
unsigned long long phi[N];

void init(int n) {
  phi[1] = 1;
  for (int i = 2; i <= n; i++) {
    if (!flag[i]) {
      p[tot++] = i;
      phi[i] = i - 1; // 素数的欧拉函数值等于素数-1
    }
    for (int j = 0; j < tot; j++) {
      if (i * p[j] > n) {
        break;
      }
      flag[i * p[j]] = true;
      if (i % p[j] == 0) {
        phi[i * p[j]] = phi[i] * p[j];//性质3
        break;
      }
      phi[i * p[j]] = phi[i] * (p[j] - 1);// 性质4
    }
  }
  for (int i = 1; i <= n; i++) {
    phi[i] = phi[i - 1] + 1LL * phi[i] * phi[i];
  }
}
int main() {
  init(5000000);
  int T, ca = 1, a, b;
  scanf("%d", &T);
  while (T--) {
```

```
    scanf("%d%d", &a, &b);
    printf("Case %d: %llu\n", ca++, phi[b] - phi[a - 1]);
  }
  return 0;
}
```

2.3 C 题

这是一道求逆元的练习题

n 个不同的物体里面选取 k 个一共有 $\binom{n}{k}$ 种取法, 即 C_n^k

又可以表示为 $\frac{n!}{k!(n-k)!}$

因为有除法，需要办计算边取模，因此需要用到逆元

具体做的时候我们可以预处理出 $f[i]$ 表示 $i!\pmod{1)000003}$

在计算的时候直接计算 $f[n] * inv(f[k]) * inv(f[n-k])$ 即可

inv 表示逆元, 由于此题模数是质数，所以可以使用费马小定理

阶乘逆元

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

const int N = 1000010;
const int md = 1000003;

int f[N];

void init() {
  f[0] = 1;
  for (int i = 1; i < N; i++) {
    f[i] = 1LL * f[i - 1] * i % md;
  }
}

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 inv(int a) {
  return pow_mod(a, md - 2, md);
}

int main() {
  init();
  int T, ca = 1;
  scanf("%d", &T);
  while (T--) {
    printf("Case %d: ", ca++);
    int n, k;
    scanf("%d%d", &n, &k);
    //注意中间结果可能比较大，需要边乘边取模
    printf("%d\n", 1LL * f[n] * inv(f[k]) % md * inv(f[n - k]) % md );
  }
  return 0;
}
```


2.4 D 题

这个题涉及到对扩展欧几里得通解的考察，有一定的细节，需要仔细讨论各类情况

对于一个通解 $x = x_0 + Bt\ y = y_0 - At$, 我们需要求有多少个 t 能使得

$x_1 \le x \le x_2, y_1 \le y \le y_2$

我们可以分别对 x 和 y 求出两个 t 的合法区间 $[l, r]$，以及 $[w, e]$

最后求两个区间的并集即可

此题很考验代码细节的基本功

扩展欧几里得

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

#define ll long long

ll exgcd(ll a, ll b, ll &x, ll &y) {
    if(b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= (a / b)*x;
    return d;
}

int main() {
    int T;
    int cnt = 0;
    scanf("%d", &T);
    while(T--)
    {
        ll a, b, c, x1, x2, y1, y2;
        scanf("%lld%lld%lld", &a, &b, &c);
        scanf("%lld%lld%lld%lld",&x1, &x2, &y1, &y2);
        c = -c;
        if(a < 0)
        {
            a = -a;
            swap(x1, x2);
            x1 = -x1;
            x2 = -x2;
        }
```

```
if(b < 0)
{
    b = -b;
    swap(y1, y2);
    y1 = -y1;
    y2 = -y2;
}
printf("Case %d: ", ++cnt);

ll x0 = 0 , y0 = 0;
ll g = exgcd(a, b, x0, y0);
if(g!=0 && c % g != 0)
{
    printf("0\n");
    continue;
}
if(a == 0 && b == 0)
{
    if(!c)
        printf("%lld\n", (x2 - x1 + 1) * (y2 - y1 + 1));
    else
        printf("0\n");
    continue;
}
if(a == 0)
{
    if(c / b >= y1 && c / b <= y2)
        printf("%lld\n", x2 - x1 + 1);
    else
        printf("0\n");
    continue;
}
if(b == 0)
{
    if(c / a >= x1 && c / a <= x2)
        printf("%lld\n", y2 - y1 + 1);
```

```
            else
                printf("0\n");
            continue;

        }
        //以上特判
        x0 = x0 * c / g;
        y0 = y0 * c / g;
        a /= g;
        b /= g;
        ll l = ceil((double)(x1 - x0)/(double)(b));
        ll r = floor((double)(x2 - x0)/(double)(b));
        ll w = ceil((double)(y0 - y2)/(double)(a));
        ll e = floor((double)(y0 - y1)/(double)(a));
        ll q = max(l, w);
        ll p = min(r, e);
        if(p < q)
            printf("0\n");
        else printf("%lld\n", p - q + 1);
    }
    return 0;
}
```


2.5 E 题

有一个结论是，平面直角坐标系中，假设一条线段的两个端点都在整数格点上，坐标分别为 $(x_1, y_1), (x_2, y_2)$

那么这条线段上的整数格点的数量为 $gcd(|x_1 - x_2|, |y_1 - y_2|) + 1$

为什么呢，大家可以尝试证明一下

线段整点数量

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

long long gcd(long long a, long long b) {
```

```
  return !b ? a : gcd(b, a % b);
}

int main() {
  int T, ca = 1;
  scanf("%d", &T);
  while (T--) {
    printf("Case %d: ", ca++);
    long long ax,ay,bx,by;
    scanf("%lld%lld%lld%lld", &ax, &ay, &bx, &by);
    printf("%lld\n", gcd(abs(bx - ax), abs(by - ay)) + 1);
  }
  return 0;
}
```

2.6 F 题

简单题，预处理每个数的因子数量然后排序即可

约数的个数

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

struct node {
  int cnt;
  int value;
}in[1010];

bool cmp(node a, node b) {
  return a.cnt < b.cnt || a.cnt == b.cnt && a.value > b.value;
}

void init() {
  for (int i = 1; i <= 1000; i++) {
    int cnt = 0;
```

```
    for (int j = 1; j * j <= i; j++) {
      if (i % j == 0) {
        cnt++;
        if (i / j != j) {
          cnt++;
        }
      }
    }
    in[i].cnt = cnt;
    in[i].value = i;
  }
  sort (in + 1, in + 1001, cmp);
}

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


2.7 G 题

这是一个中国剩余定理的模版题，直接根据中国剩余定理的原理去做即可
注意数据范围，最终结果要 long long

中国剩余定理

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

```
long long exgcd(long long a, long long b, long long &x, long long &y) {
  long long d = a;
  if(b != 0) {
    d = exgcd(b, a % b, x, y);
    x -= (a / b) * y;
    std::swap(x, y);
  } else {
    x = 1; y = 0;
  }
  return d;
}

long long crt(int n, int a[], int m[]) {
  long long mul = 1;
  for (int i = 0; i < n; i++) {
    mul = mul * m[i];
  }
  long long ret = 0;
  for (int i = 0; i < n; i++) {
    long long x, y;
    exgcd(mul / m[i], m[i], x, y);
    x = (x % m[i] + m[i]) % m[i];
    ret += mul / m[i] * x * a[i];
    ret %= mul;
  }
  return ret;
}

int main() {
  int T, ca = 1;
  scanf("%d", &T);
  while (T--) {
    int n;
    scanf("%d", &n);
    int m[15], a[15];
```

```
    for (int i = 0; i < n; i++) {
      scanf("%d%d", &m[i], &a[i]);
    }
    long long ret = crt(n, a, m);
    printf("Case %d: %lld\n", ca++, ret);
  }
  return 0;
}
```

2.8 H 题

两个数的最小公倍数就是质因子分解后的每个质因子的幂次取较大值

可以先求出 $x = lcm(a, b)$

然后找最小的 y 满足 $lcm(x, y) = L$

那么对于某个质因子，如果 x 这里跟 L 已经相等了，y 就可以不要这个质因子，否则应该跟 L 一样

对于每个质因子就这样贪心的去构造即可

最小公倍数 + 分解质因子

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

const int N = 1000010;
bool flag[N];
int p[N], tot;

void init() {
  for (int i = 2; i < N; i++) if(!flag[i]){
    p[tot++] = i;
    for (int j = i + i; j < N; j += i) {
      flag[j] = true;
    }
  }
}
```

```
int gcd(int a, int b) {
  return !b ? a : gcd(b, a % b);
}

vector <pair<long long,int> > split(long long L) {
  vector <pair<long long, int> > num;
  for (int i = 0; i < tot; i++) {
    if (1LL * p[i] * p[i] > L) {
      break;
    }
    int cnt = 0;
    while (L % p[i] == 0) {
      L /= p[i];
      cnt++;
    }
    if (cnt > 0) {
      num.push_back(make_pair(p[i], cnt));
    }
  }
  if (L > 1) {
    num.push_back(make_pair(L, 1));
  }
  return num;
}

int main() {
  init();
  int T, ca = 1;
  scanf("%d", &T);

  while (T--) {
    printf("Case %d: ", ca++);
    int a, b;
    long long L;
    scanf("%d%d%lld", &a, &b, &L);
```

```
long long x = 1LL * a * b / gcd(a,b);
if (L % x != 0) {
  printf("impossible\n");
  continue;
}
//对L分解质因数，枚举L的每种质因子去分析
//对于某种质因子p，如果a b中的最高次幂与L相同，就可以不取，否则就要取最高次幂
//注意无解的判断

vector <pair<long long,int> > num = split(L);
bool error = false;
long long ret = 1;
for (int i = 0; i < (int)num.size(); i++) {
  int ta = a, tb = b;
  int cnta = 0, cntb = 0;
  while (ta % num[i].first == 0) {
    ta /= num[i].first;
    cnta ++;
  }
  while (tb % num[i].first == 0) {
    tb /= num[i].first;
    cntb++;
  }
  if (max(cnta, cntb) > num[i].second) {
    error = true;
    break;
  } else if (max(cnta, cntb) < num[i].second) {
    for (int j = 0; j < num[i].second; j++) ret *= num[i].first;
  }
}
if (!error) {
  printf("%lld\n", ret);
} else {
  printf("impossible\n");
}
```

```
  }
  return 0;
}
```

2.9 I 题

考察基本的余数的性质

判断是否整除

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

int main() {
  int T, ca = 1;
  scanf("%d", &T);

  while (T--) {
    printf("Case %d: ", ca++);
    int d;
    char s[220];
    scanf("%s%d", s, &d);
    if (d < 0) d = -d;
    int n = strlen(s);
    long long ret = 0;
    for (int i = 0; i < n; i++) {
      if (s[i] == '-') continue;
      ret = ret * 10 + (s[i] -'0');
      ret = ret % d;
    }
    if (ret == 0) {
      printf("divisible\n");
    } else {
      printf("not divisible\n");
    }
```

```
  }
  return 0;
}
```

2.10 J 题

设首项为 a, 末项为 $a+k-1$, 等差数列求和可以得到

$2a-1=\frac{2n}{k}-k$

k 是 $2n$ 的一个因子，$\frac{2n}{k}$ 也是 $2n$ 的因子，而且 $\frac{2n}{k}$ 与 k 是一奇一偶

无论是奇数减去偶数还是偶数减去奇数, 右边的值肯定要大于 0，实际上对于任意一个 $2n$ 的的奇数因子 x，如果它大于 $\sqrt{2n}$ k 就可以是一个偶数, $k=\frac{2n}{x}$, 如果它小于 $\sqrt{2n}$, $k=x$

因此我们只需要求出 $2n$ 有几个奇数因子即可，那么本质上就是 n 有几个奇数因子

奇数因子就是不包含质因子 2 的因子

做法就是先分解质因子，然后计算奇因子的个数

奇因子的个数

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

const int N = 10000010;

int p[N / 10], tot;
bool flag[N];
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;
```

```
      if (i % p[j] == 0) {
        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 && 1LL * p[i] * p[i] <= n; i++) {
      int cnt = 0;
      while (n % p[i] == 0) {
        if (i != 0) {
          cnt++;
        }
        n /= p[i];
      }
      ret *= (cnt + 1);
    }
    if (n > 1 && n != 2) ret *= 2;
    printf("%lld\n", ret - 1);
  }
  return 0;
}
```


2.11 K 题

题目已经给出了约数之和的表达式，相当于等比数列求和后相乘

写成另一种形式就是 $\sigma(n) = (1 + p_1 + {p_1}^2 + ... + {p_1}^{a_1}) * (1 + p_2 + {p_2}^2 + ... + {p_2}^{a_2})...$

我们发现只要有一项为偶数，这个数就为偶数了，这个比较难考虑，于是我们反过来考虑

就是总数减去所有项都是奇数的情况

素数除了 2 以外都是奇数

假如 $p_i = 2$, p_i 所在的项的和是 2

其他情况 a_i 必须是偶数

所以满足约数和为奇数的数要么自己本身就是一个完全平方数，要么就是 2 的幂次为奇数，即去掉一个 2 之后是完全平方数

综合分析题

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

int main() {
  int T, ca = 1;
  scanf("%d", &T);
  while (T--) {
    printf("Case %d: ", ca++);
    long long n;
    scanf("%lld", &n);
    printf("%lld\n", n - (int)sqrt(n) - (int)sqrt(n / 2));
  }
  return 0;
}
```