具体数学初级班第五周参考答案

517 老师

2020 年 5 月 29 日

0.1 A 题

因为 $a = c = 1$，所以问题转换为求 $[1, b/k]$ 与 $[1, d/k]$ 之间有多少对数的最大公约数为 1

那么我们可以枚举某个较小区间的一个数 x，去计算另外一个区间里面有多少的数跟 x 的 gcd 为 1

本质上就是计算 $[1, n]$ 里面有多少数与 r 互质这样一个问题

求 1-n 里面有多少个数与 r 互质

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
int prm[N];
int tot=0;
bool flag[N];
//筛素数
void init() {
  memset(flag,false,sizeof(flag));
  for(int i=2;i<N;i++) {
    for(int j=2*i;j<N;j+=i) {
      flag[j]=true;
    }
  }
  for(int i=2;i<N;i++) {
    if(!flag[i])
      prm[++tot]=i;
```

```
  }
}

int sum;
int p[50];

//容斥原理：枚举每一种质数乘积的组合
void dfs(int dep, int tot, int mul, int has, int n) {
  if (dep == tot) {
    if (has == 0) {
      return ;
    }
    if (has % 2 == 1) {
      sum += n / mul;
    } else {
      sum -= n / mul;
    }
    return ;
  }
  dfs(dep + 1, tot, mul * p[dep], has + 1, n);
  dfs(dep + 1, tot, mul, has, n);
}

//求[1,n]里面有多少数与r互质
int solve(int r,int n){
  int cnt=0;
  int i;
  if(!flag[r]) {
    if(r>1)
      p[cnt++]=r;
  }
  else {
    for(i=1;i<=tot;i++) {
      if(r%prm[i]==0) {
        p[cnt++]=prm[i];
        while(r%prm[i]==0) {
```

```
        r/=prm[i];
      }
      if(!flag[r]) {
        if(r>1) {
          p[cnt++]=r;r=1;
        }
        break;
      }
    }
  }
  if(r>1) p[cnt++]=r;
 }
 sum=0;
 dfs(0, cnt, 1, 0, n);
 return n-sum;
}
int main(){
 int t;
 init();
 int a,b,c,d,e;
 int cases=1;
 scanf("%d",&t);
 while(t--){
  scanf("%d%d%d%d%d",&a,&b,&c,&d,&e);
  if(e==0||b<e||d<e)
  {
    printf("Case %d: 0\n",cases++);
    continue;
  }
  a = b < d ? b : d;
  b = b > d ? b : d;
  a /= e;
  b /= e;
  long long ans=0;
  for(int i=1;i<=a;i++)
  {
```

```
      if(i>1)
        ans+=solve(i,b)-solve(i,i-1);
      else ans+=b;
    }
    printf("Case %d: %lld\n",cases++,ans);
  }
  return 0;
}
```

0.2 B 题

不被 m 个数中的任何一个整除，可以理解为总数减去被 m 个数中的至少一个数整除

这个可以用容斥原理来做，所有一个数的倍数减去两个数的倍数加上三个数的倍数。。

注意，多个数的倍数实际上就是最小公倍数的倍数

容斥

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

int a[16];
int n,m;
int sum;

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

long long lcm(long long a, long long b) {
  return a * b / gcd(a, b);
}

void dfs(int dep, int tot, long long mul, int has, int n) {
  if (dep == tot) {
```

```
    if(has == 0) {
      return ;
    }
    if (has % 2 == 1) {
      sum += n / mul;
    } else {
      sum -= n / mul;
    }
    return ;
  }
  dfs(dep + 1, tot, lcm(mul, a[dep]), has + 1, n);
  dfs(dep + 1, tot, mul, has, n);
}

int main() {
  while (scanf("%d%d", &n, &m) == 2) {
    for (int i = 0; i < m; i++) {
      scanf("%d", &a[i]);
    }
    sum = 0;
    dfs(0, m, 1, 0, n);
    printf("%d\n", n - sum);
  }
  return 0;
}
```


0.3 C 题

我们发现对于没有出现的数字可以分为两类，一类是自己的数所在的位置已经被别的数占据了，那么他可以随便放

另一类是自己的数所在的位置还没有放也就是-1，那么他就不能放这个位置

所以我们可以假设两类数一共有 s 个，有 a 个数可以随便放，那么 $s-a$ 个数就要错排

我们可以采用容斥：所有数随便排的方案减去至少有一个数放在原来位置的方案加上至少两个数放在原来位置。。。

$$s! - \sum_{i=1}^{s-a} (-1)^i \binom{s-a}{i} (s-i)!$$

错排变形

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

const int md = 1000000007;

const int N = 2020;
int c[N][N];
int pos[N];
int p[N];
int fac[N];

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

int main() {
  init();
  int n;
  scanf("%d", &n);
  int s = 0;
  memset(pos, -1, sizeof(pos));
  for (int i = 1; i <= n; i++) {
    scanf("%d", &p[i]);
    if (p[i] == -1) {
      s++;
    } else {
```

```
      pos[p[i]] = i;
    }
  }

  int a = 0;// 可以随便放的数的数量
  for (int i = 1; i <= n; i++) {
    if (pos[i] == -1 && p[i] != -1) {
      a++;
    }
  }

  fac[0] = 1;
  for (int i = 1; i <= n; i++) {
    fac[i] = 1LL * fac[i - 1] * i % md;
  }

  //枚举几个数在自己位置
  int ret = fac[s];
  for (int i = 1; i <= s-a; i++) {
    if (i & 1) {
      ret -= 1LL * c[s-a][i] * fac[s - i] % md;
      ret = (ret % md + md) % md;
    } else {
      ret += 1LL * c[s-a][i] * fac[s - i] % md;
      ret = ret % md;
    }
  }
  printf("%d\n", ret);


  return 0;
}
```

0.4 D 题

题解直接在代码注释中

猴子

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

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

inline void add(int &a, int b) {
  a += b;
  if (a >= md) a -= md;
}

inline void sub(int &a, int b) {
  a -= b;
  if (a < 0) a += md;
}

inline int mul(int a, int b) {
  return (long long) a * b % md;
}

inline int power(int a, int b) {
  int res = 1;
  while (b > 0) {
    if (b & 1) {
      res = mul(res, a);
    }
    a = mul(a, a);
    b >>= 1;
  }
  return res;
}
```

```
inline int inv(int a) {
  return power(a, md - 2);
}

const int N = 200010;
int fac[N], inverse[N];

void init () {
  fac[0] = 1;
  inverse[0] = 1;
  for (int i = 1; i < N; i++) {
    fac[i] = 1LL * fac[i - 1] * i % md;
    //预处理阶乘的逆元
    inverse[i] = mul(inverse[i - 1], inv(i));
  }
}

int bino(int n, int m) {
  int ret = fac[n];
  ret = mul(ret, inverse[m]);
  ret = mul(ret, inverse[n - m]);
  return ret;
}

//n个球放入$m$个盒子里，盒子可以为空
int ball_to_boxes(int n, int m) {
  return bino(n + m - 1, m - 1);
}

void solve (int n, int m) {
  if (n == 1 || m == 1) {
    printf("1\n");
    return ;
  }
  int ret = 0;
  //枚举第一只猴子拿了多少桃子
```

```
  for (int first_monkey = 1; first_monkey <= n; first_monkey++) {
    int mi = (n - first_monkey) / (m - 1) + ( (n - first_monkey) % (m - 1) !=
        0) ;
    //剩下的桃子按照平均分的原则如果都比第一只猴子大，就继续枚举
    if (mi >= first_monkey) {
      continue;
    }
    int f = 1;
    //容斥原理，枚举有几个猴子的桃数大于等于第一只猴子
    //也就是至少0个猴子比第一只多-至少1只猴子比第一只多+至少两只...

    //可以发现big_equal变量的大小的级别是 n / first_monkey
    //那么总体复杂度就是n/1+n/2+...为调和级数 O(nlogn)
    for (int big_equal = 0; big_equal <= m - 1; big_equal++) {
      int nn = n - (big_equal + 1) * first_monkey;
      //先分配跟第一只猴子一样的桃子，剩下nn个桃子
      if (nn < 0) {
        break;
      }
      //从m-1只猴子中选取big_equal只，乘上将剩下的桃子分给m-1只猴子的方案数
      int tmp = mul(bino(m - 1, big_equal), ball_to_boxes(nn, m - 1));
      if (f == 1) {
        add(ret, tmp);
      } else {
        sub(ret, tmp);
      }
      f = -f;
    }
  }
  printf("%d\n", ret);
}

int main () {
  init();
  int t, n, m;
  scanf("%d", &t);
```

```
  while (t--) {
    scanf("%d%d", &n, &m);
    solve (n, m);
  }
  return 0;
}
```