Bedao contest
id::02

MULDIV

- Thuật toán O(TN): với mỗi N, ta duyệt qua tất cả các số i ϵ [1;N] nếu i là ước của N thì cập nhật vào kết quả.

- Thuật toán O($T\sqrt{N}$): nhận xét với mỗi ước số i <$\sqrt{N}$ của N thì j = N/i>$\sqrt{N}$ cũng là ước của N.

➔ ta chỉ cần chạy từ 1 đến $\sqrt{N}$ để tìm tất cả các ước của N.

```
cin>>T;
 for(test=1;test<=T;test++){
     ans=1;

    cin>>n;

    for(  i=1;i*i<n;i++) if(!n%i) ans=ans*n%998244353; /// i*(n/i) = n

     if(int(sqrt(n))==sqrt(n)) ans=ans*int(sqrt(n))%998244353;

     cout<<ans<<"\n";

  }
```

- Thuật toán O($Tlog(N)$):

- Từ thuật toán O($T\sqrt{N}$) → Nếu gọi m là số ước của N.

  - Nếu N không phải là số chính phương; ans = $N^{m/2}$ mod 998244353.
  - Nếu N là số chính phương; ans =$N^{m/2}\sqrt{N}$ mod 998244353.

- Vấn đề còn lại là với mỗi N thì làm sao để tìm m (1) và tính $N^{m/2}$ (2) trong độ phức tạp log.

  + Với vấn đề (1), Giả sử N được phân tích thành thừa số nguyên tố như sau:

  $$x_1^{p1}.\ x_2^{p2}\ldots\ x_k^{pk}$$

  - → Số ước của N = (p1+1)(p2+1)...(pk+1)

  Như vậy ta đơn giản hóa vấn đề trở thành đếm phân tích N ra thừa số nguyên tố trong O(log(N)). Vì N≤10^7 nên ta có thể sử dụng sàng.

  + Vấn đề (2) bạn có thể tham khảo tại đây.

- Tiện thể mình giới thiệu đến các bạn VNOI wiki
- Đây là thư viện về các vấn đề tin học nói chung, giải thuật và cấu trúc dữ liệu nói riêng.
- Bạn nên tự mình khám phá kho tàng tri thức nằm trong thư viện đầy tâm huyết của các anh đi trước đi đã xây dựng.