Bài yêu cầu tính a^b sử dụng thuật toán luỹ thừa nhị phân

Lý do chúng ta sử dụng luỹ thừa nhị phân (Binary Exponentiation) trong bài toán này là vì nó giúp tăng tốc độ tính toán đáng kể.

Trong trường hợp thông thường, để tính a^b, chúng ta sẽ phải thực hiện $b - 1$ phép nhân. Điều này có thể tốn rất nhiều thời gian nếu b là một số lớn.

Tuy nhiên, với luỹ thừa nhị phân, chúng ta có thể giảm số lượng phép nhân xuống còn log2b. Điều này được thực hiện bằng cách biểu diễn b dưới dạng nhị phân và chỉ tính toán các lũy thừa của a mà ứng với các bit 1 trong biểu diễn nhị phân của b.

Ví dụ, để tính 2^8, thay vì phải nhân 2 với chính nó 7 lần, chúng ta chỉ cần tính 2^2, sau đó bình phương kết quả để có 2^4, và cuối cùng bình phương kết quả một lần nữa để có 2^8. Như vậy, chúng ta chỉ cần 3 phép nhân thay vì 7.

Do đó, luỹ thừa nhị phân giúp giảm thiểu đáng kể thời gian tính toán, đặc biệt khi làm việc với các số lớn. Đây là lý do chính tại sao chúng ta sử dụng luỹ thừa nhị phân trong bài toán này.

Code :

```
Cách 1:
#include<iostream>
using namespace std;
const int mod =1e9+7;
long long exp(long long a,long long b){
   if(b==1) return a;
   if(b==0) return 1;
   if(b%2) return ((exp(a,b/2)%mod *exp(a,b/2)%mod)%mod*a)%mod ;
   else return (exp(a,b/2)%mod *exp(a,b/2)%mod)%mod;
}
int main(){
   long long a, n;
   cin >> a>> n;
   cout<< exp(a,n);
}
Cách 2:
#include<iostream>
using namespace std;
const int mod =1e9+7;
int main(){
   long long a, n;
   cin >> a>> n;
   long long temp = 1;
   while (n > 0) {
```

```
    if (n & 1)
        temp = temp%mod * a;
    temp%=mod;
    a = a * a %mod;
    n \= 2;
  }
  cout<<temp;
}
```

Việc lấy dư sau mỗi lần nhân trong thuật toán Binary Exponentiation giúp giảm bớt kích thước của số đang xử lý, giúp tránh tràn số khi làm việc với các số rất lớn. Điều này đặc biệt quan trọng khi bạn đang làm việc với các số nguyên có kích thước cố định, như `int` hoặc `long long` trong C++.

Cuối cùng, việc này cũng tuân theo tính chất của phép modulo:

$$(a \cdot b) \% m = \big((a \% m) \cdot (b \% m)\big) \% m.$$

Điều này cho phép chúng ta lấy dư sau mỗi lần nhân mà không làm thay đổi kết quả cuối cùng.