斐波那契数列

递归

```
#include<iostream>
#include<cstring>
using namespace std;
const int inf=7+1e9;
int t;
long long m;
struct matrix{
    int row,column;
    long long val[5][5];
    matrix(){
        row=column=2;
        memset(val,0,sizeof(val));
    };
    void set_fbnq(int a,int b){
        row=a,column=b;
        if(row==2&&column==2){
            val[1][1]=val[1][2]=val[2][1]=1;
            val[2][2]=0;
        };
    };
    void get_mod(){
        for(int r=1;r<=row;r++)
            for(int c=1;c<=column;c++)
                val[r][c]%=inf;
    };
}A;
matrix operator *(matrix a,matrix b){
    matrix ans;
    if(a.column!=b.row) return ans;
    for(int r=1;r<=a.row;r++)
        for(int c=1;c<=b.column;c++)
            for(int i=1;i<=a.column;i++)
                ans.val[r][c]+=a.val[r][i]*b.val[i][c],ans.get_mod();
    return ans;
}
matrix matrix_pow(long long n){
    if(n==1) return A;
    if(n&1){
        matrix ans=A*matrix_pow(n-1);
        ans.get_mod();
        return ans;
    };
    matrix ans=matrix_pow(n>>=1);
    ans=ans*ans;
    ans.get_mod();
    return ans;
}
long long get_fbnq(long long  n){
    matrix ans=matrix_pow(n-1);
    return ans.val[1][1];
```

```
}
int main(){
    A.set_fbnq(2,2);
    cin>>t;
    for(int i=1;i<=t;i++){
        cin>>m;
        cout<<get_fbnq(m)<<endl;
    };
}
```

迭代

```
#include<iostream>
#include<cstring>
using namespace std;
const int inf=7+1e9;
int t;
long long m;
struct matrix{
    int row,column;
    long long val[5][5];
    matrix(){
        row=column=2;
        memset(val,0,sizeof(val));
    };
    void set_fbnq(int a,int b){
        row=a,column=b;
        if(row==2&&column==2){
            val[1][1]=val[1][2]=val[2][1]=1;
            val[2][2]=0;
        };
    };
    void get_mod(){
        for(int r=1;r=row;r++)
            for(int c=1;c<=column;c++)
                val[r][c]%=inf;
    };
    void set_e(){
        val[1][1]=val[2][2]=1;
        val[1][2]=val[2][1]=0;
    };
}A;
matrix operator *(matrix a,matrix b){
    matrix ans;
    if(a.column!=b.row) return ans;
    for(int r=1;r<=a.row;r++)
        for(int c=1;c<=b.column;c++)
            for(int i=1;i<=a.column;i++)
                ans.val[r][c]+=a.val[r][i]*b.val[i][c],ans.get_mod();
    return ans;
}
matrix matrix_pow(long long n){
    matrix ans,w=A;
    ans.set_e();
    while(n){


```

```
        if(n&1) ans=ans*w;
        w=w*w;
        n>>=1;
    };
    return ans;

}
long long get_fbnq(long long n){
    matrix ans=matrix_pow(n-1);
    return ans.val[1][1];
}
int main(){
    A.set_fbnq(2,2);
    cin>>t;
    for(int i=1;i<=t;i++){
        cin>>m;
        cout<<get_fbnq(m)<<endl;
    };
    return 0;
}
```