

# 斐波那契数列

## 递归

```
1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  const int inf=7+1e9;
5  int t;
6  long long m;
7  struct matrix{
8      int row,column;
9      long long val[5][5];
10     matrix(){
11         row=column=2;
12         memset(val,0,sizeof(val));
13     };
14     void set_fbnq(int a,int b){
15         row=a,column=b;
16         if(row==2&&column==2){
17             val[1][1]=val[1][2]=val[2][1]=1;
18             val[2][2]=0;
19         };
20     };
21     void get_mod(){
22         for(int r=1;r<=row;r++)
23             for(int c=1;c<=column;c++)
24                 val[r][c]%=inf;
25     };
26 }A;
27 matrix operator *(matrix a,matrix b){
28     matrix ans;
29     if(a.column!=b.row) return ans;
30     for(int r=1;r<=a.row;r++)
31         for(int c=1;c<=b.column;c++)
32             for(int i=1;i<=a.column;i++)
33                 ans.val[r][c]+=a.val[r][i]*b.val[i][c],ans.get_mod();
34     return ans;
35 }
36 matrix matrix_pow(long long n){
37     if(n==1) return A;
38     if(n&1){
39         matrix ans=A*matrix_pow(n-1);
40         ans.get_mod();
41         return ans;
42     };
43     matrix ans=matrix_pow(n>>=1);
44     ans=ans*ans;
45     ans.get_mod();
46     return ans;
47 }
48 long long get_fbnq(long long n){
49     matrix ans=matrix_pow(n-1);
50     return ans.val[1][1];
```

```

51 }
52 int main(){
53     A.set_fbnq(2,2);
54     cin>>t;
55     for(int i=1;i<=t;i++){
56         cin>>m;
57         cout<<get_fbnq(m)<<endl;
58     };
59 }

```

## 迭代

```

1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  const int inf=7+1e9;
5  int t;
6  long long m;
7  struct matrix{
8      int row,column;
9      long long val[5][5];
10     matrix(){
11         row=column=2;
12         memset(val,0,sizeof(val));
13     };
14     void set_fbnq(int a,int b){
15         row=a,column=b;
16         if(row==2&&column==2){
17             val[1][1]=val[1][2]=val[2][1]=1;
18             val[2][2]=0;
19         };
20     };
21     void get_mod(){
22         for(int r=1;r=row;r++)
23             for(int c=1;c<=column;c++)
24                 val[r][c]%=inf;
25     };
26     void set_e(){
27         val[1][1]=val[2][2]=1;
28         val[1][2]=val[2][1]=0;
29     };
30 }A;
31 matrix operator *(matrix a,matrix b){
32     matrix ans;
33     if(a.column!=b.row) return ans;
34     for(int r=1;r<=a.row;r++)
35         for(int c=1;c<=b.column;c++)
36             for(int i=1;i<=a.column;i++)
37                 ans.val[r][c]+=a.val[r][i]*b.val[i][c],ans.get_mod();
38     return ans;
39 }
40 matrix matrix_pow(long long n){
41     matrix ans,w=A;
42     ans.set_e();
43     while(n){
44
45

```

```
46         if(n&1) ans=ans*w;
47         w=w*w;
48         n>>=1;
49     };
50     return ans;
51 }
52 }
53 long long get_fbnq(long long n){
54     matrix ans=matrix_pow(n-1);
55     return ans.val[1][1];
56 }
57 int main(){
58     A.set_fbnq(2,2);
59     cin>>t;
60     for(int i=1;i<=t;i++){
61         cin>>m;
62         cout<<get_fbnq(m)<<endl;
63     };
64     return 0;
65 }
```