# 矩阵快速幂

**概述**

矩阵快速幂主要用于加速运算，当递推的范围很大时，线性超时的情况下，可以考虑矩阵快速幂加速

**矩阵快速幂在斐波那契数列的运用**

我们知道，设,则 ，由于矩阵运算的可结合性，可使用类似快速幂的思想对运算进行加速

****

**代码**

1. #include <iostream>
2. #include <cstring>
3. #include <algorithm>
4. #define int long long
5. **using** **namespace** std;
6. **const** **int** N = 3;
7. **int** n,m;
9. **int** a[N][N]=
10. {
11. {0,1,0},
12. {1,1,1},
13. {0,0,1}
14. };
15. **int** res[N]={1,1,1};
16. **void** mul(**int** a[][N],**int** b[][N],**int** res[][N],**int** mod)//二维乘二维
17. {
18. **int** tmp[N][N]={0};//不能直接使用传入的res数组进行运算
19. **for**(**int** i=0;i<N;i++)
20. {
21. **for**(**int** j=0;j<N;j++)
22. {
23. **for**(**int** k=0;k<N;k++)
24. {
25. tmp[i][j]+=a[i][k]\*b[k][j];//i行k列 \* k行j列
26. tmp[i][j]%=mod;
27. }
28. }
29. }
30. memcpy(res,tmp,**sizeof**(tmp));
31. }
32. **void** mul(**int** a[],**int** b[][N],**int** res[],**int** mod)//一维乘二维
33. {
34. **int** tmp[N]={0};//不能直接使用传入的res数组进行运算
35. **for**(**int** i=0;i<N;i++)
36. {
37. **for**(**int** j=0;j<N;j++)
38. {
39. tmp[i]+=a[j]\*b[j][i];
40. tmp[i]%=mod;
41. }
42. }
43. memcpy(res,tmp,**sizeof**(tmp));
44. }
45. **signed** main()
46. {
47. cin>>n>>m;
48. n--;
49. **while**(n)
50. {
51. **if**(n&1) mul(res,a,res,m);
52. mul(a,a,a,m);
53. n>>=1;
54. }
55. cout<<res[2]<<endl;
56. **return** 0;
57. }