//f[]：可以取走的石子个数

//sg[]:0~n的SG函数值

//hash[]:mex{}

int f[N],sg[N],hash[N];

void getSG(int n)

{

int i,j;

memset(sg,0,sizeof(sg));

for(i=1;i<=n;i++)

{

memset(hash,0,sizeof(hash));

for(j=1;f[j]<=i;j++)

hash[sg[i-f[j]]]=1;

for(j=0;j<=n;j++) //求mes{}中未出现的最小的非负整数

{

if(hash[j]==0)

{

sg[i]=j;

break;

}

}

}

}

//注意 S数组要按从小到大排序 SG函数要初始化为-1 对于每个集合只需初始化1遍

//n是集合s的大小 S[i]是定义的特殊取法规则的数组

int s[110],sg[10010],n;

int SG\_dfs(int x)

{

int i;

if(sg[x]!=-1)

return sg[x];

bool vis[110];

memset(vis,0,sizeof(vis));

for(i=0;i<n;i++)

{

if(x>=s[i])

{

SG\_dfs(x-s[i]);

vis[sg[x-s[i]]]=1;

}

}

int e;

for(i=0;;i++)

if(!vis[i])

{

e=i;

break;

}

return sg[x]=e;

}

hdu  1848

题意：取石子问题，一共有3堆石子，每次只能取斐波那契数个石子，先取完石子者胜利，问先手胜还是后手胜

1. 可选步数为一系列不连续的数，用GetSG(计算)
2. 最终结果是所有SG值异或的结果

AC代码如下：

1. #include<stdio.h>
2. #include<string.h>
3. #define N 1001
4. //f[]：可以取走的石子个数
5. //sg[]:0~n的SG函数值
6. //hash[]:mex{}
7. **int** f[N],sg[N],hash[N];
8. **void** getSG(**int** n)
9. {
10. **int** i,j;
11. memset(sg,0,**sizeof**(sg));
12. **for**(i=1;i<=n;i++)
13. {
14. memset(hash,0,**sizeof**(hash));
15. **for**(j=1;f[j]<=i;j++)
16. hash[sg[i-f[j]]]=1;
17. **for**(j=0;j<=n;j++)    //求mes{}中未出现的最小的非负整数
18. {
19. **if**(hash[j]==0)
20. {
21. sg[i]=j;
22. **break**;
23. }
24. }
25. }
26. }
27. **int** main()
28. {
29. **int** i,m,n,p;
30. f[0]=f[1]=1;
31. **for**(i=2;i<=16;i++)
32. f[i]=f[i-1]+f[i-2];
33. getSG(1000);
34. **while**(scanf("%d%d%d",&m,&n,&p)!=EOF)
35. {
36. **if**(m==0&&n==0&&p==0)
37. **break**;
38. **if**((sg[m]^sg[n]^sg[p])==0)
39. printf("Nacci\n");
40. **else**
41. printf("Fibo\n");
42. }
43. **return** 0;
44. }

大整数

#include <cstdio>

#include <algorithm>

#include <cstring>

using namespace std;

struct BigInteger{

int A[25];

enum{MOD = 10000};

BigInteger(){memset(A, 0, sizeof(A)); A[0]=1;}

void set(int x){memset(A, 0, sizeof(A)); A[0]=1; A[1]=x;}

void print(){

printf("%d", A[A[0]]);

for (int i=A[0]-1; i>0; i--){

if (A[i]==0){printf("0000"); continue;}

for (int k=10; k\*A[i]<MOD; k\*=10) printf("0");

printf("%d", A[i]);

}

printf("\n");

}

int& operator [] (int p) {return A[p];}

const int& operator [] (int p) const {return A[p];}

BigInteger operator + (const BigInteger& B){

BigInteger C;

C[0]=max(A[0], B[0]);

for (int i=1; i<=C[0]; i++)

C[i]+=A[i]+B[i], C[i+1]+=C[i]/MOD, C[i]%=MOD;

if (C[C[0]+1] > 0) C[0]++;

return C;

}

BigInteger operator \* (const BigInteger& B){

BigInteger C;

C[0]=A[0]+B[0];

for (int i=1; i<=A[0]; i++)

for (int j=1; j<=B[0]; j++){

C[i+j-1]+=A[i]\*B[j], C[i+j]+=C[i+j-1]/MOD, C[i+j-1]%=MOD;

}

if (C[C[0]] == 0) C[0]--;

return C;

}

};

int main() {

BigInteger a, b;

a.set(1); b.set(1);

(a+b).print();

return 0;

}