

```

1  #include <algorithm>
2  #include <chrono>
3  #include <iostream>
4  #include <limits>
5  #include <vector>
6  using namespace std;
7
8  bool isOdd(uint64_t num)
9  {
10     return num & 0x1;
11 }
12
13 class bigInt
14 {
15 public:
16     uint64_t hi;
17     uint64_t lo;
18
19     bigInt()
20     {
21         hi = 0x0;
22         lo = 0x0;
23     }
24     bigInt(unsigned long long num)
25     {
26         hi = 0x0;
27         lo = num; // 赋值不会过大
28     }
29     bigInt(const bigInt &rhs)
30     {
31         hi = rhs.hi;
32         lo = rhs.lo;
33     }
34     bigInt &operator=(const bigInt &rhs)
35     {
36         hi = rhs.hi;
37         lo = rhs.lo;
38         return *this;
39     }
40     bigInt operator*(const bigInt &rhs)
41     {
42         bigInt ret;
43         ret.lo = this->lo * rhs.lo;
44         uint64_t a_lo = (uint32_t)(this->lo);
45         uint64_t a_hi = this->lo >> 32;
46         uint64_t b_lo = (uint32_t)(rhs.lo);

```

```

47     uint64_t b_hi = rhs.lo >> 32;
48
49     uint64_t a_x_b_hi = a_hi * b_hi;
50     uint64_t a_x_b_mid = a_hi * b_lo;
51     uint64_t b_x_a_mid = b_hi * a_lo;
52     uint64_t a_x_b_lo = a_lo * b_lo;
53
54     uint64_t carry_bit = ((uint64_t)(uint32_t)a_x_b_mid +
55                          (uint64_t)(uint32_t)b_x_a_mid +
56                          (a_x_b_lo >> 32)) >>
57                          32;
58     ret.hi = a_x_b_hi +
59             (a_x_b_mid >> 32) + (b_x_a_mid >> 32) +
60             carry_bit;
61     return ret;
62 }
63 bigInt &operator<=(int value)
64 {
65     for (int i = 0; i < value; i++)
66     {
67         int msb = this->lo >> 63;
68         this->lo <= 1;
69         this->hi <= 1;
70         this->hi |= msb;
71     }
72     return *this;
73 }
74 bigInt &operator>=(int value)
75 {
76     for (int i = 0; i < value; i++)
77     {
78         uint64_t lsb = this->hi << 63;
79         this->hi >= 1;
80         this->lo >= 1;
81         this->lo |= lsb;
82     }
83     return *this;
84 }
85 bigInt operator<<(int value)
86 {
87     bigInt ret;
88     ret.hi = this->hi;
89     ret.lo = this->lo;
90     for (int i = 0; i < value; i++)
91     {
92         int msb = ret.lo >> 63;

```

```

93         ret.lo <= 1;
94         ret.hi <= 1;
95         ret.hi |= msb;
96     }
97     return ret;
98 }
99 bigInt operator>>(int value)
100 {
101     bigInt ret;
102     ret.hi = this->hi;
103     ret.lo = this->lo;
104     for (int i = 0; i < value; i++)
105     {
106         uint64_t lsb = ret.hi << 63;
107         ret.hi >>= 1;
108         ret.lo >>= 1;
109         ret.lo |= lsb;
110     }
111     return ret;
112 }
113 bigInt operator-(const bigInt &rhs)
114 {
115     bigInt ret;
116     ret.hi = this->hi - rhs.hi;
117     if (this->lo > rhs.lo)
118     {
119         ret.lo = this->lo - rhs.lo;
120     }
121     else
122     {
123         ret.hi -= 1;
124         ret.lo = 0xffffffffffffffff - (rhs.lo - this->lo) +
125 1;
126     }
127     return ret;
128 }
129 bigInt operator/(int div)
130 {
131     bigInt ret;
132     ret.hi = this->hi / div;
133     ret.lo = this->lo / div;
134     return ret;
135 }
136 bool operator==(const bigInt &rhs)
137 {
138     if (this->hi == rhs.hi && this->lo == rhs.lo)

```

```

138         {
139             return true;
140         }
141         return false;
142     }
143     bool operator>(const bigInt &rhs)
144     {
145         if (this->hi > rhs.hi)
146         {
147             return true;
148         }
149         else if (this->hi < rhs.hi)
150         {
151             return false;
152         }
153         else
154         {
155             if (this->lo > rhs.lo)
156             {
157                 return true;
158             }
159             return false;
160         }
161     }
162     bool operator<(const bigInt &rhs)
163     {
164         if (this->hi < rhs.hi)
165         {
166             return true;
167         }
168         else if (this->hi > rhs.hi)
169         {
170             return false;
171         }
172         else
173         {
174             if (this->lo < rhs.lo)
175             {
176                 return true;
177             }
178             return false;
179         }
180     }
181     bool operator>=(const bigInt &rhs)
182     {
183         return (*this > rhs) || (*this == rhs);

```

```

184     }
185     bool operator<=(const bigInt &rhs)
186     {
187         return (*this < rhs) || (*this == rhs);
188     }
189     bool operator!=(int num)
190     {
191         bigInt a;
192         a.lo = num;
193         return !(*this == a);
194     }
195 };
196
197 bigInt modulo(bigInt A, uint64_t mod)
198 {
199     bigInt B;
200     B.lo = mod;
201     bigInt X = B;
202     while (X <= A / 2)
203     {
204         // X = X << 1;
205         X <<= 1;
206     }
207     while (A >= B)
208     {
209         if (A >= X)
210         {
211             A = A - X;
212         }
213         // X = X >> 1;
214         X >>= 1;
215     }
216     return A;
217 }
218
219 bigInt powm(bigInt base, bigInt exp, uint64_t mod)
220 {
221     bigInt ret;
222     ret.lo = 0x1;
223     bigInt temp = modulo(base, mod);
224     while (exp != 0)
225     {
226         if (exp.lo & 0x1)
227         {
228             ret = modulo((ret * temp), mod);
229         }

```

```

230         // exp = exp >> 1;
231         exp >>= 1;
232         temp = modulo((temp * temp), mod);
233     }
234     return ret;
235 }
236
237 void test()
238 {
239     vector<int> vec;
240     uint64_t mod = 0xc0001000004000b;
241     bigInt g(12332102632472395673ULL);
242     bigInt s(20337250ULL);
243     for (int i = 0; i < 512; i++)
244     {
245         s = powm(g, s, mod);
246         vec.push_back(isOdd(s.lo));
247     }
248     for (int i : vec)
249         cout << i;
250     cout << "\n0: " << count(vec.begin(), vec.end(), 0) << " "
251         << "1: " << count(vec.begin(), vec.end(), 1);
252 }
253
254 int main()
255 {
256     auto start = chrono::high_resolution_clock::now();
257     test();
258     auto end = chrono::high_resolution_clock::now();
259     std::chrono::duration<double> fp_ms = end - start;
260     cout << "\nTime: " << fp_ms.count() << endl;
261 }

```