算法分析习题选讲(第四章)

chyx111@qq.com

1259 Sum of Consecutive Primes

1259 Sum of Consecutive Primes 题目大意

给出一个正整数N，求出它有多少种方法可以表示成连续的素数的和

例如：

53 = 5 + 7 + 11 + 13 + 17

53 = 53

两种方法

$2 \leq N \leq 10000$

1259 Sum of Consecutive Primes 解题思路

先求出10000以内的所有素数

对每个输入，枚举连续的素数的起点，寻找是否有一段连续素数的和与它相等，如果有则累加答案

1259 Sum of Consecutive Primes 素数筛法

```
const int kMaxN = 11000;
bool is_prime[kMaxN];
```

```
memset(is_prime, 1, sizeof (is_prime));
is_prime[0] = is_prime[1] = false;
for (int i = 4; i < kMaxN; i += 2) is_prime[i] = false;
for (int i = 3; i < kMaxN; ++i) if (is_prime[i]) {
  for (int j = i + i; j < kMaxN; j += i) {
    is_prime[j] = false;
  }
}
```

复杂度分析

估算：

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + \frac{n}{n} = n \log n$$

实际

$O(n \log \log n)$

1259 Sum of Consecutive Primes 素数筛法 优化

```
memset(is_prime, 1, sizeof (is_prime));
is_prime[0] = is_prime[1] = false;
for (int i = 4; i < kMaxN; i += 2) is_prime[i] = false;
for (int i = 3; i * i < kMaxN; ++i) if (is_prime[i]) {
  for (int j = i * i; j < kMaxN; j += i) {
    is_prime[j] = false;
  }
}
```

枚举起点和连续序列

```
int ans = 0;
for (int i = 2; i <= n; ++i) if (is_prime[i]) {
  int sum = 0;
  for (int j = i; sum <= n; ++j) if (is_prime[j]){
    sum += j;
    if (sum == n) {
      ++ans;
      break;
    }
  }
}
```

1240 Faulty Odometer

1240 Faulty Odometer 题目大意

有个损坏的里程表，不能显示数字4，会从数字3直接跳到数字5

给出里程表的读数，求出实际里程

1240 Faulty Odometer 解题思路

里程表能显示的数字为012356789，总共9个，等价于九进制

以九进制的方式计算实际里程

1240 Faulty Odometer 进制转化

```
int ex = 1;
int ans = 0;
for (; n != 0; n /= 10) {
  int digit = n % 10;
  if (digit > 4) --digit;
  ans += ex * digit;
  ex *= 9;
}
```

1231 The Embarrassed Cryptography

1231 The Embarrassed Cryptography

题目大意

给出两个正整数K和L，问K是否存在小于L的质因数，有的话则找出最小的质因数

$4 \leq K \leq 10^{100}, 2 \leq L \leq 10^6$

1231 The Embarrassed Cryptography 解题思路

先预处理不超过10^6的所有素数

对每个不超过L的素数，检查是否能整除K。

高精度除法

1231 The Embarrassed Cryptography
素数筛法，用bitset优化

```
const int kMaxN = 1100000;
bitset<kMaxN> is_prime;
vector<int> primes;
```

```
is_prime.set();
is_prime[0] = is_prime[1] = false;
for (int i = 4; i < kMaxN; i += 2) is_prime[i] = false;
for (int i = 3; i * i < kMaxN; ++i) if (is_prime[i]) {
  for (int j = i * i; j < kMaxN; j += i) {
    is_prime[j] = false;
  }
}
for (int i = 2; i < kMaxN; ++i) if (is_prime[i]) {
  primes.push_back(i);
}
```

1231 The Embarrassed Cryptography
大整数除普通整数，压缩

```
char input[1024];
int num[1024];
int n_num;
const int kBase = 1000000000;
```

```
int n = strlen(input);
n_num = 0;
for (int i = n - 1; i >= 0; i -= 9) {
  ++n_num;
  num[n_num - 1] = 0;
  for (int j = 8; j >= 0; --j) if (i - j >= 0) {
    num[n_num - 1] = num[n_num - 1] * 10 + input[i - j] - '0';
  }
}
// input: 从高到低
// num: 从低到高!
```

```
bool can_divide(int divisor) {
  long long res = 0;
  for (int i = n_num - 1; i >= 0; --i) {
    res = (res * kBase + num[i]) % divisor;
  }
  return res == 0;
```

```
}
```

1214 信号分析

1214 信号分析 题目大意

$$a_1 = 1, a_3 = 3$$
$$a_{2n} = a_n$$
$$a_{4n+1} = 2a_{2n+1} - a_n$$
$$a_{4n+3} = 3a_{2n+1} - 2a_n$$

给出L，求$1 \leq i \leq L$中$a_L = L$的个数

1214 信号分析 解题思路

找规律

按二进制找

```
a(1) = 1
a(10) = 1
a(11) = 11
a(100) = 1
a(101) = 101
a(110) = 11
a(111) = 111
```

猜想规律：数列的第n项为n的二进制串的倒置

1214 信号分析 证明

数学归纳法

当n=1和3时，a(1)=1,a(11)=11满足

设n = a...z_2，则a(a...z0) = a(a...z) = z...a = 0z...a，满足

a(a...z01) = 2 a(a...z1) - a(a...z) = 1z...a0 - z...a = 10z...a，满足

a(a...z11) = 3 a(a...z1) - 2 a(a...z) = 1z...a0 + 1z...a - z...a0 = 11z...a，满足

1214 信号分析 问题转化 解法1

$1 \leq i \leq L$的二进制里有多少个是回文串

枚举构成回文串的其中一半数字，从而构造得到回文串，再与L比较大小

L最多为32位，因此枚举的二进制串最多为16位

1214 信号分析 问题转化 解法2

前提：L是n位

小于n位的数字一定都满足，i位数字的方案数为$2^{\lceil \frac{i}{2} \rceil - 1}$

对于n位的数字，枚举回文串的前半部分，则这一部分必须小于L的前半部分

1214 信号分析 代码

```
unsigned rev(unsigned n) {
  unsigned ret = 0;
  for (; n; n >>= 1) {
    ret <<= 1;
    if (n & 1) ret++;
  }
  return ret;
}
```

1214 信号分析 代码

```
int fun(unsigned L) {
#define TWO(x) (1 << (x))
  int len = 32 - __builtin_clz(L);
  int ans = 1;
  for (int i = 2; i <= len - 1; ++i) {
    ans += TWO((i + 1) / 2 - 1);
  }
  if (len % 2 == 0) {
    unsigned hi = L >> len / 2;
    unsigned lo = L & TWO(len / 2) - 1;
    if (hi < TWO(len / 2 - 1)) return ans;
    ans += hi - TWO(len / 2 - 1);
    hi = rev(hi);
    if (hi <= lo) ans++;
  } else {
    unsigned hi = L >> len / 2 + 1;
    unsigned lo = L & TWO(len / 2) -1;
    if (hi < TWO(len / 2 - 1)) return ans;
    ans += 2 * (hi - TWO(len / 2 - 1));
    if (L & TWO(len / 2)) ans++;
    hi = rev(hi);
    if (hi <= lo) ans++;
  }
  return ans;
}
```

1203 The Cubic End

1203 The Cubic End 题目大意

如果一个数字串，以1，3，7，9结尾

则会有一个数，它的三次方以这个数字串结尾，且长度不会超过这个数字串

现在给出这个数字串，求出这个数

字符串大小不超过10^{10}

1203 The Cubic End 解题思路

设字符串长度为n

$$x^3 \equiv y \pmod{10^n}$$

1203 The Cubic End 解题思路

从低位向高位枚举，尝试用每个数字填入这个位置，检查所得到的尾数是否相符

相符则下一个位，直至枚举完所有位置

1203 The Cubic End 高精度乘法

```
long long mul(long long a, long long b, long long mod) {
  long long c = 0;
  const int base = 2;
  for (; b != 0; b /= base) {
    c += (b % base) * a;
    c %= mod;
    a = (a * base) % mod;
  }
  return c;
}
```

1203 The Cubic End 代码

```
long long curr = 0, new_value;
for (long long mod = 10; mod <= m * 10; mod *= 10) {
  for (int i = 0; i <= 9; ++i) {
    new_value = i * (mod / 10) + curr;
    if (mul(mul(new_value, new_value, mod), new_value, mod) == m % mod) break;
  }
  curr = new_value;
}
```

1099 Packing Passengers

1099 Packing Passengers 题目大意

有两种飞机A和B

花费分别为costA和costB

容量分别为passengersA和passengersB

现在有N个乘客，每个飞机都要完全装满乘客才能起飞，问最小费用

1099 Packing Passengers 解题思路

当A，B其中一个容量为0时特判

求出两种飞机的性价比，使得性价比低的飞机尽量少

从多到少枚举性价比高的飞机的数量，如果剩余的乘客都能被另一种飞机装满，则存在解

1099 Packing Passengers 代码

```
bool Solve(long long n, long long cap1, long long cap2,
    long long *num1, long long *num2) {
  *num1 = n / cap1;
  while ((n - *num1 * cap1) % cap2 !=0) {
    if (*num1 > 0) {
      --*num1;
    } else {
      return false;
    }
  }
  *num2 = (n - *num1 * cap1) / cap2;
  return true;
}
```

```
// cost1 / cap1 <= cost2 / cap2
if (cost1 * cap2 <= cost2 * cap1) {
  valid = Solve(n, cap1, cap2, &num1, &num2);
} else {
  valid = Solve(n, cap2, cap1, &num2, &num1);
}
```

1099 Packing Passengers 解法2

扩展欧几里得算法

1014 Specialized Four-Dig

1014 Specialized Four-Dig 题目大意

求出所有在十进制，十二进制，十六进制下各位数字之和相等的四位十进制数

1014 Specialized Four-Dig 解题思路

枚举所有四位数，求出十进制，十二进制，十六进制下各位数字之和，再判断是否相等

1014 Specialized Four-Dig 代码

```
int sum_base(int n, int base) {
  int sum = 0;
  for (; n > 0; n /= base) {
    sum += n % base;
  }
  return sum;
}

bool check(int n) {
    return sum(n, 10) == sum(i, 12) && sum(i, 10) == sum(i, 16);
}
```

1119 Factstone Benchmark

1119 Factstone Benchmark 题目大意

1960年发行了4位计算机，从此以后每过10年，计算机的位数变成两倍

输入某一个年份，求出在这个年份的最大的整数n使得n!能被一个字表示

1119 Factstone Benchmark 解题思路

先求出年份对应的字长

用科学记数法表示阶乘，以2为底，从指数可以得到位数，与字长比较即可

1119 Factstone Benchmark 代码

```
int bit_len = 4 << (year - 1960) / 10;

int n = 1;
double sum = 0;
for (int i = 1; ; ++i) {
  sum += log(i);
  if (sum / log(2) > bit_len) {
    break;
  }
  n = i;
}
```

1500 Prime Gap

1500 Prime Gap 题目大意

给出一个正整数k，计算出两个相邻的素数，使得k在这两个素数之间，求出两个素数之差

如果不存在这两个素数则输出0

1500 Prime Gap 解题思路

求出小于等于k的最大素数与大于等于k的最小素数，并求出它们的差

从k分别向下和向上枚举每个整数，判断是否为素数，直到找到这两个素数

1500 Prime Gap 解题思路

```
for (int i = k; !is_prime[i]; --i) { }
for (int j = k; !is_prime[j]; ++j) { }
```

每个查询复杂度

$$O(\log n)$$