

## 【第三十八课】金融系统中的 RSA 算法（二）

### make\_rsa\_key

```
1  /*****
2   > File Name: 1.make_rsa_key.cpp
3   > Author: huguang
4   > Mail: hug@haizeix.com
5   > Created Time:
6   *****/
7
8  #include <iostream>
9  #include <cstdio>
10 #include <cstdlib>
11 #include <queue>
12 #include <stack>
13 #include <algorithm>
14 #include <string>
15 #include <map>
16 #include <set>
17 #include <vector>
18 using namespace std;
19 #define MAX_N 50000
20 long long prime[MAX_N + 5] = {0};
21 void init_prime() {
22     for (long long i = 2; i <= MAX_N; i++) {
23         if (!prime[i]) {
24             prime[++prime[0]] = i;
25         }
26         for (long long j = 1; j <= prime[0]; j++) {
27             if (i * prime[j] > MAX_N) break;
28             prime[i * prime[j]] = 1;
29             if (i % prime[j] == 0) break;
30         }
31     }
32     return ;
33 }
34
35 long long gcd(long long a, long long b) {
36     if (b) return gcd(b, a % b);
37     return a;
38 }
39
40 long long get_inv(long long a, long long b, long long &x, long long &y) {
41     if (b == 0) {
42         x = 1, y = 0;
43         return a;
```

```

44     }
45     long long r = get_inv(b, a % b, y, x);
46     y -= (a / b) * x;
47     return r;
48 }
49
50 int main() {
51     srand(time(0));
52     init_prime();
53     cout << "prime[0] = " << prime[0] << endl;
54     cout << "last prime : " << prime[prime[0]] << endl;
55     long long p, q, p_ind, q_ind;
56     do {
57         p_ind = rand() % 100 + prime[0] - 99;
58         q_ind = rand() % 100 + prime[0] - 99;
59     } while (p_ind == q_ind);
60     p = prime[p_ind];
61     q = prime[q_ind];
62     long long n = p * q, phi_n = (p - 1) * (q - 1);
63     long long e, d, y;
64     do {
65         e = rand() % phi_n;
66     } while (e == 0 || gcd(e, phi_n) != 1);
67     get_inv(e, phi_n, d, y);
68     d = ((d % phi_n) + phi_n) % phi_n;
69     assert(e * d % phi_n == 1);
70     cout << "./a.out " << e << " " << n << " " << d << " " << n << endl;
71     return 0;
72 }

```

## use\_ras\_key

```

1  /*****
2  > File Name: 2.use_ras_key.cpp
3  > Author: huguang
4  > Mail: hug@haizeix.com
5  > Created Time:
6  *****/
7
8  #include <iostream>
9  #include <cstdio>
10 #include <cstdlib>
11 #include <queue>
12 #include <stack>
13 #include <algorithm>
14 #include <string>
15 #include <map>
16 #include <set>

```

```

17 #include <vector>
18 using namespace std;
19 long long e, d, n;
20
21 long long quick_mod(long long a, long long b, long long c) {
22     long long ans = 1, temp = a;
23     while (b) {
24         if (b & 1) ans = ans * temp % c;
25         temp = temp * temp % c;
26         b >>= 1;
27     }
28     return ans;
29 }
30
31 int main(int argc, char *argv[]) {
32     sscanf(argv[1], "%lld", &e);
33     sscanf(argv[2], "%lld", &n);
34     sscanf(argv[3], "%lld", &d);
35     long long m, c;
36     while (cin >> m) {
37         c = quick_mod(m, e, n);
38         cout << m << "^" << e << " % " << n << " = " << c << endl;
39         cout << c << "^" << d << " % " << n << " = " << quick_mod(c, d, n) << endl;
40     }
41     return 0;
42 }

```

## LCP 02. 分式化简

有一个同学在学习分式。他需要将一个连分数化成最简分数，你能帮助他吗？

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots}}}$$

连分数是形如上图的分式。在本题中，所有系数都是大于等于0的整数。

输入的 `cont` 代表连分数的系数（`cont[0]` 代表上图的 `a0`，以此类推）。返回一个长度为2的数组 `[n, m]`，使得连分数的值等于 `n / m`，且 `n, m` 最大公约数为1。

示例 1:

```
1 输入: cont = [3, 2, 0, 2]
2 输出: [13, 4]
3 解释: 原连分数等价于  $3 + (1 / (2 + (1 / (0 + 1 / 2))))$ 。注意 [26, 8], [-13, -4] 都不是正确答案。
```

```
1  class Solution {
2  public:
3      vector<int> fraction(vector<int>& cont) {
4          int n = 1, m = 0;
5          for (int i = cont.size() - 1; i >= 0; i--) {
6              swap(n, m);
7              n += m * cont[i];
8              int r = gcd(n, m);
9              n /= r;
10             m /= r;
11         }
12         vector<int> ret(2);
13         ret[0] = n, ret[1] = m;
14         return ret;
15     }
16 };
```

## 1031. 两个非重叠子数组的最大和

给出非负整数数组  $A$ ，返回两个非重叠（连续）子数组中元素的最大和，子数组的长度分别为  $L$  和  $M$ 。（这里需要澄清的是，长为  $L$  的子数组可以出现在长为  $M$  的子数组之前或之后。）

从形式上看，返回最大的  $v$ ，而  $v = (A[i] + A[i+1] + \dots + A[i+L-1]) + (A[j] + A[j+1] + \dots + A[j+M-1])$  并满足下列条件之一：

- $0 \leq i < i + L - 1 < j < j + M - 1 < A.length$ ，或
- $0 \leq j < j + M - 1 < i < i + L - 1 < A.length$ 。

示例 1:

```
1 输入: A = [0,6,5,2,2,5,1,9,4], L = 1, M = 2
2 输出: 20
3 解释: 子数组的一种选择中, [9] 长度为 1, [6,5] 长度为 2。
```

```
1  class Solution {
2  public:
3      int maxSumTwoNoOverlap(vector<int>& nums, int firstLen, int secondLen) {
4          int n = nums.size();
5          int lmax[n + 1], mmax[n + 1];
6          memset(lmax, 0, sizeof(lmax));
7          memset(mmax, 0, sizeof(mmax));
8          for (int i = n - 1, lsum = 0, msum = 0; i >= 0; i--) {
9              lsum += nums[i];
```

```

10     msum += nums[i];
11     if (i + firstLen < n) lsum -= nums[i + firstLen];
12     if (i + secondLen < n) msum -= nums[i + secondLen];
13     if (i + firstLen <= n) lmax[i] = max(lmax[i + 1], lsum);
14     if (i + secondLen <= n) mmax[i] = max(mmax[i + 1], msum);
15 }
16 int ans = 0;
17 for (int i = 0, lsum = 0, msum = 0; i < n; i++) {
18     lsum += nums[i];
19     msum += nums[i];
20     if (i >= firstLen) lsum -= nums[i - firstLen];
21     if (i >= secondLen) msum -= nums[i - secondLen];
22     ans = max(ans, lsum + mmax[i + 1]);
23     ans = max(ans, msum + lmax[i + 1]);
24 }
25 return ans;
26 }
27 };

```

## 502. IPO

假设力扣 (LeetCode) 即将开始 **IPO**。为了以更高的价格将股票卖给风险投资公司，力扣 希望在 IPO 之前开展一些项目以增加其资本。由于资源有限，它只能在 IPO 之前完成最多  $k$  个不同的项目。帮助力扣 设计完成最多  $k$  个不同项目后得到最大总资本的方式。

给你  $n$  个项目。对于每个项目  $i$ ，它都有一个纯利润  $profits[i]$ ，和启动该项目需要的最小资本  $capital[i]$ 。

最初，你的资本为  $w$ 。当你完成一个项目时，你将获得纯利润，且利润将被添加到你的总资本中。

总而言之，从给定项目选择最多  $k$  个不同项目的列表，以 **最大化最终资本**，并输出最终可获得的最多资本。

答案保证在 32 位有符号整数范围内。

### 示例 1:

```

1 输入: k = 2, w = 0, profits = [1,2,3], capital = [0,1,1]
2 输出: 4
3 解释:
4 由于你的初始资本为 0，你仅可以从 0 号项目开始。
5 在完成后，你将获得 1 的利润，你的总资本将变为 1。
6 此时你可以选择开始 1 号或 2 号项目。
7 由于你最多可以选择两个项目，所以你需要完成 2 号项目以获得最大的资本。
8 因此，输出最后最大化的资本，为 0 + 1 + 3 = 4。

```

```

1 class Solution {
2 public:
3     int findMaximizedCapital(int k, int w, vector<int>& profits, vector<int>& capital) {
4         int n = profits.size();

```



```

5     vector<int> ind(n);
6     for (int i = 0; i < n; i++) ind[i] = i;
7     sort(ind.begin(), ind.end(), [](int i, int j) -> bool { return capital[i]
< capital[j]; });
8     priority_queue<int> q;
9     int i = 0;
10    while (k--) {
11        while (i < n && capital[ind[i]] <= w) q.push(profits[ind[i]]), i += 1;
12        if (q.empty()) break;
13        w += q.top();
14        q.pop();
15    }
16    return w;
17 }
18 };

```

## 2029. 石子游戏 IX

Alice 和 Bob 再次设计了一款新的石子游戏。现有一行  $n$  个石子，每个石子都有一个关联的数字表示它的价值。给你一个整数数组 `stones`，其中 `stones[i]` 是第  $i$  个石子的价值。

Alice 和 Bob 轮流进行自己的回合，**Alice** 先手。每一回合，玩家需要从 `stones` 中移除任一石子。

- 如果玩家移除石子后，导致 **所有已移除石子** 的价值 **总和** 可以被 3 整除，那么该玩家就 **输掉游戏**。
- 如果不满足上一条，且移除后没有任何剩余的石子，那么 Bob 将会直接获胜（即便是在 Alice 的回合）。

假设两位玩家均采用 **最佳** 决策。如果 Alice 获胜，返回 `true`；如果 Bob 获胜，返回 `false`。

**示例 1:**

```

1  输入: stones = [2,1]
2  输出: true
3  解释: 游戏进行如下:
4  - 回合 1: Alice 可以移除任意一个石子。
5  - 回合 2: Bob 移除剩下的石子。
6  已移除的石子的值总和为  $1 + 2 = 3$  且可以被 3 整除。因此, Bob 输, Alice 获胜。

```

```

1  class Solution {
2  public:
3      bool stoneGameIX(vector<int>& stones) {
4          long long cnt[3] = {0};
5          for (auto x : stones) cnt[x % 3] += 1;
6          return (cnt[0] % 2 == 0 && cnt[1] * cnt[2] >= 1) || (cnt[0] % 2 == 1 &&
abs(cnt[1] - cnt[2]) >= 3);
7      }
8  };

```

## 284. 窥探迭代器

请你在设计一个迭代器，在集成现有迭代器拥有的 `hasNext` 和 `next` 操作的基础上，还额外支持 `peek` 操作。

实现 `PeekingIterator` 类：

- `PeekingIterator(Iterator<int> nums)` 使用指定整数迭代器 `nums` 初始化迭代器。
- `int next()` 返回数组中的下一个元素，并将指针移动到下个元素处。
- `bool hasNext()` 如果数组中存在下一个元素，返回 `true`；否则，返回 `false`。
- `int peek()` 返回数组中的下一个元素，但不移动指针。

注意：每种语言可能有不同的构造函数和迭代器，但均支持 `int next()` 和 `boolean hasNext()` 函数。

示例：

```
1  输入：
2  ["PeekingIterator", "next", "peek", "next", "next", "hasNext"]
3  [[[1, 2, 3]], [], [], [], [], []]
4  输出：
5  [null, 1, 2, 2, 3, false]
6
7  解释：
8  PeekingIterator peekingIterator = new PeekingIterator([1, 2, 3]); // [1,2,3]
9  peekingIterator.next();      // 返回 1，指针移动到下一个元素 [1,2,3]
10 peekingIterator.peek();      // 返回 2，指针未发生移动 [1,2,3]
11 peekingIterator.next();      // 返回 2，指针移动到下一个元素 [1,2,3]
12 peekingIterator.next();      // 返回 3，指针移动到下一个元素 [1,2,3]
13 peekingIterator.hasNext();   // 返回 False
```

```
1  /*
2   * Below is the interface for Iterator, which is already defined for you.
3   * **DO NOT** modify the interface for Iterator.
4   *
5   * class Iterator {
6   *     struct Data;
7   *     Data* data;
8   *     public:
9   *         Iterator(const vector<int>& nums);
10   *         Iterator(const Iterator& iter);
11   *
12   *         // Returns the next element in the iteration.
13   *         int next();
14   *
15   *         // Returns true if the iteration has more elements.
16   *         bool hasNext() const;
17   * };
18   */
19
20 class PeekingIterator : public Iterator {
```

```

21 public:
22     int cur;
23     int end_flag;
24     PeekingIterator(const vector<int>& nums) : Iterator(nums) {
25         // Initialize any member here.
26         // **DO NOT** save a copy of nums and manipulate it directly.
27         // You should only use the Iterator interface methods.
28         end_flag = false;
29         if (Iterator::hasNext()) {
30             cur = Iterator::next();
31         } else {
32             end_flag = true;
33         }
34     }
35
36     // Returns the next element in the iteration without advancing the iterator.
37     int peek() {
38         return cur;
39     }
40
41     // hasNext() and next() should behave the same as in the Iterator interface.
42     // Override them if needed.
43     int next() {
44         if (end_flag) {
45             return -1;
46         }
47         int ret = cur;
48         if (Iterator::hasNext()) {
49             cur = Iterator::next();
50         } else {
51             end_flag = true;
52         }
53         return ret;
54     }
55
56     bool hasNext() const {
57         return !end_flag;
58     }
59 };

```

## 1723. 完成所有工作的最短时间

给你一个整数数组 `jobs`，其中 `jobs[i]` 是完成第 `i` 项工作要花费的时间。

请你将这些工作分配给 `k` 位工人。所有工作都应该分配给工人，且每项工作只能分配给一位工人。工人的 **工作时间** 是完成分配给他们的所有工作花费时间的总和。请你设计一套最佳的工作分配方案，使工人的 **最大工作时间** 得以 **最小化**。

返回分配方案中尽可能 **最小** 的 **最大工作时间**。



### 示例 1:

```
1 输入: jobs = [3,2,3], k = 3
2 输出: 3
3 解释: 给每位工人分配一项工作, 最大工作时间是 3 。
```

```
1 class Solution {
2 public:
3     void dfs(vector<int> &jobs, int ind, vector<int> &slot, int max_num, int &ans)
4     {
5         if (ind == jobs.size()) {
6             ans = max_num;
7             return ;
8         }
9         int k = slot.size();
10        for (int i = 0; i < k; i++) {
11            if (slot[i] + jobs[ind] > ans) continue;
12            slot[i] += jobs[ind];
13            dfs(jobs, ind + 1, slot, max(max_num, slot[i]), ans);
14            slot[i] -= jobs[ind];
15            if (slot[i] == 0) break;
16        }
17        int minimumTimeRequired(vector<int>& jobs, int k) {
18            vector<int> slot(k, 0);
19            int ans = INT_MAX;
20            dfs(jobs, 0, slot, 0, ans);
21            return ans;
22        }
23    };
24 }
```

## 1477. 找两个和为目标值且不重叠的子数组

给你一个整数数组 `arr` 和一个整数值 `target` 。

请在 `arr` 中找 **两个互不重叠的子数组** 且它们的和都等于 `target` 。可能会有多种方案, 请你返回满足要求的两个子数组长度和的 **最小值** 。

请返回满足要求的最小长度和, 如果无法找到这样的两个子数组, 请返回 **-1** 。

### 示例 1:

```
1 输入: arr = [3,2,2,4,3], target = 3
2 输出: 2
3 解释: 只有两个子数组和为 3 ([3] 和 [3])。它们的长度和为 2 。
```

```
1 class Solution {
2 public:
```

```

3     typedef pair<int, int> PII;
4     int minSumOfLengths(vector<int>& arr, int target) {
5         int n = arr.size(), l = 0, sum = 0;
6         vector<PII> ranges;
7         for (int r = 0; r < n; r++) {
8             sum += arr[r];
9             while (sum > target && l <= r) {
10                 sum -= arr[l];
11                 l += 1;
12             }
13             if (sum == target) ranges.push_back(PII(l, r));
14         }
15         int ans = -1, pre_ind = -1, pre_min = n + 1;
16         for (auto x : ranges) {
17             while (ranges[pre_ind + 1].second < x.first) {
18                 pre_ind += 1;
19                 pre_min = min(pre_min, ranges[pre_ind].second -
20 ranges[pre_ind].first + 1);
21             }
22             if (pre_ind == -1) continue;
23             if (ans == -1 || ans > pre_min + x.second - x.first + 1) {
24                 ans = pre_min + x.second - x.first + 1;
25             }
26         }
27         return ans;
28     };

```

## 面试题 17.13. 恢复空格

哦，不！你不小心把一个长篇文章中的空格、标点都删掉了，并且大写也弄成了小写。像句子 "I reset the computer. It still didn't boot!" 已经变成了 "iresetthecomputeritstill didntboot"。在处理标点符号和大小写之前，你得先把它断成词语。当然了，你有一本厚厚的词典 dictionary，不过，有些词没在词典里。假设文章用 sentence 表示，设计一个算法，把文章断开，要求未识别的字符最少，返回未识别的字符数。

**注意：**本题相对原题稍作改动，只需返回未识别的字符数

**示例：**

```

1  输入：
2  dictionary = ["looked","just","like","her","brother"]
3  sentence = "jesslookedjustliketimherbrother"
4  输出： 7
5  解释： 断句后为"jess looked just liketim her brother"，共7个未识别字符。

```

```

1  struct Node {
2      Node() {
3          flag = false;

```

```

4     memset(next, 0, sizeof(next));
5 }
6 bool flag;
7 Node *next[26];
8 };
9
10 struct Trie {
11     void insert(string &s) {
12         Node *p = &root;
13         for (auto x : s) {
14             int ind = x - 'a';
15             if (p->next[ind] == nullptr) p->next[ind] = new Node();
16             p = p->next[ind];
17         }
18         p->flag = true;
19     }
20     void get_mark(string &s, int pos, vector<vector<int>> &mark) {
21         Node *p = &root;
22         for (int i = pos; s[i]; i++) {
23             int ind = s[i] - 'a';
24             p = p->next[ind];
25             if (p == nullptr) break;
26             if (p->flag) mark[i + 1].push_back(pos);
27         }
28         return ;
29     }
30     Node root;
31 };
32
33 class Solution {
34 public:
35     int respace(vector<string>& dictionary, string sentence) {
36         Trie tree;
37         for (auto s : dictionary) tree.insert(s);
38         int n = sentence.size();
39         vector<int> dp(n + 1);
40         vector<vector<int>> mark(n + 1);
41         for (int i = 0; i < n; i++) {
42             tree.get_mark(sentence, i, mark);
43         }
44         dp[0] = 0;
45         for (int i = 1; i <= n; i++) {
46             dp[i] = dp[i - 1] + 1;
47             for (auto j : mark[i]) {
48                 dp[i] = min(dp[i], dp[j]);
49             }
50         }
51         return dp[n];
52     }

```

