天津大学

程序设计综合实践课程报告

字符串和数学实验

学生姓名 陈昊昆

学院名称 智算学部

专　　业 软件工程

学　　号 **3021001196**

1. 字符串

1.1 题目分析

使用两个嵌套的循环来遍历字符串 a 中所有可能的子串，并检查它们是否与字符串 b 相等

如果找到了一个匹配的子串，则返回匹配的位置

如果没有找到匹配的子串，则返回-1

1.2 题目代码（带注释）

```
#include<bits/stdc++.h>
using namespace std;

int main() {
    int t;
    cin >> t;
    while (t--) {
        string a, b;
        cin >> a >> b;
        int index = -1;
        // 遍历字符串 a 中所有可能的子串
        for (int i = 0; i <= a.size() - b.size(); i++) {
            bool match = true;
            //  检查以字符 a[i]为起始点的长度为字符串 b 长度的子串是否与字符串 b 相等
            for (int j = 0; j < b.size(); j++) {
                if (a[i + j] != b[j]) {
                    match = false; // 如果不相等，则将 match 标记为 false
                    break;
                }
            }
            if (match) { // 如果匹配成功，则返回匹配的位置
                index = i;
                break;
            }
        }
        cout << index << endl;
```

```
    }
    return 0;
}
```

2. Oulipo

2.1 题目分析

比第一题增加一个用于统计的变量

每发现一个匹配的子串，统计值加一

最后输出统计值

2.2 题目代码（带注释）

```
#include<bits/stdc++.h>
using namespace std;

// 实现字符串匹配算法
int match(string w, string t) {
    int n = w.size(), m = t.size();
    int cnt = 0;
    for (int i = 0; i <= m - n; i++) {
        // 查询是否为子串
        if (t.substr(i, n) == w) {
            cnt++;
        }
    }
    return cnt;
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        string w, t;
        cin >> w >> t;
        int cnt = match(w, t);
        cout << cnt << endl;
    }
    return 0;
}
```

3. 本质不同的子串

3.1 题目分析

利用 hash_str 函数，用于计算字符串的哈希值

利用 count_substrings 函数，用于计算本质不同的子串数目

使用一个哈希集合来记录所有出现过的子串的哈希值，最终返回集合大小即本质不同的子串数目

为了处理字符串环，对于每个字符串，将其末尾部分和前缀部分拼接起来，然后枚举所有子串，将其哈希值加入集合中

3.2 题目代码（带注释）

```
#include <iostream>
#include <string>
#include <unordered_set>
using namespace std;

const int N = 510, P = 131;

// 计算字符串哈希值
unsigned long long hash_str(string s) {
    int n = s.size();
    unsigned long long h = 0;
    for (int i = 0; i < n; i++) {
        h = h * P + s[i];
    }
    return h;
}

// 计算本质不同的子串数目
int count_substrings(string s) {
    int n = s.size();
    unordered_set<unsigned long long> hash_set;
    for (int i = 0; i < n; i++) {
        // 构造字符串环
        string t = s.substr(i) + s.substr(0, i);
```

```
        for (int j = 0; j < n; j++) {
            hash_set.insert(hash_str(t.substr(j)));
        }
    }
    return hash_set.size();
}

int main() {
    int n;
    cin >> n;
    while (n--) {
        string s;
        cin >> s;
        cout << count_substrings(s) << endl;
    }
    return 0;
}
```

4. 最小公倍数

4.1 题目分析

使用辗转相除法实现求两个数的最大公约数，递归地调用自身，直到找到两个数的公约数为止

使用公式 a*b = gcd(a,b)*lcm(a,b)求两个数的最小公倍数

4.2 题目代码（带注释）

```
#include <iostream>
using namespace std;

// 求两个数的最大公约数
int gcd(int a, int b) {
    return b == 0 ? a : gcd(b, a % b);
}

// 求两个数的最小公倍数
int lcm(int a, int b) {
    return a * b / gcd(a, b);
}

int main() {
    int a, b;
    while (cin >> a >> b) {
        cout << lcm(a, b) << endl;
    }
    return 0;
}
```

5. 素数求和

5.1 题目分析

利用埃氏筛法求得小于等于 n 的素数

然后将其求和

需要注意的是，由于 n 较大，需用 long 类型来存取

5.2 题目代码（带注释）

```
#include<bits/stdc++.h>
using namespace std;


// 埃氏筛法求素数
long* getPrimes(long n) {
    long *prime = new long [n+5];
    bool *isPrime = new bool [n+5];
    for (long i = 2; i <= n; i++) {
        prime[i] = i;
        isPrime[i] = true;
    }

    // 所有其倍数都是非素数
    for (long i = 2; i <= n; i++) {
        if (isPrime[i]) {
            for (long j = i * i; j <= n; j += i) {
                isPrime[j] = false;
                prime[j] = 0;
            }
        }
    }
    return prime;
}

int main() {
    long n;
    cin >> n;
```

```
    long* primes = getPrimes(n);
    unsigned long long sum = 0;
    for (long i = 2; i <= n; i++) {
        sum += primes[i];
    }
    cout << sum << endl;
    return 0;
}
```

6. 人见人爱 A^B

6.1 题目分析

利用快速幂的思想，求出后三位数

将阶数看作一个二进制数，然后对阶数进行遍历，当该位为 1 时，更新 ans

同时更新 a，进行下一位的计算

6.2 题目代码（带注释）

```
#include<bits/stdc++.h>
using namespace std;


int main() {
    int a, b;
    while (cin >> a >> b && (a || b)) {
        int ans = 1;
        // 快速幂的思想
        while (b) {
            // 当 b 该位是 1，则乘上一个 a
            if (b & 1) {
                ans = ans * a % 1000;
            }
            a = a * a % 1000;
            b >>= 1;
        }
        cout << ans << endl;
    }
    return 0;
}
```

7. XORinacci

7.1 题目分析

与求斐波那契数列思路相同

将加法+替换为异或运算^

7.2 题目代码（带注释）

```
#include <iostream>
using namespace std;

int main() {
    int t;
    cin >> t;
    while (t--) {
        int a, b, n;
        cin >> a >> b >> n;
        if (n == 0) {
            cout << a << endl;
        } else if (n == 1) {
            cout << b << endl;
        } else {
            int f0 = a, f1 = b, fn;
            for (int i = 2; i <= n; i++) {
                fn = f0 ^ f1; // 计算 异或值
                f0 = f1;
                f1 = fn;
            }
            cout << fn << endl;
        }
    }
    return 0;
}
```

8. 不同的 n/i

8.1 题目分析

由于数据量大且数据本身非常大

故用数学方法得出通项公式后直接计算出答案

2 *sqrt(n) - (!(n / sqrt(n) - sqrt(n)))

8.2 题目代码（带注释）

```
#include <iostream>
#include <cmath>
#include <cstring>

using namespace std;

int main()
{
    long long T, n;
    cin >> T;
    while (T--)
    {
        cin >> n;
        //直接利用公式输出答案即可
        cout << 2 * long long(sqrt(n)) - (!(n / long long(sqrt(n)) - 
long long(sqrt(n)))) << endl;
    }
}
```