什么是高精度算法

在计算机的 `C++` 语言中，基本数据类型（`int`，`long long`），他们的表达范围是有限的。`int` 只能表示到 $-2^{31} \sim 2^{31}-1$，`long long` 只能表示 $-2^{63} \sim 2^{63}-1$。

当数字大小高于我们常用的整型变量，且我们依旧要让它们做整型运算（`+`，`-`，`*`，`/`），这种运算我们就需要手动模拟进行运算，即为高精度算法。

对于该知识点，如果只是为了做题，`Java` 与 `Python` 无需掌握。`Java` 有自带的 `BigInteger,BigDecimal`，可以进行高精度运算。`Python` 在默认情况下数字是无限大的。

高精度加法

原理为模拟小学列竖式。

```
  123
+ 4567
```

1. 对于两个数字，首先从最低位到最高位依次相加。
2. 两个长度为 N, M 的数字相加。得到的数字长度结果一定是不超过 $\max(N, M)+1$。
3. 因此从最低位开始到 $\max(N, M)+1$ 进行模拟，依次处理进位即可。
4. 这里需要用字符串存储数字。

- C++

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

// 定义常量 N 为 510，表示最多处理的数字位数（考虑进位）
const int N = 510;
// 定义两个字符数组 s1 和 s2 来存储输入的两个大数
char s1[510], s2[510];
// 定义一个整数数组 res 存储计算结果
int res[N];
// n 和 m 分别表示 s1 和 s2 的长度
int n, m;
// len 用来记录结果数组 res 最后一个非零数字的位置
int len = 0;

// add 函数：将两个大数相加，存储结果在 res 数组中
void add(char s1[], char s2[], int n, int m) {
    // 把 s1 数组反转存入 res 数组（从低位到高位存储）
    for (int i = n - 1, j = 0; i >= 0; i--, j++) {
        res[j] = s1[i] - '0'; // 将字符数字转为整数并存入 res 数组
    }

    // 把 s2 数组反转存入 res 数组，执行加法
    for (int i = m - 1, j = 0; i >= 0; i--, j++) {
        res[j] = res[j] + s2[i] - '0'; // 对应位置加上 s2 数字
    }

    // 处理可能的进位，遍历 res 数组处理进位
    for (int j = 0; j <= 505; j++) {
        res[j + 1] = res[j + 1] + res[j] / 10; // 当前位的进位加到高一位
```

```
        res[j] %= 10; // 当前位只保留个位数
    }

    // 寻找最终结果的最高有效位，记录其位置
    for (int j = 505; j >= 0; j--) {
        if (res[j]) { // 找到第一个非零数字
            len = j; // 记录最后一个非零位的位置
            break;
        }
    }
}

int main() {
    // 输入两个大数 s1 和 s2
    scanf("%s %s", s1, s2);

    // 计算 s1 和 s2 的长度
    n = strlen(s1);
    m = strlen(s2);

    // 调用 add 函数进行大数相加
    add(s1, s2, n, m);

    // 从最高位开始输出结果
    for (int i = len; i >= 0; i--) {
        cout << res[i]; // 输出每一位
    }

    return 0;
}

```