学习小结：

1. 位运算

&（全1则1）

|（全0则0）

^（不同则1）

<<左移

>>右移

用一题解释上述内容：

求a+b的值（低精度）

#include<stdio.h>

int main() {

int a, b;

int sum1 = 0;

int sum2 = 0;

int sum = 0;

scanf("%d%d", &a, &b);

sum1 = a ^ b; //无进位加法

sum2 = (a & b) << 1; //有进位加法

sum = sum1 + sum2;

printf("%d", sum);

return 0;

}

基本思路：一个数进行加法，可以分为无进位与有进位。

1. 高精度算法

对上诉方法无法解决的较大的数的加法，可以选用高精度算法

int n = strlen(numN);

int m = strlen(numM);

for (i = 1, j = n; i <= n; i++, j--) {

a[i] = numN[j]-'0';

}

for (i = 1, j = m; i <= m; i++, j--) {

b[i] = numM[j]-'0';

}

（此步用于拆开字符串，并把单个数字放进数组）

for (i = 1; i < m + n; i++) {

c[i] = a[i] + b[i];

if (c[i] >= 10) {

c[i + 1] += c[i] / 10;

c[i] %= 10;

}

}

（此步进行加法及进位计算）

for (i = m + n; i>=1&&c[i] == 0; i--);

for (j = 1; j <= i; j++) {

printf("%d", c[j]);

}

（此步进行输出）

1. 贪心算法

简单解释为局部最优选择

比如选奖品，有100份选10份，使选的为最重的，则每一次都选剩余的最重的

（在上述问题中，每一次的最优解则对应最后的最优解，但是有很多情况并不是如此，因此贪心算法有一定的局限）