PTA 循环结构 E

rogeryoungh

2021 年 05 月 08 日

目录

1 约定 **2**

2 时间复杂度 **2**

3 简单循环 **3**

3.1 7-1 输出等腰杨辉三角 . . . 3
3.2 7-2 含 8 的数字的个数 . . . 4
3.3 7-3 立方和 . . . 4
3.4 ！ 7-4 统计单词数量 . . . 5
3.5 7-6 中国余数定理 . . . 6
3.6 7-7 多项式求值 . . . 6
3.7 7-9 英文字母替换加密 . . . 7
3.8 7-10 字母塔 . . . 7
3.9 7-12 质因子分解 . . . 7
3.10 7-13 启程几何 . . . 8
3.11 7-14 进度条 . . . 9
3.12 7-17 计算到任意日期的总天数 . . . 9
3.13 7-18 最大值和最小值 . . . 10
3.14 7-19 计算评分 . . . 10
3.15 7-20 我们爱运动 . . . 10
3.16 7-21 累加 a-aa+aaa-aaaa+... . . . 11
3.17 7-23 循环 - n 个数最大值 . . . 11
3.18 7-25 求简单交错序列的前 N 项和 . . . 12
3.19 7-26 谁最高 . . . 12

4 格式输出 **13**

4.1 7-4 输出 2 到 n 之间的全部素数 . . . 13
4.2 7-5 输出前 n 个 Fibonacci 数 . . . 14
4.3 7-11 数字菱形 . . . 15
4.4 7-15 重排矩阵 . . . 16
4.5 7-16 1000 以内所有各位数字之和为 n 的正整数 . . . 18
4.6 7-22 嵌套循环-素数的和 . . . 19

PTA 循环结构 EASY 部分，PDF。

1 约定

随着程序逐渐复杂，我决定介绍一些算法竞赛中常用的定义，来简化我们的程序。

```
typedef long long ll;
#define _fora(i,a,n) for(int i=(a);i<=(n);i++)
#define _forz(i,a,n) for(int i=(a);i>=(n);i--)
```

即 `ll` 是 `long long` 类型的简写，能够带来比 `int` 更大的范围。
同样，`_fora` 和 `_forz` 是 `for` 的简写。
可能会在讲解中使用它们，感兴趣的可以尝试。

2 时间复杂度

可能已经有同学察觉到，尽管很多写法都可以通过测试，可它们并不是一样快的。
比如，求

$$\sum_{i=1}^{n}\sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{i(i+1)}{2} = \frac{1}{6}n(n+1)(n+2)$$

我们可以写出四种写法

```
int sum0(int n) { // 不会真有人这么写吧
    int sum = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= i; j++)
            for (int k = 1; k <= j; k++)
                sum++;
    return sum;
}

int sum1(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= i; j++)
            sum += j;
    return sum;
}

int sum2(int n) {
    int sum = 0;
```

```
    for (int i = 1; i <= n; i++)
        sum += i * (i + 1) / 2;
    return sum;
}

int sum3(int n) {
    return n * (n + 1) * (n + 2) / 6;
}
```

可以得出，`sum0` 累加执行的次数是 $\frac{1}{6}(n^3+3n^2+2n)$，`sum1` 累加执行的次数是 $\frac{1}{2}(n^2+n)$，而 `sum2` 需要 n 次运算，`sum3` 只需要 1 次。

尽管电脑的运行速度很快，可并不是无限快，不同的算法所需时间差别可能会很大。

当 n 较大时，比如 $n=10000$，`sum0` 需要接近 10^{12} 次，`sum1` 大致 10^8 次，`sum2` 大致 10^4，而 `sum3` 还是 1 次，差距非常明显。

为了凸显算法的运行时间和 n 的关系，在表示算法的时间复杂度时常常略去常数和低阶项，系数也会被忽略。比如

$$\frac{1}{6}(n^3+3n^2+2n) \sim \frac{1}{6}n^3 \sim n^3$$

我们可将上述四种算法的时间复杂度记为 $O(n^3)$，$O(n^2)$，$O(n)$ 和 $O(1)$。时间复杂度不一定是全是多项式，还可能是 $O(2^n)$，$O(\log n)$，$O(n\log n)$ 等等。

在着手编写之前先估算算法的复杂度，一些显然过不去的算法就没必要写了，一般把计算机的执行速度定为 10^9 次，我们可以列出算法的最大规模

运算量	n	$n!$	2^n	n^2	n^3	$n\log_2 n$
数据范围	10^9	13	29	31622	1000	3.9×10^7
二倍速度	2×10^9	13	30	44721	1259	7.6×10^7

时间复杂度本身是很复杂的概念，有兴趣的可以查阅资料。因为时间很容易测不准，不能简单的以运行时间评判程序的优劣，尤其运行时间极其短时；同样也不能以上界分析的结果简单的断定程序的速度。在不少情况下，算法实际能解决的问题规模与上表有着较大差异。

尽管如此，此表还是有一定借鉴意义的，比如一个指明 $n\leqslant 20$ 的题目可能 2^n 的算法已经足够，而 $n\leqslant 10^4$ 的题目可能需要 n^2 的算法，$n\leqslant 10^6$ 则可能至少要 $n\log n$ 的算法了。

相对于充斥着各种奇妙优化的算法，朴素的算法常称为暴力。

我们应尽量思考，尝试发现更优的算法，以更佳的方法解决问题。

3 简单循环

大部分题还是常规的。

3.1 7-1 输出等腰杨辉三角

样例有毒，仔细读题。应该没有人会真的写循环吧（

连续的字符常量会自动合并。

```c
int main() {
    printf(
        "              1\n"
        "            1   1\n"
        "          1   2   1\n"
        "        1   3   3   1\n"
        "      1   4   6   4   1\n"
        "    1   5  10  10   5   1\n"
    );
    return 0;
}
```

3.2 7-2 含 8 的数字的个数

暴力即可。

```c
int main() {
    int a, b;
    scanf("%d%d", &a, &b);
    int sum = 0;
    for (int i = a; i <= b; i++) {
        int ti = i;
        while (ti > 0) {
            if (ti % 10 == 8) {
                sum++;
                break;
            }
            ti /= 10;
        }
    }
    printf("%d", sum);
    return 0;
}
```

假如要求的数字范围更大，设 a, b 为 `int` 范围的数据。

不妨令 `a = 1, b = 123456987` 试试看你的算法要跑多久？怎么优化？详细见 PTA 循环结构 H。

3.3 7-3 立方和

把自身不断立方求和看作数字之间的链，那么关键在于判环。一种办法是记下所有的数，每增加一个数就和之前所有的数比对。有 $O(n^2)$ 和 $O(n)$ 两种写法。

但还有一种思考方式。注意到一条链的长度不可能超过 1000，而且要判的环是自身成环，于是暴力 1000 次即可。

整数的幂一般不要用 `pow`，因为其结果是 `double`，可能会有精度问题。

```c
int sum(int x) {
    int a = x % 10;
    x /= 10;
    int b = x % 10;
    x /= 10;
    int c = x % 10;
    return a * a * a + b * b * b + c * c * c;
}

int main() {
    int x;
    scanf("%d", &x);
    for (int i = 0; i < 1000; i++) {
        x = sum(x);
    }
    if (sum(x) == x)
        printf("%d\n", x);
    else
        printf("error");
    return 0;
}
```

3.4 ！7-4 统计单词数量

怎么题目没了?

没给数据，猜一手 1000086。
大数组不要开在函数里，尽量做为全局变量，否则数组过大可能会爆栈，会段溢出。
或者使用 getchar 做到在线运行，那样给多长都不怕了。

```c
int isAlpha(int x) {
    if ('A' <= x && x <= 'Z')
        return 1;
    else if ('a' <= x && x <= 'z')
        return 1;
    return 0;
}

char s[1000086];

int main() {
    gets(s);
    int len = strlen(s);
```

```
    int p = isAlpha(s[0]);
    int sum = 0;
    for (int i = 1; i <= len; i++) {
        int t = isAlpha(s[i]);
        if (p && !t)
            sum++;
        p = t;
    }
    printf("%d", sum);
    return 0;
}
```

3.5 7-6 中国余数定理

显然答案只会在 $1 \sim 105$ 之间，暴力枚举即可，代码就不放了。

中国剩余定理解法

```
int main() {
    int a, b, c;
    while (scanf("%d%d%d", &c, &a, &b) != EOF) {
        int ans = a * 70 + b * 21 + c * 15;
        ans = (ans - 1 + 105) % 105 + 1;
        printf("%d\n", ans);
    }
    return 0;
}
```

解释见 PTA 循环结构 H

3.6 7-7 多项式求值

显然

$$\sum_{k=1}^{n}(-1)^{k+1}k^2 = (-1)^{n+1}\frac{n(n+1)}{2}$$

```
int main() {
    int n;
    scanf("%d", &n);
    if (n % 2 == 0)
        printf("-");
    printf("%d", n * (n + 1) / 2);
    return 0;
}
```

3.7 7-9 英文字母替换加密

```
int jiami(int x) {
    if ('a' <= x && x <= 'z') {
        x = (x - 'a' + 1) % 26 + 'A';
    } else if ('A' <= x && x <= 'Z') {
        x = (x - 'A' + 1) % 26 + 'a';
    }
    return x;
}

char s[100086];

int main() {
    gets(s);
    int len = strlen(s);
    for (int i = 0; i < len; i++) {
        s[i] = jiami(s[i]);
    }
    printf(s);
    return 0;
}
```

3.8 7-10 字母塔

```
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n - i; j++)
            printf(" ");
        for (int j = 1; j <= i; j++)
            printf("%c", j + 'A' - 1);
        for (int j = i - 1; j >= 1; j--)
            printf("%c", j + 'A' - 1);
        printf("\n");
    }
    return 0;
}
```

3.9 7-12 质因子分解

试除即可。

```c
int rst[1000], cnt;

int main() {
    int n;
    scanf("%d", &n);
    if (n == 2) {
        printf("2=2");
        return 0;
    }
    printf("%d=", n);
    while (n % 2 == 0) {
        rst[++cnt] = 2;
        n /= 2;
    }
    int sn = (int) sqrt(n * 1.0);
    for (int i = 3; i <= sn; i += 2) {
        while (n % i == 0) {
            rst[++cnt] = i;
            n /= i;
        }
    }
    if (n > 1)
        rst[++cnt] = n;
    printf("%d", rst[1]);
    for (int i = 2; i <= cnt; i++) {
        printf("*%d", rst[i]);
    }
    return 0;
}
```

3.10 7-13 启程几何

```c
int main() {
    int sa, sb;
    scanf("%d %d", &sa, &sb);
    int a = 5, b = 30, day = 1;
    int sum = -sb;
    while (sum < sa) {
        day++;
        a *= 2;
        b += 30;
        sum += a + b - sb;
```

```
    }
    printf("%d", day + 1);
    return 0;
}
```

3.11 7-14 进度条

```
int main() {
    int seed;
    scanf("%d", &seed);
    srand(seed);
    for (int i = 1; i <= seed; i++) {
        int t = 1 + rand() % 35;
        for (int i = 1; i <= t; i++)
            printf(">");
        for (int i = t + 1; i <= 35; i++)
            printf(".");
        double d = t * 100 / 35.0;
        printf(" %2.0lf%%\n", d);
    }
    return 0;
}
```

3.12 7-17 计算到任意日期的总天数

```
int a[13]={0,31,59,90,120,151,181,212,243,273,304,334,365};

int main() {
    int y,m,d;
    scanf("%d%d%d", &y, &m, &d);
    int run = y / 4 - y/100 + y/400;

    int sum = run + 365 * (y-1);
    if(y % 400 == 0 || (y%4==0 && y%100!=0))
        sum--;
    printf("%d\n", sum);

    sum += a[m-1];
    if(y % 400 == 0 || (y%4==0 && y%100!=0))
        sum++;
    printf("%d\n", sum);

```

```
    sum += d;
    printf("%d\n", sum);
    return 0;
}
```

3.13 7-18 最大值和最小值

```
int main() {
    int n;
    scanf("%d", &n);
    int max = 0, min = 0x3f3f3f3f;
    for (int i = 1; i <= n; i++) {
        int t;
        scanf("%d", &t);
        max = max < t ? t : max;
        min = min > t ? t : min;
    }
    printf("%d", max - min);
    return 0;
}
```

3.14 7-19 计算评分

```
int main() {
    int n;
    scanf("%d", &n);
    double sum = 0;
    double min = 101, max = 0;
    for (int i = 1; i <= n; i++) {
        double t;
        scanf("%lf", &t);
        max = max < t ? t : max;
        min = min > t ? t : min;
        sum += t;
    }
    double score = (sum - min - max) / (n - 2);
    printf("%.0lf %.1lf %.1lf", score, min, max);
    return 0;
}
```

3.15 7-20 我们爱运动

```c
double nn[30005];

int main() {
    int n, a, b;
    scanf("%d %d %d", &n, &a, &b);
    for (int i = 1; i <= n; i++) {
        scanf("%lf", &nn[i]);
    }
    int sa = 1, sb = 1;
    for (int i = 1; i <= n; i++) {
        if (nn[a] > nn[i])
            sa++;
        if (nn[b] > nn[i])
            sb++;
    }
    printf("%d %d", sa, sb);
    return 0;
}
```

3.16 7-21 累加 a-aa+aaa-aaaa+...

不难找规律。

```c
int main() {
    char a;
    int n;
    scanf("%c %d", &a, &n);
    printf("sum=");
    if(n % 2 == 0)
        printf("-");
    for (int i = 1; i <= n; i++) {
        if(i % 2 == 1)
            putchar(a);
        else
            putchar('0');
    }
    return 0;
}
```

3.17 7-23 循环 - n 个数最大值

```c
int main() {
```

```
    int n;
    scanf("%d", &n);
    if(n <= 0)
        return 0;
    int max = 0;
    for (int i = 1; i <= n; i++) {
        int t;
        scanf("%d", &t);
        max = max < t ? t : max;
    }
    printf("%d", max);
    return 0;
}
```

3.18 7-25 求简单交错序列的前 N 项和

```
int main() {
    int n;
    scanf("%d", &n);
    double sum = 0;
    double flag = 1;
    for (int i = 1; i <= n; i++) {
        sum += flag * 1 / (2 * i - 1.0);
        flag = -flag;
    }
    printf("sum = %.3lf\n", sum);
    return 0;
}
```

3.19 7-26 谁最高

```
char s[100], cnt;

int main() {
    double a, b, c;
    scanf("%lf%lf%lf", &a, &b, &c);
    double max = a > b ? a : b;
    max = max > c ? max : c;
    if (max - a <= 0.001) {
        s[cnt] = ' ';
        s[cnt + 1] = 'A';
        cnt += 2;
```

```
    }
    if (max - b <= 0.001) {
        s[cnt] = ' ';
        s[cnt + 1] = 'B';
        cnt += 2;
    }
    if (max - c <= 0.001) {
        s[cnt] = ' ';
        s[cnt + 1] = 'C';
        cnt += 2;
    }
    s[cnt] = 0;
    printf("%s", &s[1]);
    return 0;
}
```

4 格式输出

格式输出挺烦的，要仔细。
建议先把答案存到数组里，再想办法套格式。

4.1 7-4 输出 2 到 n 之间的全部素数

暴力竟然能过。。正解应该是筛法。

```
int isprime(int n) {
    if (n < 3)
        return n == 2;
    if (n % 2 == 0)
        return 0;
    // 否则每次循环都要开根
    int sn = (int) sqrt(n * 1.0);
    for (int i = 3; i <= sn; i += 2)
        if (n % i == 0)
            return 0;
    return 1;
}

int cnt, prime[20000001];
void init(int n) {
    for (int i = 2; i <= n; i++) {
        if (isprime(i)) {
            prime[++cnt] = i;
```

```
        }
    }
}

int main() {
    int n;
    scanf("%d", &n);
    init(n);
    for (int i = 1 i < cnt; i++) {
        printf("%6d", prime[i]);
        if (i % 10 == 0)
            printf("\n");
    }
    printf("%6d", prime[cnt]);
    return 0;
}
```

4.2 7-5 输出前 n 个 Fibonacci 数

```
int cnt, fib[20000001];
void init(int n) {
    fib[1] = 1;
    fib[2] = 1;
    for (int i = 3; i <= n; i++)
        fib[i] = fib[i - 1] + fib[i - 2];
    cnt = n;
}

int main() {
    int n;
    scanf("%d", &n);
    if (n < 1) {
        printf("Invalid.");
        return 0;
    }
    init(n);
    for (int i = 1 i < cnt; i++) {
        printf("%11d", fib[i]);
        if (i % 5 == 0)
            printf("\n");
    }
    printf("%11d\n", fib[cnt]);
```

```
    return 0;
}
```

4.3 7-11 数字菱形

递归法

```
int n, h;

void printline(int l, int t) {
    for (int i = 0; i < l; i++)
        putchar(' ');
    printf("%d", t);
    if(l != h) {
        for (int i = 1; i < (h - l) * 2; i++)
            putchar(' ');
        printf("%d", t);
    }
    printf("\n");
}

void solve(int x) {
    printline(h - x, (n + x) % 10);
    if(x == h)
        return;
    solve(x + 1);
    printline(h - x, (n + x) % 10);
}

int main() {
    scanf("%d %d", &n, &h);
    h /= 2;
    solve(0);
    return 0;
}
```

循环法

```
#define INC(x) (x = (x + 1) % 10)
#define DEC(x) (x = (x - 1 + 10) % 10)

int main() {
    int a, h;
    scanf("%d %d", &a, &h);
```

```
    int b = h / 2;
    _fora(i, 1, b) printf(" ");
    printf("%d\n", a);
    INC(a);
    _fora(i, 1, b) {
        _fora(j, 1, b - i)
            printf(" ");
        printf("%d", a);
        _fora(j, 1, i * 2 - 1)
            printf(" ");
        printf("%d", a);
        INC(a);
        printf("\n");
    }
    DEC(a);
    DEC(a);
    _forz(i, b - 1, 1) {
        _fora(j, 1, b - i)
            printf(" ");
        printf("%d", a);
        _fora(j, 1, i * 2 - 1)
            printf(" ");
        printf("%d", a);
        DEC(a);
        printf("\n");
    }
    _fora(i, 1, b)
        printf(" ");
    printf("%d\n", a);
    return 0;
}
```

4.4 7-15 重排矩阵

排序后再想办法绕圈圈。

```
int nn[10086];
int mtx[10086];
int tm, tn;

void quick_sort(ll l, ll r) {
    if (l >= r)
        return;
```

```
8       int i = l, j = r;
9       int x = nn[(l + r) / 2];
10      while (i <= j) {
11          while (nn[j] > x)
12              j--;
13          while (nn[i] < x)
14              i++;
15          if (i <= j) {
16              int t = nn[i];
17              nn[i] = nn[j];
18              nn[j] = t;
19              i++;
20              j--;
21          }
22      }
23      quick_sort(l, j);
24      quick_sort(i, r);
25  }
26
27  int min(int a, int b) {
28      return a > b ? b : a;
29  }
30
31  int pos(int x, int y) {
32      return (y - 1) * tn + x;
33  }
34
35  int main() {
36      int n = rr();
37      _fora(i, 1, n)
38          nn[i] = rr();
39      quick_sort(1, n);
40      int sn = (int)sqrt(n);
41      tn = 1;
42      _forz(i, sn, 1) {
43          if (n % i == 0) {
44              tn = i;
45              break;
46          }
47      }
48      tm = n / tn;
49      int p = n;
```

```
    _fora(i, 1, (tn + 1) / 2) {
        if (p <= 0)
            break;
        _fora(j, i, tn - i)
            mtx[pos(j, i)] = nn[p--];
        if (p <= 0)
            break;
        _fora(j, i, tm - i + 1)
            mtx[pos(tn - i + 1, j)] = nn[p--];
        if (p <= 0)
            break;
        _forz(j, tn - i, i)
            mtx[pos(j, tm - i + 1)] = nn[p--];
        if (p <= 0)
            break;
        _forz(j, tm - i, i + 1)
            mtx[pos(i, j)] = nn[p--];
    }
    _fora(i, 1, tm) {
        _fora(j, 1, tn - 1)
            printf("%d ", mtx[pos(j, i)]);
        printf("%d\n", mtx[pos(tn, i)]);
    }
    return 0;
}
```

4.5 7-16 1000 以内所有各位数字之和为 n 的正整数

```
int cnt, nn[20000001];
void init(int n) {
    _fora(i, 1, 1000) {
        int sum = 0;
        int ti = i;
        while (ti) {
            sum += ti % 10;
            ti /= 10;
        }
        if (sum == n)
            nn[++cnt] = i;
    }
}

```

```
int main() {
    int n;
    scanf("%d", &n);
    init(n);
    _fora(i, 1, cnt - 1) {
        printf("%8d", nn[i]);
        if (i % 6 == 0)
            printf("\n");
    }
    printf("%8d\n", nn[cnt]);
    return 0;
}
```

4.6 7-22 嵌套循环-素数的和

```
int notp[100000001];
int cnt, prime[20000001];
void init(int n) {
    _fora(i, 2, n) {
        if (!notp[i])
            prime[++cnt] = i;
        int t = n / i;
        _fora(j, 1, cnt) {
            if (prime[j] > t)
                break;
            notp[i * prime[j]] = 1;
            if (i % prime[j] == 0)
                break;
        }
    }
}

int main() {
    init(100000);
    int a, b;
    scanf("%d%d", &a, &b);
    if (a > b) {
        int t = a; a = b; b = t;
    }
    if (a < 0)
        return 0;
    int sum = 0;
```

```
28      _fora(i, a, b) {
29          if (!notp[i])
30              sum += i;
31      }
32      printf("%d", sum);
33      return 0;
34  }
```