lesson 1

学习方法

对于没有计划参加算法竞赛的同学， acwing 算法基础课对企业笔试面试匹配得很好，可以直接参考学习。笔试之前到 leetcode 上刷题，基本可以满足需求。

对于有计划参加算法竞赛的同学，我（刁总）觉得 acwing 算法基础课的安排是稍显不合理的。我个人推荐的入门学习顺序是：

- 贪心
- 二分查找、二分答案、三分
- dfs、bfs 和 简单剪枝
- 栈、队列、二叉堆、并查集
- 线性 dp、背包 dp、区间 dp
- 位运算、整除理论、组合计数
- 图论基础、拓扑排序、最短路、最小生成树
- ST 表、最近公共祖先
- DAG 上 dp、树形 dp、状压 dp
- 单调栈和单调队列
- 树状数组和线段树
- 字符串哈希、kmp 和 trie 树

关于学习方式，可以到 洛谷 找以【模板】开头的题目题解学习，然后做官方和用户分享的题单中的简单题。

这部分一定不要花太多时间，争取在大一下学期之前结束。

系统学习完上面的内容后，就可以开始到 CodeForces 上刷题了。可以在 problemset 中筛选题目难度，从 rating 1200 开始一道一道做。遇到不会的科技根据题解中提到的关键字去找模板题学习。如果有时间也可以打打这上面的比赛，作为自己阶段学习成果的证明。

AtCoder 上的 Atcoder Beginner Contest 是很好的锻炼自己算法模板和常见套路的地方。这里面题通常比较简单，属于是学过了就肯定能做出来。

对高级科技有需求的，到 OIwiki 目录里面找不会的学。比如主席树、平衡树、网络流等。在 XCPC 中一般不作为铜牌题及以下的点考察。

学习过程中可以整理一个自己的算法模板库，引流，因为 XCPC 是开卷考试。

推荐阅读：ACM 怎么样零基础到入门？ - geruome的回答 - 知乎

时间复杂度

衡量算法快慢的函数。大 O 记号表示上界。规定计算机进行一次基本运算（加减乘除模、swap、比较等）所用时间为一个小常数。

- 冒泡、选择排序：$O(n^2)$
- 快速、归并排序：$O(n \log n)$

假设一个算法消耗时间的函数为

$$T(n)=\begin{cases}1 & (n=1)\\ T\left(\frac{n}{2}\right)+2n & (n>1)\end{cases}$$

展开可得 $T(n)=n+2n\log_2 n$.

忽略掉常数项和低阶项，用大 O 记号表示为 $O(n\log n)$.

- $O(n)$

```
for (int i = 1; i <= n; ++i) {
    x += 1;
}
```

- $O(n^2)$

```
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
        x += 1;
    }
}
```

- $O(n^3)$

```
string s[n + 1], t[n + 1];
// s[i], t[i] 长度 <= n
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
        if (s[i] == t[j]) ++cnt;
    }
}
```

- $O(\sqrt{n})$

```
for (int i = 2; i * i <= n; ++i) {
    if (n % i == 0) break;
}
```

- $O(n\log n)$

```
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j * i <= n; ++j) {
        x += 1;
    }
    // 调和级数
}
```

二分答案

前期学习中的一个**重难点**

使用二分答案要满足两个条件：

- 答案满足单调性

- 容易验证一个解是否是满足题意的。
- 及时return `check` 函数有时会避免一些可能的错误
- 考虑边界问题

[路标设置]

[数字组合]

位运算

可以思考，不需要硬记

功能	示例	式子
去掉最后一位	10110 → 1011	x >> 1
在最后添加 0	1011 → 10110	x << 1
在最后添加 1	1011 → 10111	(x << 1) | 1
右数第 k 位变成 1	100011 → 101011，k = 4	x | (1 << (k − 1))
右数第 k 位变成 0	101011 → 100011，k = 4	x & ~(1 << (k − 1))
获取右数第 k 位	101011 → 1，k = 4	(x >> (k − 1)) & 1
截取最后 k 位	101011 → 1011，k = 4	x & ((1 << k) − 1)
把右边连续的 1 变成 0	101011 → 101000	[illegible]
把右起第一个 0 变成 1	101011 → 101111	[illegible]
把右边连续的 0 变成 1	101000 → 101111	[illegible] 思考
把右起第一个 1 变成 0	101000 → 100000	[illegible]
取右边连续的 1	101111→ 1111	(x ^ (x + 1)) >> 1

```
//判断一个数字是否是2的幂次
{
    int x;
    bool f = (x & (x - 1)) == 0;
}

//计算二进制有几个1
{
    int x, c = 0;
    for(;x;c++){
        x = x & (x - 1)
    }
}

void swap(){
    b ^= a;
    a ^= b;
    b ^= a;
}

//得到右边第一个1和后面的0构成的数
int lowbit(int x){
    return x & (-x);
}

//判断两数符号是否相同
{
    bool f = ((x ^ y) < 0);
```

```
}

{
    if(x == y) ->  x ^ y == 0;
    0 | x == x
}

a + b =  (a ^ b) + ((a & b) << 1)

a - b = a + (~b + 1)

a ^ b == ~(a & b) & (a | b)

for (int i = x; i; i = (i - 1) & x) //子集枚举
```

例题选讲

1. NIT orz!
2. Odd One Out
3. XOR Sequences
4. XOR Mixup