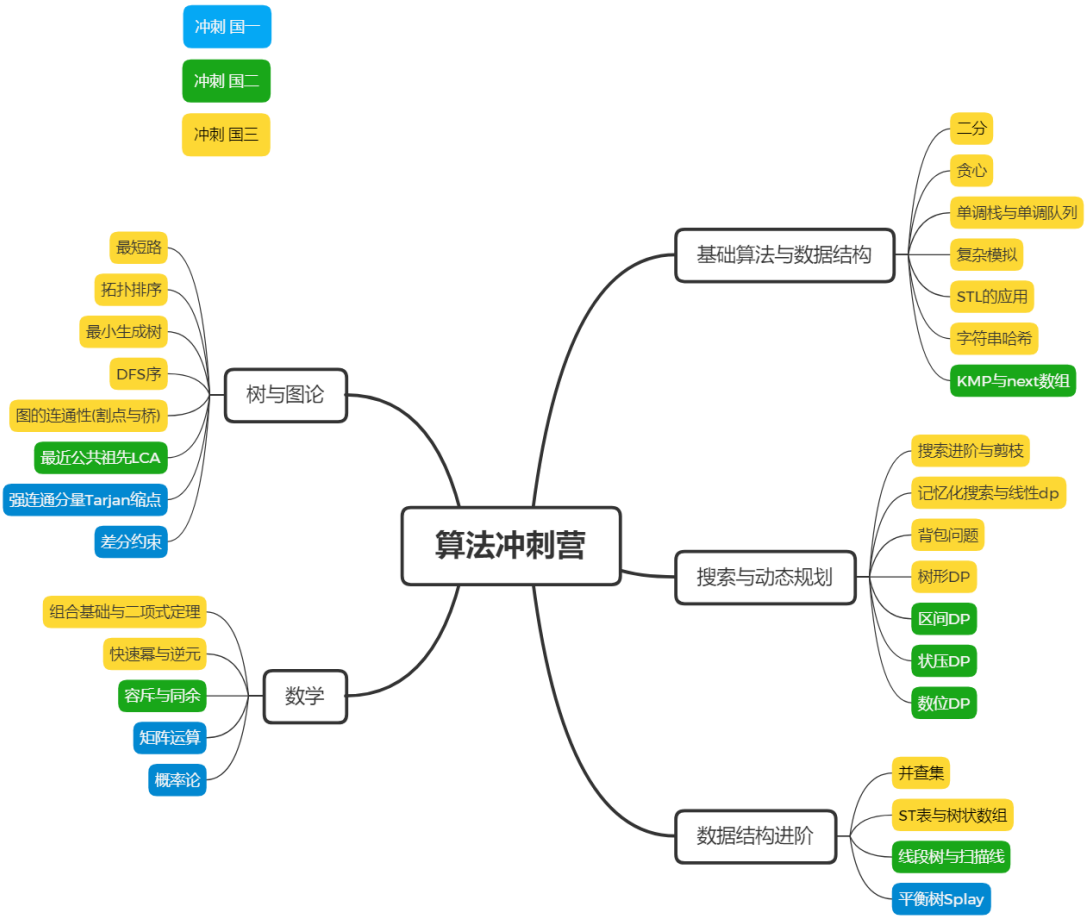
蓝桥杯30天算法冲刺集训



Day-4 模拟

模拟没有什么特别好讲的技巧，每个人的模拟方式也都略有不同，所以目前给出了四种比较常见的模拟类型，并且给出相应的例题。同学们只需要把题做做就好，今天的内容如果是有基础的同学，也可以跳过。不过在蓝桥杯国赛当中模拟也是不论是直接出模拟题还是水分都是重中之重的，所以还是希望大家能够掌握。如果说下面的题目模拟起来还是有些难度的话，也没关系，代码底力也会随着学习编程而变强的。

蓝桥杯真题

字符串模拟

内存空间(蓝桥杯Python2022B组国赛)

本题需要在模拟之前think twice，可以达到事半功倍的效果。首先定义语句只有两种，一种是定义变量，一种是定义数组。对于定义变量来说我们有`int`，`long`和`String`三种类型，对于前两种类型，变量的值不重要只需要知道有几个变量即可，我们再观察下面的定义语句。

```
type var1=value1,var2=value2...;
```

可以看到有几个变量就是有几个`=`，那么我们直接计算等号个数就好了。

但是对于是`String`的变量定义并不一样还有额外的`"`，如样例所示：

`String s1="hello",s2="world";`

我们要计算出来这里面总共有十个字符也就是有10B，那么我们只需要开一个状态，状态是0的时候遇到引号就状态变换为1并开始开始计算字符个数，再遇到引号就状态变换成0，总体答案记录一下，一直往复循环即可。

对于数组定义语句如下：

`type[] arr1=new type[size1],arr2=new type[size2]…;`

可以看到需要计算里面的`size`只需要找到中括号即可，模拟思路同上面的`String`类型定义语句类似。

最后还需要转换成对应的格式，与时间转换是一样的，直接除数取模即可。

```
def solve(x):
    mod = 1024
    b = x % mod
    kb = (x // mod) % mod
    mb = (x // (mod*mod)) % mod
    gb = (x // (mod*mod*mod)) % mod
    res = ''
    if gb:
        res += str(gb) + 'GB'
    if mb:
        res += str(mb) + 'MB'
    if kb:
        res += str(kb) + 'KB'
    if b:
        res += str(b) + 'B'
    return res

n = int(input())
num = 0
for _ in range(n):
    pre = input()
    cur = list(pre.split())
    op = cur[0]
    if op == 'int':
        num += len(list(cur[1].split(','))) * 4
    elif op == 'long':
        num += len(list(cur[1].split(','))) * 8
    elif op == 'String':
        for i in list(cur[1].split(',')):
            k = i.index('=')
            num += len(i) - (k+1+2)
        num -= 1
    elif op == 'long[]':
        pre = pre[7:-1]
        u = list(pre.split(','))
        for i in u:
            start = i.index('[')
            b = i[start+1:-1]
            num += 8*int(''.join(b))
    else:
        pre = pre[6:-1]
        u = list(pre.split(','))
        for i in u:
```

```
            start = i.index('[')
            b = i[start+1:-1]
            num += 4*int(''.join(b))
print(solve(num))
```

进制与计算模拟

小数第n位(蓝桥杯C/C++2017C组国赛)

这题虽然数据范围很大，但是说不定模拟也能得到80分，也是不错的练习模拟的题目。

我们思考竖式除法的过程。针对当前的被除数 x（其中 $x < b$），我们获得的商是 $\lfloor \frac{10x}{b} \rfloor$，然后将当前的数更新为 $(10x) \bmod b$。因为 $x < b$，所以得到的商是一个 $[0, 9]$ 内的整数，而后续的数仍然是模 b 的余数。

我们可以注意到，这个过程实际上是不断将 x 更新为 $(10x) \bmod b$，直到达到我们想要的那个位置的前一个位置，获得一个数 y，然后计算 $\lfloor \frac{10y}{b} \rfloor$ 就是所求的商。

由于取模的性质，我们可以推知 $y = 10^k x \bmod b$。为了计算 $10^k \bmod b$，我们可以运用快速幂算法。

然后接着模拟求商过程，模拟三次即可。

```
def ksm(a, b, mod):
    ret = 1
    while b > 0:
        if b & 1 == 1:
            ret = ret * a % mod
        a = a * a % mod
        b >>= 1
    return ret

if __name__ == "__main__":
    a, b, n = map(int, input().split())
    t = a * ksm(10, n - 1, b) % b

    for i in range(3):
        print(int(t * 10 / b), end="")
        t = int(t * 10 % b)

    print()
```

练习题

树的模拟

中缀表达式

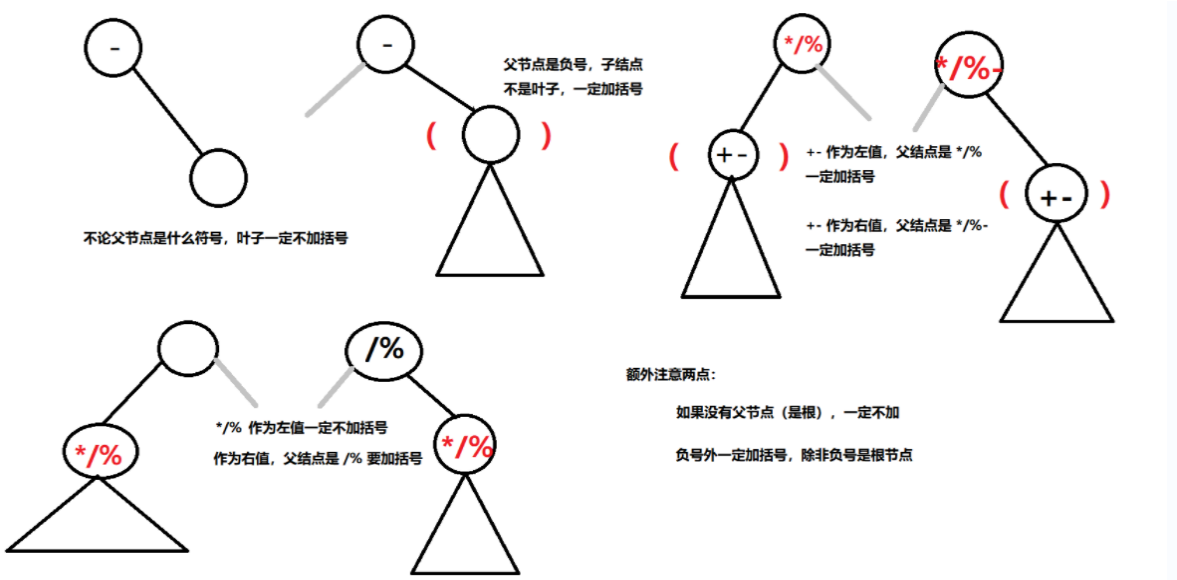
可以看作是一道模拟题，考察表达式树的遍历。考察两点：

1. 表达式树的遍历（中序遍历输出即可）
2. 添加括号（难点，对于每一棵子数，都需要判断是否要添加括号）

具体添加括号的原则，则和运算符的优先级相关，想到这里，你就需要去分析有哪些情况需要添加，或者不添加括号：

1. 叶子一定不加括号

2. 负号外一定加括号（否则就可能会出现连续两个运算符，导致表达式有歧义）
3. 负号内，如果不是叶子，一定加括号
4. `+-` 加减号，作为左值，碰到 `*/%` 作为父节点符号一定加括号，作为右值，只要父节点不是 `+` 就一定加括号
5. `*/%` 作为左值，一定不加括号，作为右值，碰到 `/%` 加括号




```
N = 10005

def need(u):
    global isleaf, w, fa, l, r
    if isleaf[u]:
        return False
    if fa[u] and w[fa[u]] == "-" and l[fa[u]] == -1:
        return True
    if w[u] == "-" and l[u] == -1:
        return True
    if w[u] == "-" or w[u] == "+":
        if fa[u]:
            if r[fa[u]] == u:
                if w[fa[u]] != "+":
                    return True
            else:
                if w[fa[u]] != "+" and w[fa[u]] != "-":
                    return True
    if w[u] == "*" or w[u] == "/" or w[u] == "%":
        if fa[u]:
            if l[fa[u]] == u:
                return False
            else:
                if w[fa[u]] == "/" or w[fa[u]] == "%":
                    return True
    return False

def dfs(u):
    global w, l, r
    lt, rt = "", ""
    if l[u] != -1:
        lt = dfs(l[u])
        if need(l[u]):
```

```
            lt = "(" + lt + ")"
    if r[u] != -1:
        rt = dfs(r[u])
        if need(r[u]):
            rt = "(" + rt + ")"
    return lt + w[u] + rt

if __name__ == "__main__":
    n = int(input())
    w = [""] * N
    l = [0] * N
    r = [0] * N
    fa = [0] * N
    isleaf = [False] * N

    for i in range(1, n + 1):
        w[i], l[i], r[i] = input().split()
        l[i], r[i] = int(l[i]), int(r[i])
        if l[i] != -1:
            fa[l[i]] = i
        if r[i] != -1:
            fa[r[i]] = i
        if l[i] == -1 and r[i] == -1:
            isleaf[i] = True

    for i in range(1, n + 1):
        if not fa[i]:
            print(dfs(i))
            break
```

图形模拟

小蓝的矩阵

本题中对于`#`构成的图像有平移和旋转 $90°$ 两种操作。

关键在于一个矩形中`#`构成的图像是否能通过旋转 $90°$，$180°$，$270°$ 得到另一个矩阵中`#`构成的图形。只要能通过旋转得到相同的图形，图形的位置只要通过平移操作移动到相同位置即可。

比如对于样例 1

```
4
.###
.##.
....
....
....
..##
.###
....
```

只要将第一个矩阵中`#`构成的图形旋转 $180°$，再向下平移一次即可。

对于这道题我们如何模拟呢，其实思路很简单：

- 先用三个二维数组存储下`#`构成的图形旋转 $90°$，$180°$，$270°$ 之后的图形

- 再把原来的图形和旋转 $90°$，$180°$，$270°$ 之后的总共四个图形依次与第二个矩阵的`#`构成的图形进行比较
- 如果图形旋转后形状与第二个矩阵相同，每个对应的点偏移量也就是横纵坐标之差 $x_1 - x_2$ 和 $y_1 - y_2$ 都应该相同，我们再拿样例举个例子帮助大家理解这个偏移量的概念

```
..##
.###
第一个点平移后这5个`#`坐标从上到下从左到右依次为：
     (1,3)(1,4)
(2,2)(2,3)(2,4)

....
..##
.###
     (2,3)(2,4)
(3,2)(3,3)(3,4)

所有对应点的横坐标之差为-1，纵坐标之差0
```

```
N = 205
n, m, k, sum = 0, 0, 0, 0
s = ['' for _ in range(N)] for _ in range(N)]
t = ['' for _ in range(N)] for _ in range(N)]
m2 = ['' for _ in range(N)] for _ in range(N)]
m3 = ['' for _ in range(N)] for _ in range(N)]
m4 = ['' for _ in range(N)] for _ in range(N)]
vt = []

def check(a):
    va = []
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            if a[i][j] == '#':
                va.append((i, j))
    if len(va) != len(vt):
        return False
    else:
        mx = va[0][0] - vt[0][0]
        my = va[0][1] - vt[0][1]
        for i in range(1, len(va)):
            if va[i][0] - vt[i][0] != mx or va[i][1] - vt[i][1] != my:
                return False
        return True

def main():
    global n
    n = int(input())
    for i in range(1, n + 1):
        s[i][1:] = input()
    for i in range(1, n + 1):
        t[i][1:] = input()
        for j in range(1, n + 1):
            if t[i][j] == '#':
                vt.append((i, j))

    for i in range(1, n + 1):
```

```
        for j in range(1, n + 1):
            m2[i][n - j + 1] = s[j][i]
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            m3[i][n - j + 1] = m2[j][i]
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            m4[i][n - j + 1] = m3[j][i]

    if check(s) or check(m2) or check(m3) or check(m4):
        print("Yes")
    else:
        print("No")

if __name__ == "__main__":
    main()
```