

## 第一题

朋辈辅导

biao

2024.10.19

给你两段代码，求时间复杂度。

去年第二小问是  $1^2+2^2+3^2+\dots+n^2$ ，有个公式

$$\text{令 } S_n = (1^2 + 2^2 + 3^2 + \dots + n^2)$$

$$\text{则 } S_n - S_{n-1} = n^2 \quad (n \geq 2)$$

$$\text{设 } S_n + An^3 + Bn^2 + Cn = S_{n-1} + A(n-1)^3 + B(n-1)^2 + C(n-1)$$

$$\text{整理得 } S_n - S_{n-1} = -3An^2 + (3A - 2B) + B - A - C$$

$$\text{对比系数得 } A = -\frac{1}{3} \quad B = -\frac{1}{2} \quad C = -\frac{1}{6}$$

$$\text{则 } S_n - \frac{1}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n = S_{n-1} - \frac{1}{3}(n-1)^3 - \frac{1}{2}(n-1)^2 - \frac{1}{6}(n-1) \quad (n \geq 2)$$

$$\text{即 } S_n - \frac{1}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n = S_{n-1} - \frac{1}{3} - \frac{1}{2} - \frac{1}{6} = 1 - \frac{1}{3} - \frac{1}{2} - \frac{1}{6} = 0$$

$$\text{所以 } S_n = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n = \frac{n(n+1)(2n+1)}{6}$$

同时可以关注一下，增长率比较，群文件有之前学长姐整理的一个资料，可以自行看看。

### 算法时间效率的比较

- 当  $n$  取得很大时，指数时间算法和多项式时间算法在所需时间上非常悬殊

$n$	$f(1)$	$f(\log n)$	$f(n)$	$f(n \log n)$	$f(n^2)$	$f(n^3)$	$f(2^n)$	$f(n!)$
1	1	0	1	0	1	1	2	1
2	1	1	2	2	4	8	4	2
4	1	2	4	8	16	64	16	24
8	1	3	8	24	64	512	256	40320
16	1	4	16	64	256	4096	65536	2.0923E+13
32	1	5	32	160	1024	32768	4.295E+09	2.6313E+35

时间复杂度  $T(n)$  按数量级递增顺序为：

常数阶	对数阶	线性阶	线性对数阶	平方阶	立方阶	...	K次方阶	指数阶
$O(1)$	$O(\log_2 n)$	$O(n)$	$O(n \log_2 n)$	$O(n^2)$	$O(n^3)$	...	$O(n^k)$	$O(2^n)$

复杂度低 → 复杂度高

## 第二题

貌似是稀疏矩阵？存储地址？有一题是用 i，j 表示 k

没什么印象了，但是不难

下三角矩阵的压缩存储\*

下图是一个 4 阶的下三角矩阵。

$$\begin{pmatrix} a_{11} & & & 0 \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

对于任意  $n$  ( $n > 0$ ) 阶下三角矩阵，如果采用压缩存储的方法，将下三角区域 ( $i \geq j$ ) 中的  $n(n+1)/2$  个元素按顺序保存在一维数组  $p$  中，则一维数组中的元素  $p[k]$  ( $0 \leq k < n$ ) 与原矩阵下三角区域中的元素  $a_{ij}$  ( $1 \leq j \leq i \leq n$ ) 之间存在着——对应的关系：

$k =$ 

i\*(i-1)/2+j-1

5 分

 $, (j \leq i)$

## 第三题 广义表！！！！

一定要会画相应的结构

GetHead 得到的是元素，GetTail 得到的是广义表！！！！

求下列广义表操作的结果。

GetHead [ GetHead [ GetTail [ GetHead [ GetTail [ (apple,(pear,(banana),orange)) ] ] ] ] ]

banana

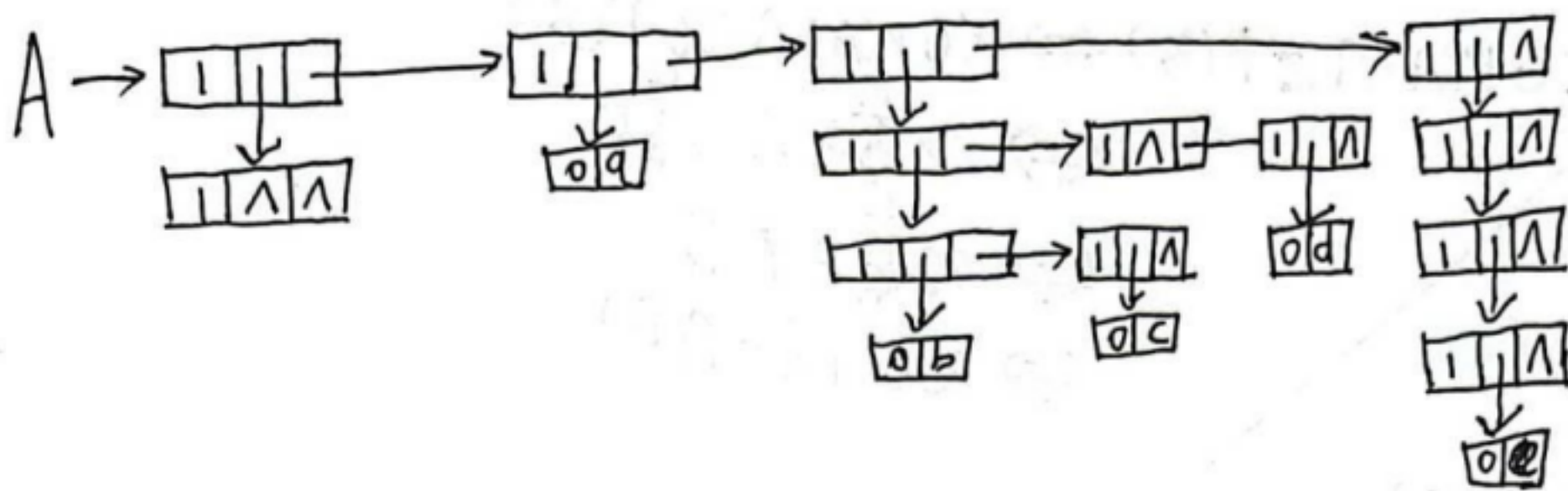
3 分

请画出下列广义表的存储结构图

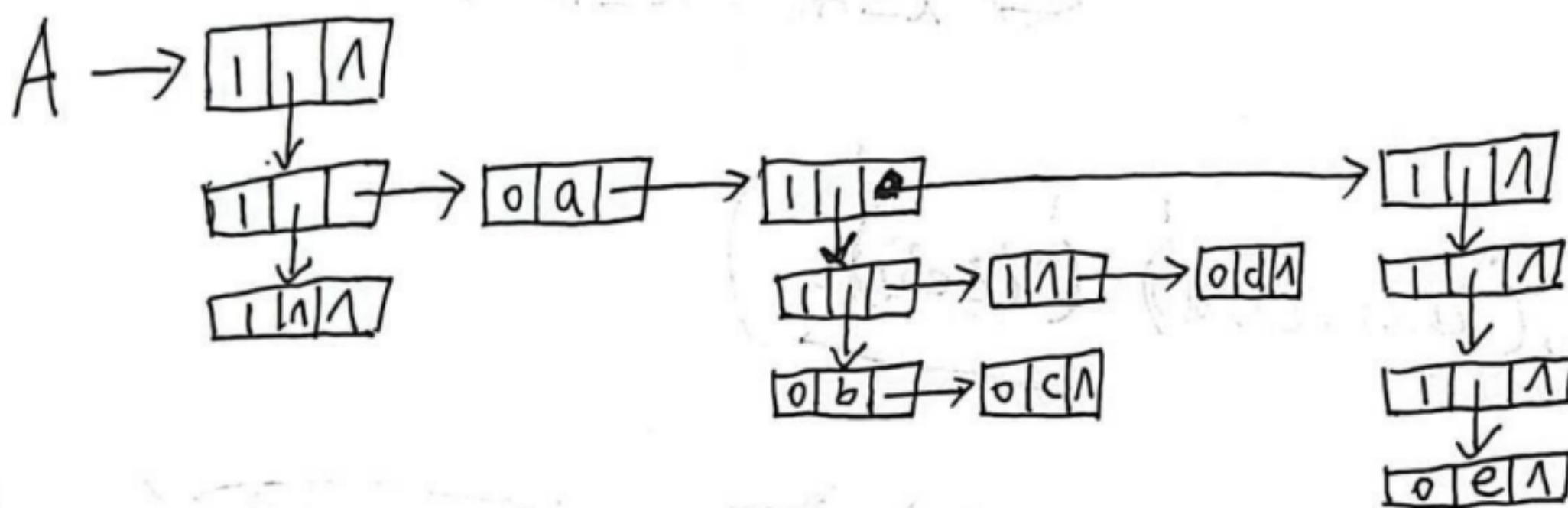
((()),a,((b,c),(),d),(((e))))



① 头尾链表存储表示: 用A表示



② 扩展线性链表存储表示: 用A表示



#### 第四题 KMP

重点掌握 kmp 朴素和优化的 `next[j]` 怎么数，算法尽量掌握。

假设模式串是 `abababaab`，则KMP模式匹配算法中的`next[j]` = `0 1 1 2 3 4 5 6 2`。

根据next值求nextval值的方法:

模式串	a	b	a	a	b	c	a	c
next值	0	1	1	2	2	3	1	2
nextval值	0	1	0	2	1	3	0	2

- 1.第一位的nextval值必定为0，第二位如果于第一位相同则为0，如果不同则为1。
- 2.第三位的next值为1，那么将第三位和第一位进行比较，均为a，相同，则第三位的nextval值为第一位的next值，为0。
- 3.第四位的next值为2，那么将第四位和第二位进行比较，不同，则第四位的nextval值为其next值，为2。
- 4.第五位的next值为2，那么将第五位和第二位进行比较，相同，第二位的next值为1，则继续将第二位与第一位进行比较，不同，则第五位的nextval值为第二位的next值，为1。
- 5.第六位的next值为3，那么将第六位和第三位进行比较，不同，则第六位的nextval值为其next值，为3。
- 6.第七位的next值为1，那么将第七位和第一位进行比较，相同，则第七位的nextval值为0。
- 7.第八位的next值为2，那么将第八位和第二位进行比较，不同，则第八位的nextval值为其next值，为2。



## 第五题 树

给出遍历顺序画结构

二叉转森林

**树与二叉树的转换**

- 将树转化为二叉树进行处理，利用二叉树的算法来实现对树的操作。
- 由于树和二叉树都可以用二叉链表作存储结构，则以**二叉链表作媒介**可以导出树与二叉树之间的一个对应关系。

给定一棵树，可以找到唯一的一棵二叉树与之对应

存储 解释 存储 解释

树 二叉树

《数据结构与算法基础》 青岛大学 数据科学与软件工程学院

**将树转换成二叉树**

- 加线：在兄弟之间加一连线
- 抹线：对每个结点，除了其左孩子外，去除其与其余孩子之间的关系
- 旋转：以树的根结点为轴心，将整树顺时针转45°

树变二叉树：兄弟相连留长子

《数据结构与算法基础》 青岛大学 数据科学与软件工程学院

**例：将树转换成二叉树**

树变二叉树：兄弟相连留长子

《数据结构与算法基础》 青岛大学 数据科学与软件工程学院

**将二叉树转换成树**

- 加线：若p结点是双亲结点的左孩子，则将p的右孩子，右孩子的右孩子.....沿分支找到的所有右孩子，都与p的双亲用线连起来
- 抹线：抹掉原二叉树中双亲与右孩子之间的连线
- 调整：将结点按层次排列，形成树结构

二叉树变树：左孩右右连双亲，去掉原来右孩线。

《数据结构与算法基础》 青岛大学 数据科学与软件工程学院

**例：将二叉树转换成树**

二叉树变树：左孩右右连双亲，去掉原来右孩线。

《数据结构与算法基础》 青岛大学 数据科学与软件工程学院



## 第六题 代码填空题

栈，树，串，两个程序填空，这里最好是直接背诵 ppt 的代码，因为基本上 ppt 上已经有代码挖空，和平时我们自己写的代码风格不一样。

## 第七题

树的前序遍历，中序遍历，后序遍历，层序遍历，都要掌握，平时的作业重点关注

去年考的是完全二叉树的层序遍历，也就是作业题最后一题，所以作业也很重要，写完整代码。

### 4.4 完全二叉树的层序遍历

分数 25

全屏浏览 切换布局

作者 陈越 单位 浙江大学

一个二叉树，如果每一个层的结点数都达到最大值，则这个二叉树就是**完美二叉树**。对于深度为  $D$  的，有  $N$  个结点的二叉树，若其结点对应于相同深度完美二叉树的层序遍历的前  $N$  个结点，这样的树就是**完全二叉树**。

给定一棵完全二叉树的后序遍历，请你给出这棵树的层序遍历结果。

**输入格式：**

输入在第一行中给出正整数  $N$  ( $\leq 30$ )，即树中结点个数。第二行给出后序遍历序列，为  $N$  个不超过 100 的正整数。同一行中所有数字都以空格分隔。

**输出格式：**

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int n, cnt, In[32], dfs[32];
4  void Func(int index) {
5      if (index > n) return;
6      Func(index << 1);
7      Func(index << 1 | 1);
8      dfs[index] = In[cnt++];
9  }
10 int main() {
11     cin >> n;
12     for (int i = 0; i < n; i++) cin >> In[i];
13     Func(1);
14     cout << dfs[1];
15     for (int i = 2; i <= n; i++) cout << " " << dfs[i];
16     return 0;
}
```