

1



时间复杂度

【考点笔记】时间复杂度

时间复杂度T(n)是指算法中所有语句的频度(执行次数)之和。

人们关心的是 $\frac{1}{2}$ n趋于无穷时T(n)的数量级,而非T(n)的准确大小,因此以T(n)的数量级来表征时间复杂度。

数量级最大的一项必定是由最深层循环的语句贡献的,称之为基本运算。

T(n)与算法中基本运算的频度f(n)同数量级,所以通常采用基本运算的频度的数量级O(f(n))来分析算法的时间复杂度,记为T(n) = O(f(n))。

并列多个循环, 满足加法规则 只保留更高阶的项

a)加法规则 T(n) = T₁(n) + T₂(n) = O(f(n)) + O(g(n)) = O(<mark>max</mark>(f(n), g(n)))

嵌套多个循环, 满足乘法规则 b) 乘法规则

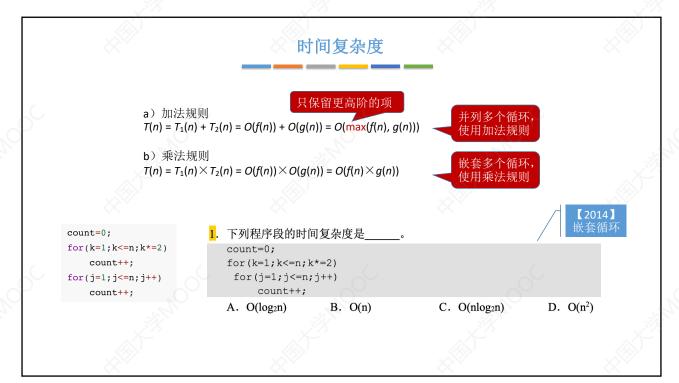
 $T(n) = T_1(n) \times T_2(n) = O(f(n)) \times O(g(n)) = O(f(n) \times g(n))$

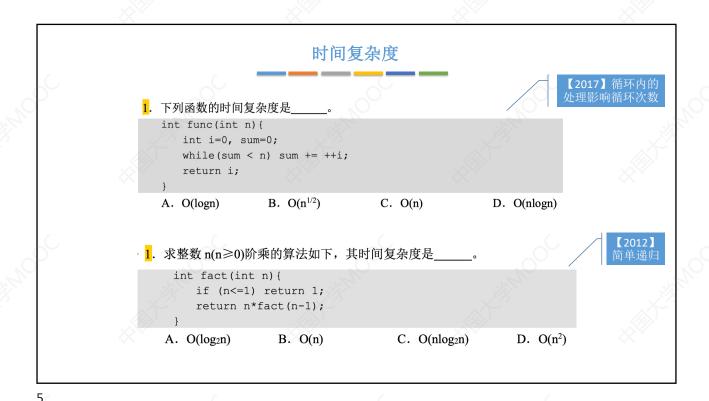


常对幂指阶

数据结构 +20分

3





空间复杂度

【考点笔记】空间复杂度

空间复杂度S(n)指算法运行过程中所使用的辅助空间的大小,通常结合算法题考查。

算法原地工作是指算法所需辅助空间是常量,即0(1)。

注意影响辅助空间大小的两个方面:

- ①算法运行过程中各种变量所占空间(如:辅助数组)
- ②递归工作栈带来的空间复杂度(通常和递归深度同等数量级)

注意:有些情况下,算法的时间复杂度、空间复杂度还随问题的输入数据集的不同而不同 (最好、最坏、平均)——常结合查找、排序算法考察

时间&空间复杂度结合算法题考查

41. (13 分)定义三元组(a,b,c)(a,b,c 均为整数)的距离 D=[a-b]+ |b-c|+|c-a|。给定3个非空整数集合S1、S2和S3,按升序分别存 储在3个数组中。请设计一个尽可能高效的算法,计算并输出所 有可能的三元组 $(a,b,c)(a \in S1,b \in S2,c \in S3)$ 中的最小距离。例 如, $S1 = \{-1,0,9\}$, $S2 = \{-25,-10,10,11\}$, $S3 = \{2,9,17,30,41\}$, 则最小距离为 2,相应的三元组为(9,10,9)。要求:

Tips: 通常算法题的最后一 个问题是分析复杂度,需要 基于自己设计的算法来分析, 即便算法不是最优的,但复 杂度分析正确该小问依然可 得满分

- (1) 给出算法的基本设计思想。
- (2) 根据设计思想,采用 C 或 C++语言描述算法,关键之处给出 注释。
- (3) 说明你所设计算法的时间复杂度和空间复杂度

```
int findMinDist(int[] S1, int[] S2, int[] S2, int N1, int N2, int N3) {
int min = INT MAX;
 for (int i=0; i<N1; i++)
  for (int j=0; j<N2; j++)
    for (int k=0; k<N3; k++)
      //计算 (S1[i], S2[j], S3[k])的距离, 并更新 min
```

时间复杂度=O(N1·N2·N3)

空间复杂度=O(N1+N2+N3)



空间复杂度=O(1)



考点 1: 时间复杂度与空间复杂度

【命题重点】

- 1. 给定代码段,分析算法的时间复杂度。 循环类代码——分析每层循环的循环次数与问题规模n之间的关系,若有多层嵌套循环,则使用乘法规则 递归类代码——分析<mark>递归次数</mark>与问题规模n之间的关系。若要分析<mark>空间复杂度</mark>,则应找到递归深度与问题规模 n之间的关系。
- 2. 结合算法题考查,分析自己设计的算法的时、空复杂度。

