1.1概述

2019年1月25日　　20:01

- 本课程结构
  - 数据结构
  - 算法
  - 设计模式
- 课程说明
  - 以喷为主
  - 实例伪代码
  - 各种语言版本实现

数据结构是算法的基础
①难
②语言非相关
③部分代码实现

概念　逻辑结构
物理结构

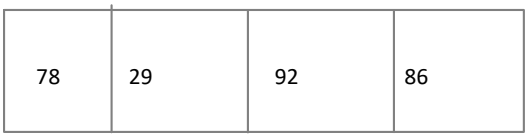
①基本类型--在数据结构关注较多
②操作--在算法关注较多
Element　e;
①age
②name
③self_intro

- 数据结构
  - 数据类型加操作

抽象数据类型
AbstractData Type

存储全班成绩：　78,29,92,89

在内存中连续存储



分为两个结构

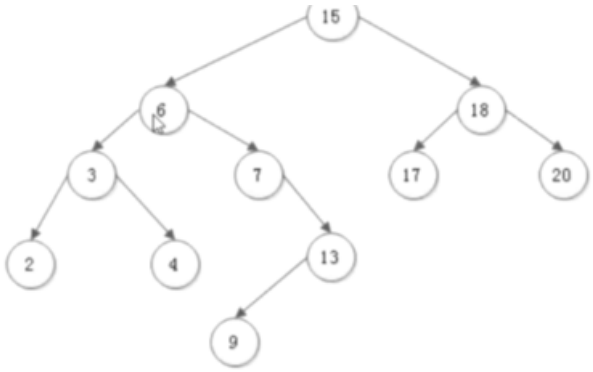
逻辑结构 :在内存中不一定是连续的。

物理结构

在硬盘上的真是存在结构

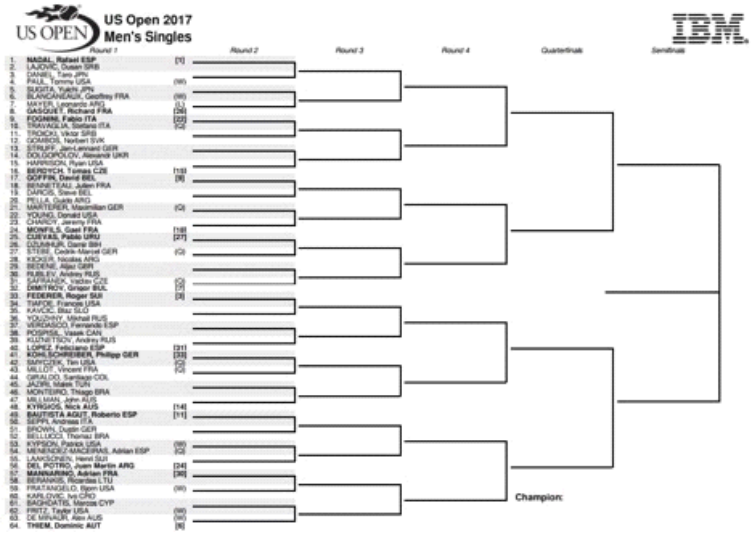
- 算法
  - 解决问题的方法，或者叫做计算方法和步骤
    - 例如：找到全班成绩最大的同学
      1. 按个比较
      2. 把男女同学分队，在男同学中查找。
      3. 画一个二叉树

a)



4. 锦标赛算法

a)



- 算法的五大特性
  - 输入总数大于等于0
  - 输出数量大于等于1
  - 有穷性
  - 确定性
  - 可行性

可行性：分布式一致性问题是不能完美解决的
拜占庭将军--信使
舍弃一定的可行性，在一定程度的条件下解决一定的问题

- 程序= ds+algo

data structure + algorithm

  - 程序 = 数据结构和算法
- 为什么要研究它?
  - 是研究程序的本质。解决问题的方法。怎么用是最基本的。数据结构和算法是用来解决问题的一种方法。

案例

01 百钱买百鸡问题。

今有鸡翁一，值钱伍；鸡母一，值钱三；鸡鶵三，值钱一。凡百钱买鸡百只，问鸡翁、母、鶵各几何？答曰：鸡翁四，值钱二十；鸡母十八，值钱五十四；鸡鶵七十八，值钱二十六。又答：鸡翁八，值钱四十；鸡 母十一，值钱三十三，鸡鶵八十一，值钱二十七。又答：鸡翁十二，值钱六十；鸡母四、值钱十二；鸡鶵八十 四，值钱二十八。”

解决:

数学解法:

1. x+y+z=100
2. 5x+3y+(1/3)z=100

解上面方程组得到正整数解即可解决该问题。

C语言

```c
#include <stdio.h>
void main()
{
    int cocks=0,hens,chicks;

while(cocks<=20)
    {
        hens=0;
        while(hens<=33)
        {
            chicks=100-cocks-hens;
            if(5.0*cocks+3.0*hens+chicks/3.0==100.0)

printf("公鸡%d只，母鸡%d只，小鸡%d只\n",cocks,hens,chicks);
            hens++;
        }
        cocks++;
    }
}
```

Python

```python
money = 100
score = 0
for g in range(1,21):
    for m in range(1,34):
        for x in range(1,301):
            score = g*5 + m*3 + float(x)/3
            if score == money and g+m+x ==100:
                print ('公鸡 %s 只，母鸡 %s 只,小鸡 %s 只' % (g,m,x))
            else:
                pass
```

Java

以上代码有轻度bug，自己去调试。

```java
public class BaiJiwenti
{
public static void main (String []
args)
{
for (int x = 0; x <= 19; x++)
{
for (int y = 0; y <= 33; y++)
{
int z = 100 - x - y;
if((x * 5 + y * 3 + z / 3 == 100 ) && z % 3 == 0)
{
System.out.println("可买鸡翁只数:" + x);
System.out.println("可买鸡母只数:" + y);
System.out.println("可买鸡雏只数:" + z);
}
}
}
}
}
```

算法衡量

- 衡量算法应该剔除机器配置，运算数量等无关因素。
- 时间复杂度
- 空间复杂度

实际运算要考虑
有些算法比较笨，但是在数量大的时候效果变好了

- 大O记法

*“大O记法”：对于单调的整数函数f，如果存在一个整数函数g和实常数c>0，使得对于充分大的n总有f(n)<=c*g(n)，就说函数g是f的一个渐近函数（忽略常数），记为f(n)=O(g(n))。也就是说，在趋向无穷的极限意义下，函数f的增长速度受到函数g的约束，亦即函数f与函数g的特征相似。*

时间复杂度：假设存在函数g，使得算法A处理规模为n的问题示例所用时间为T(n)=O(g(n))，则称O(g(n))为算法A的渐近时间复杂度，简称时间复杂度，记为T(n)

扩展：（如有困难，关注许老师的数学课）

在统计学中，我们研究的是具体的随机变量的性质（“估计”），这也就是这些数据的作用。在渐近分析中，当样本大小变得任意大时，我们专注于描述这种估计性质。当给定一个相当大的数据集，在有限的样本与任意大小样本中，这种性质很相似。

大样本统计（渐近理论)就是指当研究对象的统计量趋于无穷大时的统计方法,用该种方法得到的概率结果收敛于某一常数，即对象总体均值。 其数学表达为：以样本均值$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$[1]估计对象总体均值$\mu$，在$n\to\infty$时，$\overline{X}_n$以概率1收敛于$\mu$。这种统计方法称为大样本统计方法。

渐进等价 [编辑]

定义：给定关于自然数n的复函数f和g，

命题$f(n) \sim g(n)\ (n \to \infty)$表明（使用小o符号）

$f(n) = g(n) + o(g(n))\ (n \to \infty)$

或（等价记法）

$f(n) = (1 + o(1))g(n)\ (n \to \infty)$。

这说明，对所有正常数ϵ，存在常量N，使得对于所有的$n \geqslant N$有

$|f(n) - g(n)| \leqslant \epsilon|g(n)|$。

当$g(n)$不是0或者趋于无穷大时，该命题可等价记作

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1。$$

渐进等价是一个关于n的函数的集合上的等价关系。非正式地，函数f的等价类包含所有在极限情况下近似等于f的函数g。

渐近展开 [编辑]

主条目：渐近展开

函数$f(x)$的渐近展开是它的一种级数展开。这种展开的部分和未必收敛，但每一个部分和都表示$f(x)$的一个渐近表示式。例子：斯特灵公式。

- 时间复杂度计量

    - 最优时间复杂度
    - 最坏时间复杂度
    - 平均时间复杂度
    - 关注最坏时间复杂度
- 时间复杂度计量规则：
    - 基本操作，只有常数项， 时间复杂度是1

```
a= b
a = b+1;
b = c*2-9
for i:0->100:
    if i%2 ==0
    else
        a = 2
      a+3
      a+6
```

    - 顺序结构，时间复杂度按加法计算 3
    - 循环结构，按乘法计算 n
    - 分支结构， 取最大值 4
    - 最终得到一个方程式 4n+3
    - 然后在做化简，即取方程的阶

找带头大哥

    - 判断一个算法的效率，关注操作数量的最高次项，其余可忽略
    - 没特殊说明， 我们所分析的算法时间复杂度指的是最坏的复杂度
    - 分析案例 “百钱买鸡问题.”

分析

```
import time

def cockOne():
    start_time = time.time()

    for m in range(0,101):
        for n in range(0,101):
            for k in range(0,301):
                if m + n + k == 100 and 5*m + 3*n + k/3 == 100:
                    print("{0} - {1} - {2}".format(m, n, k))
    end_time = time.time()
    print( "CostTimes: {0}".format(end_time - start_time))


def cockThree():
    start_time = time.time()

    for m in range(0,201):
        for n in range(0,334):
            for k in range(0,1001):
                if m + n + k == 1000 and 5*m + 3*n + k/3 == 1000:
```

```
                    print("{0} - {1} - {2}".format(m, n, k))

    end_time = time.time()
    print( "CostTimes: {0}".format(end_time - start_time))


if __name__ == "__main__":
    cockOne()
    cockTwo()
    cockThree()
```