



三者最大实例分析

礼 欣

北京理工大学



分支结构设计举例-三者最大

- 分支结构能改变程序控制流
 - 好消息：我们可以开发更先进算法
 - 坏消息：复杂算法设计很难学
- 以找出三个数字中最大者的程序设计为例。IPO如下：
 - 输入：三个数值
 - 处理：三者最大算法
 - 输出：打印最大值
- 计算机如何确定哪个是用户输入的最大值？





策略1：通盘比较

- 第一种思路是通盘比较，即将每一个值与其他所有值比较以确定最大值

```
if x1 >= x2 and x1 >= x3:  
    max = x1  
elif x2 >= x1 and x2 >= x3:  
    max = x2  
else:  
    max = x3
```

- 存在的问题：
 - 目前只有三个值，比较简单
 - 如果是五个值比较，表达式包含四个and，比较复杂
- 每个表达式结果没有被互相利用，效率低(x1与x2比较了两次)



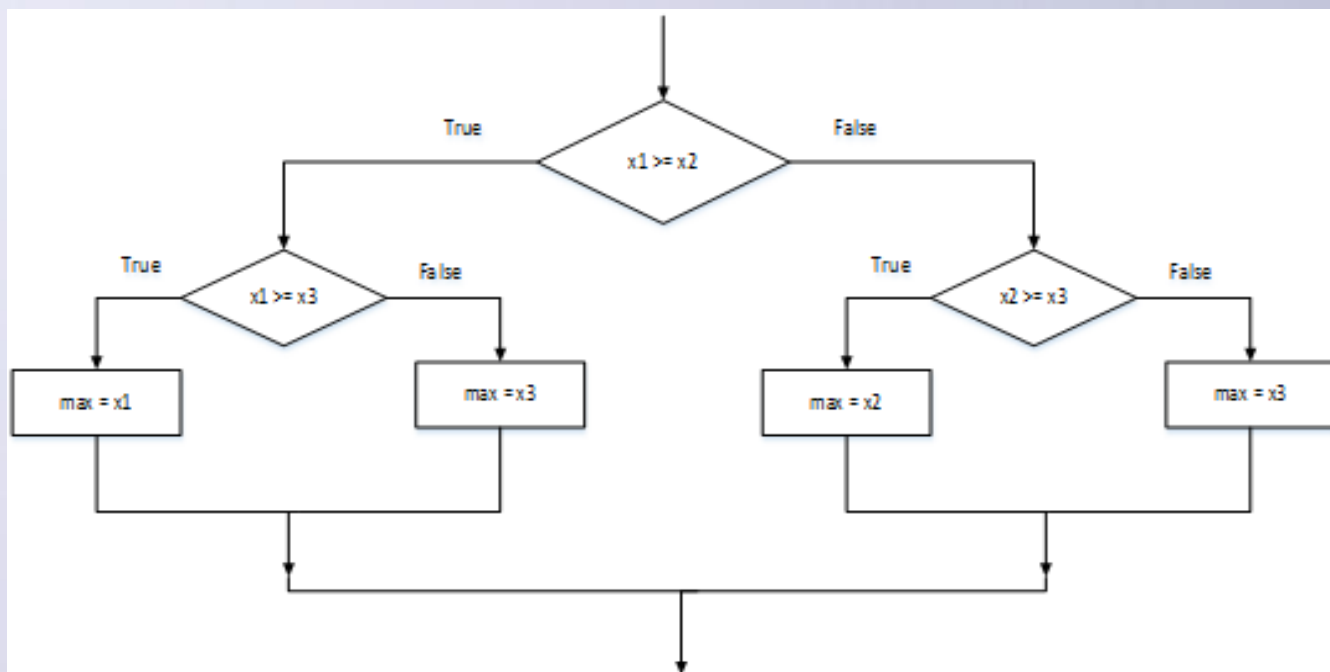
策略2：决策树

- 决策树方法可以避免冗余比较
- 先判断 $x1 > x2$ ，如果成立再判断 $x1 > x3$ ，否则判断 $x2 > x3$
- 虽然效率高，但设计三个以上的方案，复杂性会爆炸性地增长

```
if x1 >= x2:  
    if x1 >= x3:  
        max = x1  
    else:  
        max = x3  
else:  
    if x2 >= x3:  
        max = x2  
    else:  
        max = x3
```

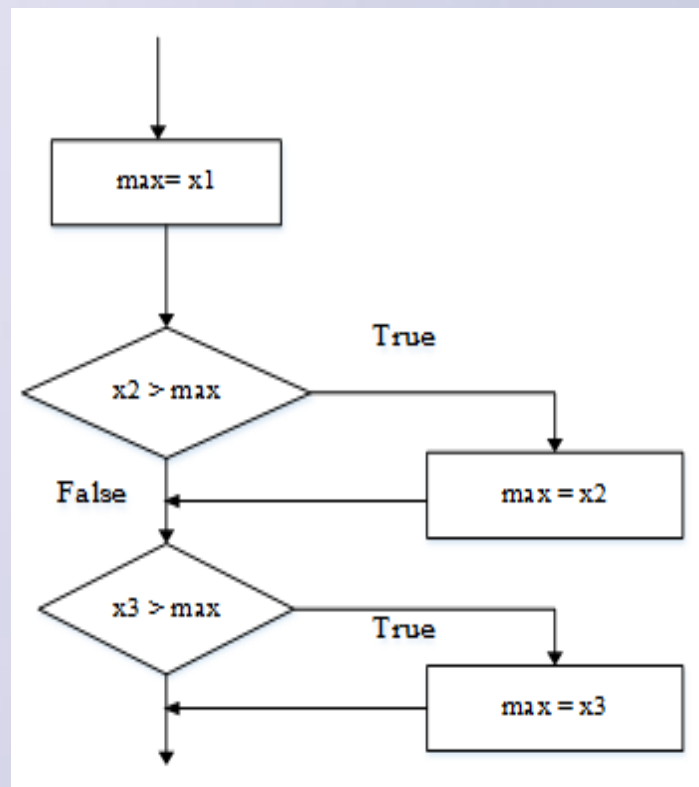
策略2：决策树

- 下图展示了三者最大问题决策树方法的流程图。这种方法的好处是效率高。然而，这种方法的结构比第一个更复杂。
- 读者可以考虑自己设计一个决策树来发现的四个值最大者。（程序将需要在if-else语句中嵌套3层结构和8个赋值语句。）



策略3：顺序处理

- 逐个扫描每个值，保留最大者
- 以max变量保存当前最大值，完成最后一个扫描时，max就是最大值



这种方法处理更大规模问题时非常有效



策略3：顺序处理代码

- 借助循环结构，我们可以实现n个数最大者求值
- 高效、易读

```
max = x1
if x2 > max:
    max = x2
if x3 > max:
    max = x3
```



策略3：N者最大

```
# program: maxn.py
# 寻找一组数中的最大值
def main():
    n = eval(input("How many numbers are there?"))
    # 将第一个值赋值给max
    max = eval(input("Enter a number >> "))
    # 连续与后面n-1值进行比较
    for i in range(n-1):
        x = eval(input("Enter a number >> "))
        if x > max:
            max = x
    print("The largest value is", max)

main()
```




策略4：Python内置函数

- Python终极解决方案 -使用Python内置的max函数 😊

```
def main():  
    x1, x2, x3 = eval(input("Please enter three values: "))  
    print("The largest value is", max(x1, x2, x3))  
main()
```





程序设计思想

- 仔细思考是否还有一个更好的办法？
 - 首先找到一个正确的计算问题，然后使其清晰、简洁、高效、优雅、可扩展。
- 良好的算法和程序应该逻辑清晰，易于阅读和维护。
- 真正专业的程序员，应该学会如何借鉴！

