# 数据结构与算法（Python版）

## 队列的应用：打印任务（下）

陈斌 北京大学 gischen@pku.edu.cn

# 打印任务问题：模拟流程



- 创建打印队列对象
- 时间按照秒的单位流逝

  按照概率生成打印作业，加入打印队列

  如果打印机空闲，且队列不空，则取出队首作业打印，记录此作业等待时间

  如果打印机忙，则按照打印速度进行1秒打印

  如果当前作业打印完成，则打印机进入空闲
- 时间用尽，开始统计平均等待时间

# 打印任务问题：模拟流程



- **作业的等待时间**
  - 生成作业时，记录生成的时间戳
  - 开始打印时，当前时间减去生成时间即可
- **作业的打印时间**
  - 生成作业时，记录作业的页数
  - 开始打印时，页数除以打印速度即可

```
from pythonds.basic.queue import Queue

import random

class Printer:
    def __init__(self, ppm):
        self.pagerate = ppm
        self.currentTask = None
        self.timeRemaining = 0

    def tick(self):
        if self.currentTask != None:
            self.timeRemaining = self.timeRemaining - 1
            if self.timeRemaining <= 0:
                self.currentTask = None

    def busy(self):
        if self.currentTask != None:
            return True
        else:
            return False

    def startNext(self,newtask):
        self.currentTask = newtask
        self.timeRemaining = newtask.getPages() \
                             * 60/self.pagerate
```

打印速度

打印任务

任务倒计时

打印1秒

打印忙?

打印新作业

# 打印任务问题：Python代码2



```
class Task:
    def __init__(self,time):
        self.timestamp = time
        self.pages = random.randrange(1,21)

    def getStamp(self):
        return self.timestamp

    def getPages(self):
        return self.pages

    def waitTime(self, currenttime):
        return currenttime - self.timestamp
def newPrintTask():
    num = random.randrange(1,181)
    if num == 180:
        return True
    else:
        return False
```

生成时间戳

打印页数

等待时间

1/180概率生成作业

```
def simulation(numSeconds, pagesPerMinute):

    labprinter = Printer(pagesPerMinute)
    printQueue = Queue()
    waitingtimes = []

    for currentSecond in range(numSeconds):

      if newPrintTask():
         task = Task(currentSecond)
         printQueue.enqueue(task)

      if (not labprinter.busy()) and \
                (not printQueue.isEmpty()):
        nexttask = printQueue.dequeue()
        waitingtimes.append( \
            nexttask.waitTime(currentSecond))
        labprinter.startNext(nexttask)

      labprinter.tick()

    averageWait=sum(waitingtimes)/len(waitingtimes)
    print("Average Wait %6.2f secs %3d tasks remaining."\
                    %(averageWait,printQueue.size()))
```

模拟

时间流逝

# 打印任务问题：运行和分析1



- 按5PPM、1小时的设定，模拟运行10次

  总平均等待时间93.1秒，最长的平均等待164秒，最短的平均等待26秒

  有3次模拟，还有作业没开始打印

```
>>> for i in range(10):
        simulation(3600,5)


Average Wait  67.00 secs   0 tasks remaining.
Average Wait  26.00 secs   0 tasks remaining.
Average Wait  46.00 secs   2 tasks remaining.
Average Wait 115.00 secs   0 tasks remaining.
Average Wait  53.00 secs   0 tasks remaining.
Average Wait 121.00 secs   0 tasks remaining.
Average Wait 164.00 secs   1 tasks remaining.
Average Wait 136.00 secs   0 tasks remaining.
Average Wait 122.00 secs   2 tasks remaining.
Average Wait  81.00 secs   0 tasks remaining.
```

# 打印任务问题：运行和分析2



## ❖提升打印速度到10PPM、1小时的设定

总平均等待时间12秒，最长的平均等待35秒，最短的平均等待0秒，就是一提交就打印了

而且，所有作业都打印了

```
>>> for i in range(10):
    simulation(3600,10)


Average Wait  35.00 secs   0 tasks remaining.
Average Wait   8.00 secs   0 tasks remaining.
Average Wait  29.00 secs   0 tasks remaining.
Average Wait   0.00 secs   0 tasks remaining.
Average Wait  13.00 secs   0 tasks remaining.
Average Wait   5.00 secs   0 tasks remaining.
Average Wait   0.00 secs   0 tasks remaining.
Average Wait   8.00 secs   0 tasks remaining.
Average Wait  17.00 secs   0 tasks remaining.
Average Wait   5.00 secs   0 tasks remaining.
```

# 打印任务问题：讨论



- **为了对打印模式设置进行决策，我们用模拟程序来评估任务等待时间**
  - 通过两种情况模拟仿真结果的分析，我们认识到
  - 如果有那么多学生要拿着打印好的程序源代码赶去上课的话
  - 那么，必须得牺牲打印质量，提高打印速度。
- **模拟系统对现实的仿真**
  - 在不耗费现实资源的情况下—有时候真实的实验是无法进行的
  - 可以以不同的设定，反复多次模拟
  - 来帮助我们进行决策。

# 打印任务问题：讨论



- 打印任务模拟程序还可以加进不同设定，来进行更丰富的模拟

  学生数量加倍了会怎么样？

  如果在周末，学生不需要赶去上课，能接受更长等待时间，会怎么样？

  如果改用Python编程，源代码大大减少，打印的页数减少了，会怎么样？

# 打印任务问题：讨论



- 更真实的模拟，来源于对问题的更精细建模，以及以真实数据进行设定和运行
- 也可以扩展到其它类似决策支持问题

  如：饭馆的餐桌设置，使得顾客排队时间变短