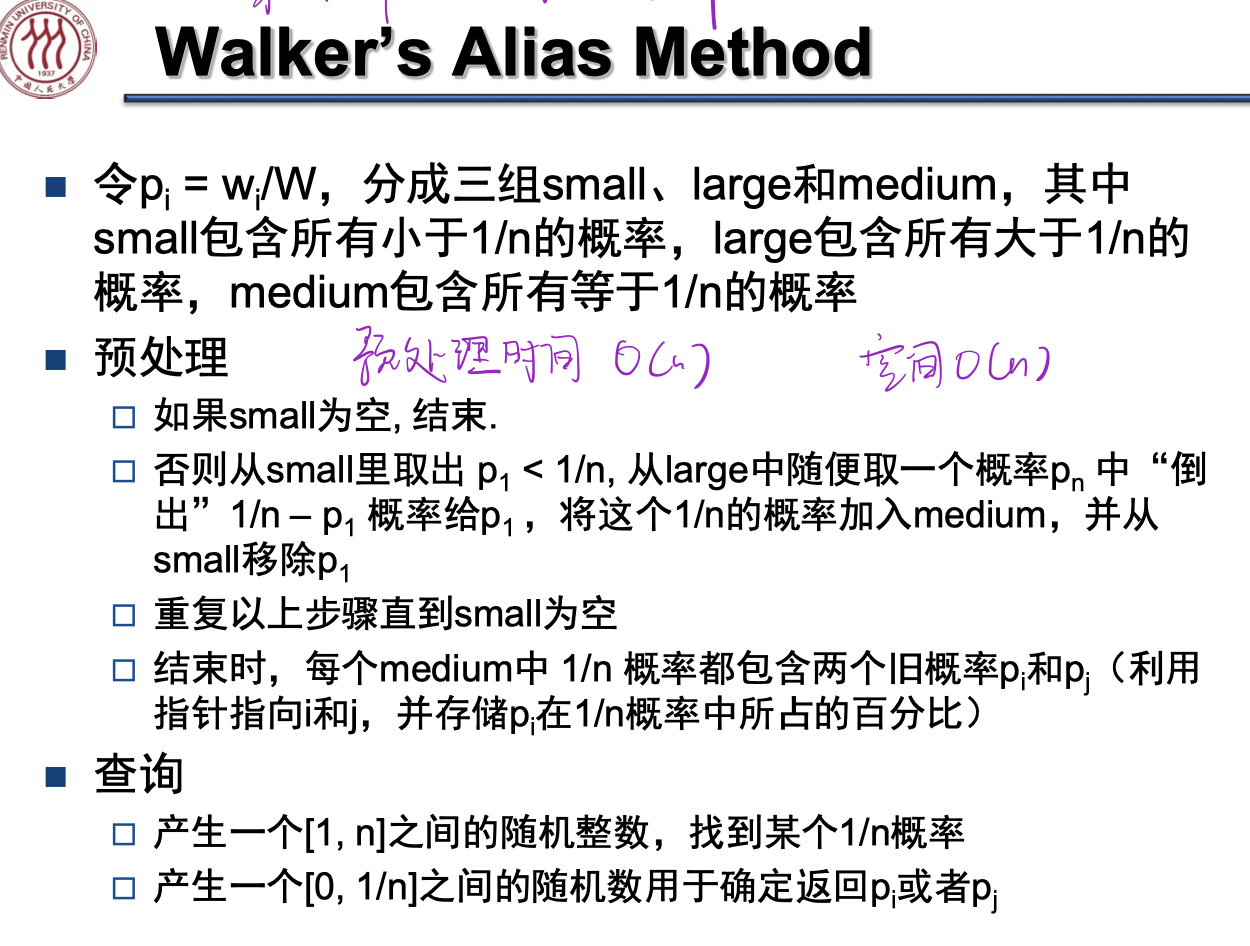
**第7周上机实验报告**

# 一、题目：

实现Walker’s Alias Method

* 预处理时间O(n)
* 空间O(n)
* 查询时间O(1)



# 二、算法思路：

## 核心思路

### 预处理

* 将原数据根据权值大小分组
  + , 将a[i]放进small
  + , 将a[i]放进large
* walker’s alias table：用标号代表第一个元素的index，储存第一个元素的概率和第二个元素的index

vector<pair<**double**, **size\_t**>> walkers\_alias;

* + 由于table一个格子中至多包含2个元素，第二个元素的概率可以由 1-第一个元素概率 得到。
  + 对于格子中只有一个元素的概率，特殊定义一个数字，在初始化时将第二个元素赋值为0xffffffff，如果第二个元素未被修改，意味着格子中只有一个元素。
* 生成walker’s alias table
  + 如果small是空，结束
  + 否则从small中取出一个元素s，再从large中取出一个元素l
    - Walkers\_alias[s].second设置为l
    - 更新l的概率
      * 根据新值将l放进large或者small
    - 进行下一次循环

# 三、程序设计框架：

核心类的名称及功能:

Class discrete\_random\_variable

* 储存数据
  + values： 记录每一个元素输入时的index以及该元素的值
  + walkers\_alias: 储存walkers\_alias\_method生成的表
    - pair中first为第一个元素的概率，second为第二个元素的index
      * 如果只有一个元素，second为0xffffffff
* 随机采样
  + 使用random\_device随机数生成引擎产生随机数
  + 再使用梅森旋转算法利用该随机数产生特定分布
  + Int型(0, n-1)的分布生成采样的index
  + double型(0, 1)的分布生成概率p，如果index对应的第一个元素的概率不小于p，则选择第一个元素，否则选择第二个
* discrete\_random\_variable(**const** vector<**int**> &vals, **const** vector<**double**> &probs)
  + 进行初始化和错误判断
* **int** give\_choice()
  + 随机采样的接口
* vector<pair<**double**, **size\_t**>> generate\_walkers\_alias(**const** vector<**double**> &probs)
  + 生成walkers\_alias\_table，是“算法思路”部分的实现

1. **class** discrete\_random\_variable
2. {
3. **private**:
4. vector<**int**> values;
5. vector<pair<**double**, **size\_t**>> walkers\_alias;
6. random\_device rd\_;                                     //随机数引擎
7. mt19937 gen\_{rd\_()};                                   //产生特定分布
8. uniform\_real\_distribution<**double**> real\_dis\_{0.0, 1.0}; //real （0，1）上的均匀分布
9. uniform\_int\_distribution<**size\_t**> int\_dis\_;             //均匀分布，在知道values的size的时候才能够初始化
11. **public**:
12. discrete\_random\_variable(**const** vector<**int**> &vals, **const** vector<**double**> &probs) : values(vals), walkers\_alias(generate\_walkers\_alias(probs)), int\_dis\_(0, probs.size() - 1)
13. {
14. assert(vals.size() == probs.size());
15. **const** **double** sum = accumulate(probs.begin(), probs.end(), 0.0);
16. assert(fabs(1.00 - sum) < 0.00000000000001);
17. }
19. **int** give\_choice()
20. {
21. **const** **size\_t** idx = int\_dis\_(gen\_);
22. **const** **double** proportion = real\_dis\_(gen\_);
23. **if** (proportion > walkers\_alias[idx].first && walkers\_alias[idx].second != numeric\_limits<**size\_t**>::max())
24. {
25. **return** values[walkers\_alias[idx].second];
26. }
27. **else**
28. {
29. **return** values[idx];
30. }
31. }
33. **private**:
34. vector<pair<**double**, **size\_t**>> generate\_walkers\_alias(**const** vector<**double**> &probs)
35. {
36. **size\_t** len = probs.size();
37. vector<pair<**double**, **size\_t**>> walkers\_alias\_table(len, {0.00, numeric\_limits<**size\_t**>::max()});
38. queue<**size\_t**> small, large;
40. **for** (**size\_t** i = 0; i < len; i++)
41. {
42. walkers\_alias\_table[i].first = probs[i] \* len;
43. **if** (walkers\_alias\_table[i].first > 1.00)
44. {
45. large.push(i);
46. }
47. **else**
48. {
49. small.push(i);
50. }
51. }
53. **while** (!small.empty())
54. {
55. **size\_t** s = small.front(), l = large.front();
56. small.pop();
57. large.pop();
58. walkers\_alias\_table[s].second = l;
59. walkers\_alias\_table[l].first = walkers\_alias\_table[l].first - (1.0 - walkers\_alias\_table[s].first);
60. //更新large剩下的部分
61. **if** (walkers\_alias\_table[l].first < 1.0)
62. {
63. small.push(l);
64. }
65. **else**
66. {
67. large.push(l);
68. }
69. }
70. **return** walkers\_alias\_table;
71. }
72. };

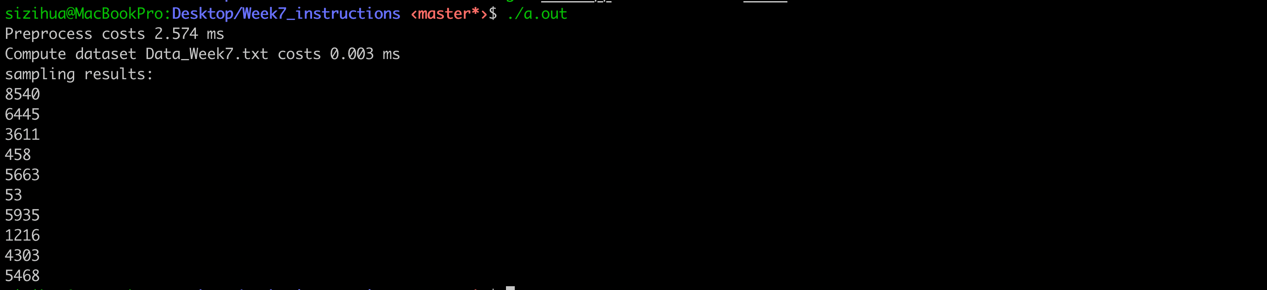
## 输入输出的格式

程序第92行中，file\_name设置为相对路径或者绝对路径均可。

1. string file\_name = "test.txt";

# 四、实验结果说明：

## 实验结果

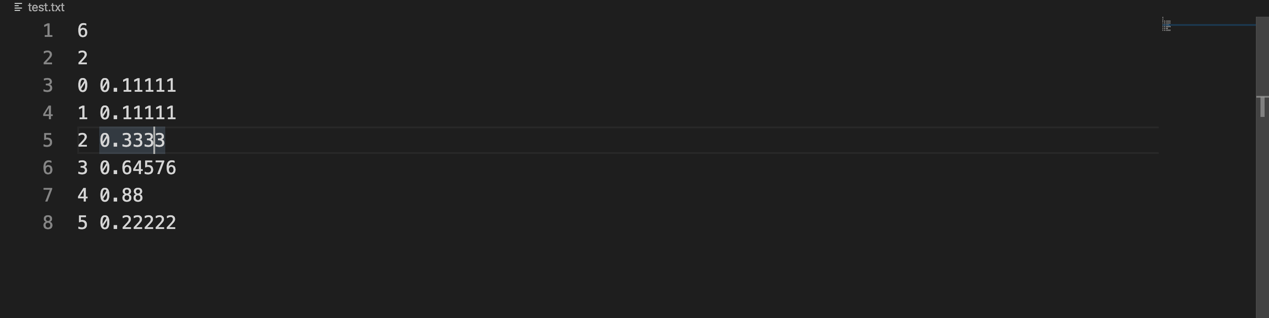


图表 a:运行提供的数据集

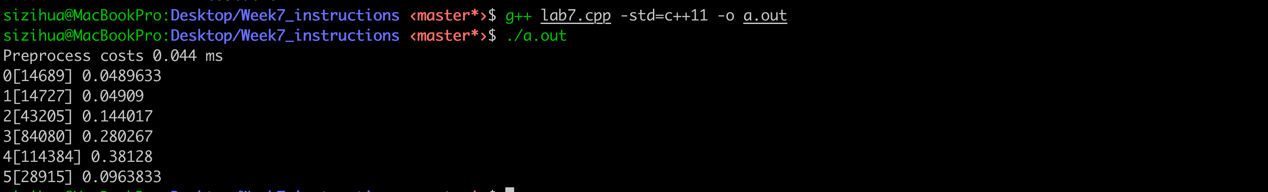
鉴于提供的数据集无法判断出程序的正确性，我写了一个简单的测试集（如图b所示）来对正确性进行测试。Main函数中注释掉的部分就是测试的部分。最终测试结果如图c所示。

验证思路：

* + 给定6个元素和其权重
  + 抽样300,000次
  + 计算样本中各个元素的占比是否和权重占比相符



图表 b：简单的正确性测试集



图表 c

## 实验结果的分析

正确性分析：

* 输出结果中a[m] p
  + a:代表该元素的index
  + m:代表元素被抽中的次数
  + p:代表抽中次数在总抽样次数中的占比
* 从以上结果和测试集中数据的权重对比可以看出，在大量地抽样下，元素i被抽中的概率近似等于其权重在总权重中的比例。故本实验的程序是正确的。

性能分析：

* 从结果上看，程序满足预处理时间复杂度O(n)，抽样时间O(1)的要求。