c. 如果我们愿意牺牲散列表 ADT 的性能，那么可以在(b)部分使程序加速：例如，如果我们刚刚计算出“excel”的散列函数，那么就不必再从头开始计算“excels”的散列函数。调整散列函数使得它能够利用前面的计算。

d. 在第 2 章我们建议使用折半查找。把使用前缀的想法结合到你的折半查找算法中。修改工作应该简单。哪个算法更快?

5.16 在某些假设下，向带有二次聚集的散列表进行的一次插入操作的期望代价由 $1/(1-\lambda)-\lambda-\ln(1-\lambda)$ 给出。不过，这个公式对于平方探测并不精确。我们假设它是准确的，确定：

a. 一次不成功查找的期望代价。

b. 一次成功查找的期望代价。

5.17 实现支持 put 和 get 操作的泛型 Map。该实现方法将存储(关键字，定义)对的散列表。图 5-27 提供 Map 的说明(去掉某些细节)。

```
1   class Map<KeyType,ValueType>
2   {
3       public Map( )
4
5       public void put( KeyType key, ValueType val )
6       public ValueType get( KeyType key )
7       public boolean isEmpty( )
8       public void makeEmpty( )
9
10      private QuadraticProbingHashTable<Entry<KeyType,ValueType>> items;
11
12      private static class Entry<KeyType,ValueType>
13      {
14          KeyType key;
15          ValueType value;
16          // Appropriate Constructors, etc.
17      }
18  }
```

图 5-27　练习 5.17 的 Map 架构

5.18 通过使用散列表实现一个拼写检查程序。设词典来自两个来源：一本现有的大词典以及包含一本个人词典的第二个文件。输出所有错拼的单词和这些单词出现的行号。再有，对于每个错拼的单词，列出应用下列任一种法则在词典中能够得到的任意的单词：

a. 添加一个字符。

b. 去掉一个字符。

c. 交换两个相邻的字符。

5.19 指出将关键字 10 111 101、00 000 010、10 011 011、10 111 110、01 111 111、01 010 001、10 010 110、00 001 011、11 001 111、10 011 110、11 011 011、00 101 011、01 100 001、11 110 000、01 101 111 插入到一个空的初始可扩散列数据结构中的结果，其中 $M=4$。

5.20 编写一个程序实现可扩散列法。如果散列表小到足可装入内存，那么它的性能与分离链接法和开放定址散列法相比如何?