表驱动法

表驱动法

- 表驱动法是一种编程模式（scheme）
  - 从表里面查找信息而不使用逻辑语句（if和case）
- 表驱动法适用于复杂的逻辑
- 表驱动法的另一个好处是可以将复杂逻辑从代码中独立出来，以便于单独维护

表驱动示例

Java示例：使用复杂的逻辑对字符分类

```
if ( ( ( 'a' <= inputChar ) && ( inputChar <= 'z' ) ) ||
   ( ( 'A' <= inputChar ) && ( inputChar <= 'Z' ) ) ) {
   charType = CharacterType.Letter;
}else if ( ( inputChar == ' ' ) || ( inputChar == ',' ) ||
   ( inputChar == '.' ) || ( inputChar == '!' ) || ( inputChar == '(' ) ||
   ( inputChar == ')' ) || ( inputChar == ':' ) || ( inputChar == ';' ) ||
   ( inputChar == '?' ) || ( inputChar == '-' ) ) {
   charType = CharacterType.Punctuation;
}else if ( ( '0' <= inputChar ) && ( inputChar <= '9' ) ) {
   charType = CharacterType.Digit;
}
```

构造一个查询表

Java示例：使用查询表对字符分类

```
chartype = charTypeTable[ inputChar ];
```

把每一个字符的类型保存在一个用字符编码访问的数组里

使用表驱动法的两个问题

- 在表里存放什么信息
  - 主要存放的是数据，但在一些特殊情况下也存放动作
- 如何快速从表中查询条目
  - 直接访问（Direct access）
  - 索引访问（Indexed access）
  - 阶梯访问（Stair-step access）

直接访问表

- 所谓“直接访问”是指通过索引值（如下标）可以直接从表中找到对应的条目

VB示例：确定各月天数的笨拙做法

```
If ( month = 1 ) Then
   days = 31
ElseIf ( month = 2 ) Then
   days = 28
ElseIf ( month = 3 ) Then
   days = 31
…
ElseIf ( month = 11 ) Then
   days = 30
ElseIf ( month = 12 ) Then
   days = 31 End If
```

你需要首先创建出这张表用来存放各个月份的天数

VB示例：确定各月天数的优雅做法

```
Dim daysPerMonth() As Integer = _
   { 31, 28, 31, 30, 31, 30, 31, 31, 30,
31, 30, 31 }
```

```
days = daysPerMonth( month-1 )
```

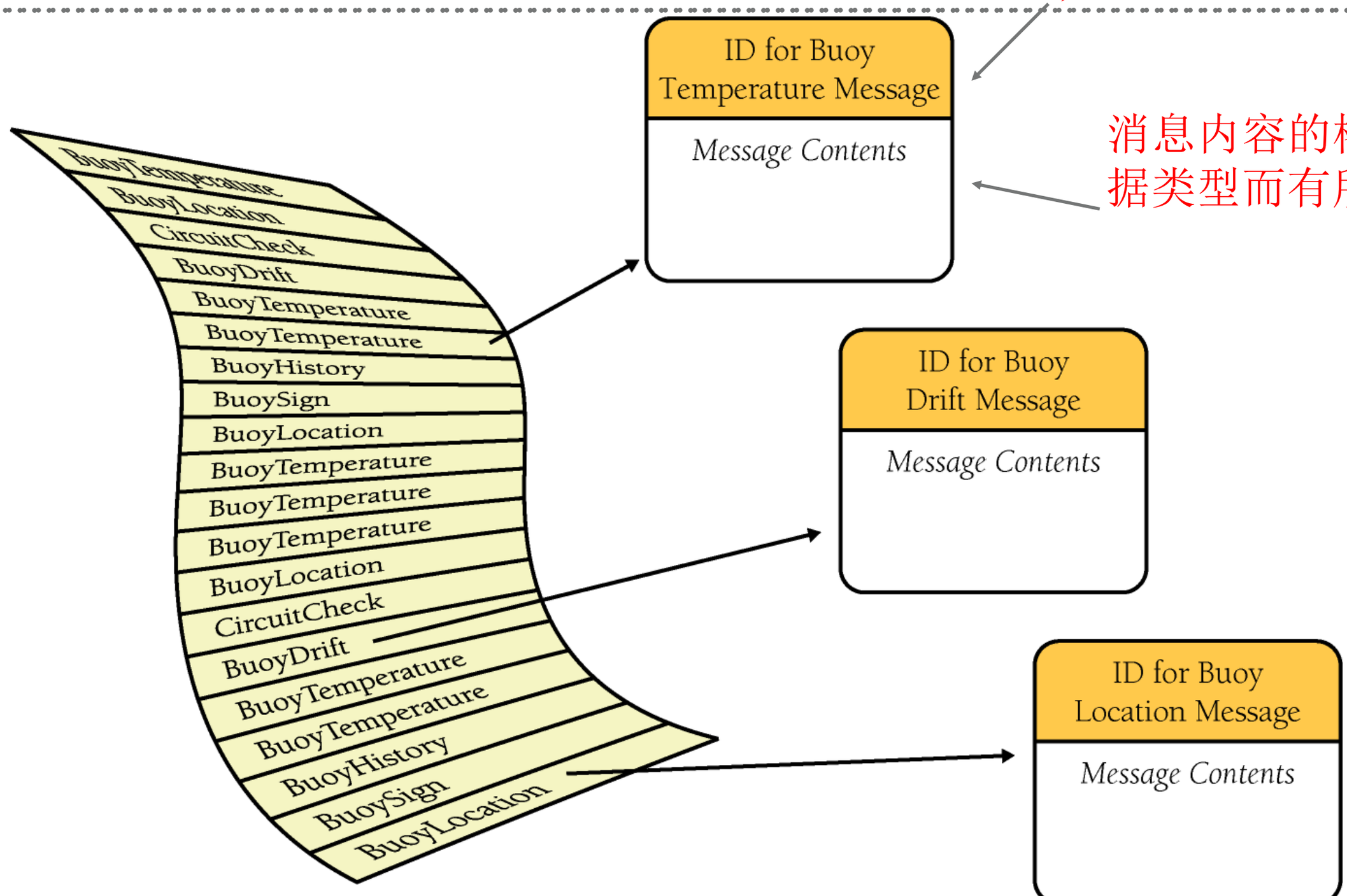
例子:灵活的消息格式

- 假设你编写一个子程序，打印存储在一份文件中的消息
  - 通常该文件中会存储大约500 条消息，而每份文件中会存有大约20 种不同的消息。这些消息源自于一些浮标(Buoy)，提供有关水温、浮标位置等信息。
  - 每一条消息都有若干字段，并且每条消息都有一个消息头，其中有一个ID，告诉你该消息属于这20 多种消息中的哪一种。
  - 这些消息的格式并不是固定不变的

消息的存储方式

消息头（**ID**确定了该消息所属的类型）

消息内容的格式根据类型而有所不同



消息的格式细节

ID for Buoy Temperature Message
Average Temperature (floating point)
Temperature Range (floating point)
Number of Samples (integer)
Location (character string)
Time of Measurement (time of day)

ID for Buoy Drift Message
Change in Latitude (floating point)
Change in Longitude (floating point)
Time of Measurement (time of day)

ID for Buoy Location Message
Latitude (floating point)
Longitude (floating point)
Depth (integer)
Time of Measurement (time of day)

基于逻辑的方法

- 读取每一条消息，检查其ID，然后调用一个用来阅读、解释以及打印一种消息的子程序
  - 消息阅读子程序包含一个循环，用来读入消息、解释其ID，以及根据该ID调用20 个子程序中的某一个
  - 如果你有20 种消息，那么就要有20 个子程序
  - 每次有任何一种消息的格式变了，你就不得不修改负责处理该消息的子程序或者类的逻辑

基于逻辑方法所用的伪代码

```
While more messages to read
   Read a message header
   Decode the message ID from the message header

   If the message header is type 1 then
      Print a type 1 message
Else if the message header is type 2 then
   Print a type 2 message
...
Else if the message header is type 19 then
   Print a type 19 message
Else if the message header is type 20 then
   Print a type 20 message
```

画向对象的方法

但是基本结构还是同样复杂

- 问题的逻辑可以隐藏在对象继承结构里

```
While more messages to read
   Read a message header
   Decode the message ID from the message header
   If the message header is type 1 then
      Instantiate a type 1 message object
   Else if the message header is type 2 then
      Instantiate a type 2 message object
   ...
   Else if the message header is type 19 then
      Instantiate a type 19 message object
   Else if the message header is type 20 then
      Instantiate a type 20 message object
   End if
End While
```

读取和打印浮标温度消息子程序

Print "Buoy Temperature Message“

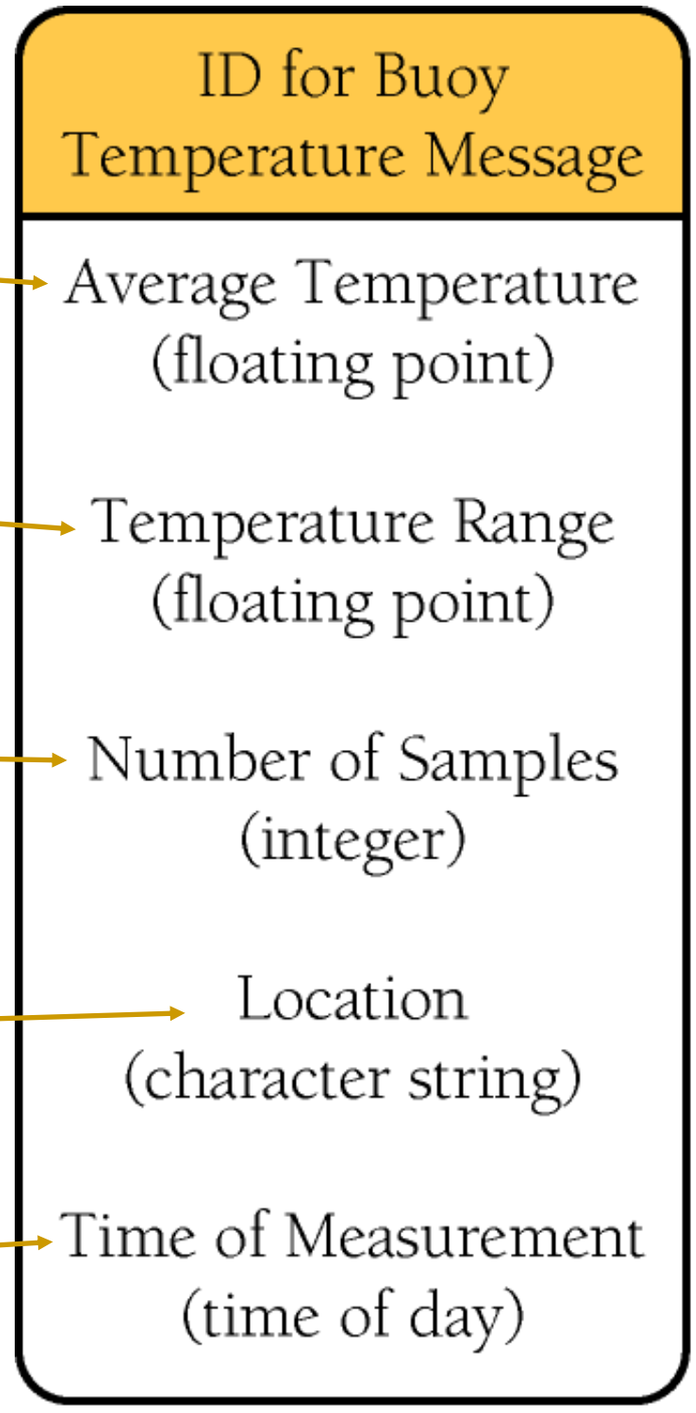
Read a floating-point value
Print "Average Temperature"
Print the floating-point value

Read a floating-point value
Print "Temperature Range"
Print the floating-point value

Read an integer value
Print "Number of Samples"
Print the integer value

Read a character string
Print "Location"
Print the character string

Read a time of day Print
Print the time of day



表驱动法

- 消息阅读子程序由一个循环组成，该循环负责读入每一个消息头，对其ID解码，在Message 数组中查询其消息描述，然后每次都调用同一个子程序来解释该消息
- 只需要用一张表来描述每种消息的格式，而不用再把它们硬编码进程序逻辑里

定义所有可能的字段类型

- 在定义消息表项之前先定义消息中可能出现的所有字段类型

C++示例：定义消息数据类型

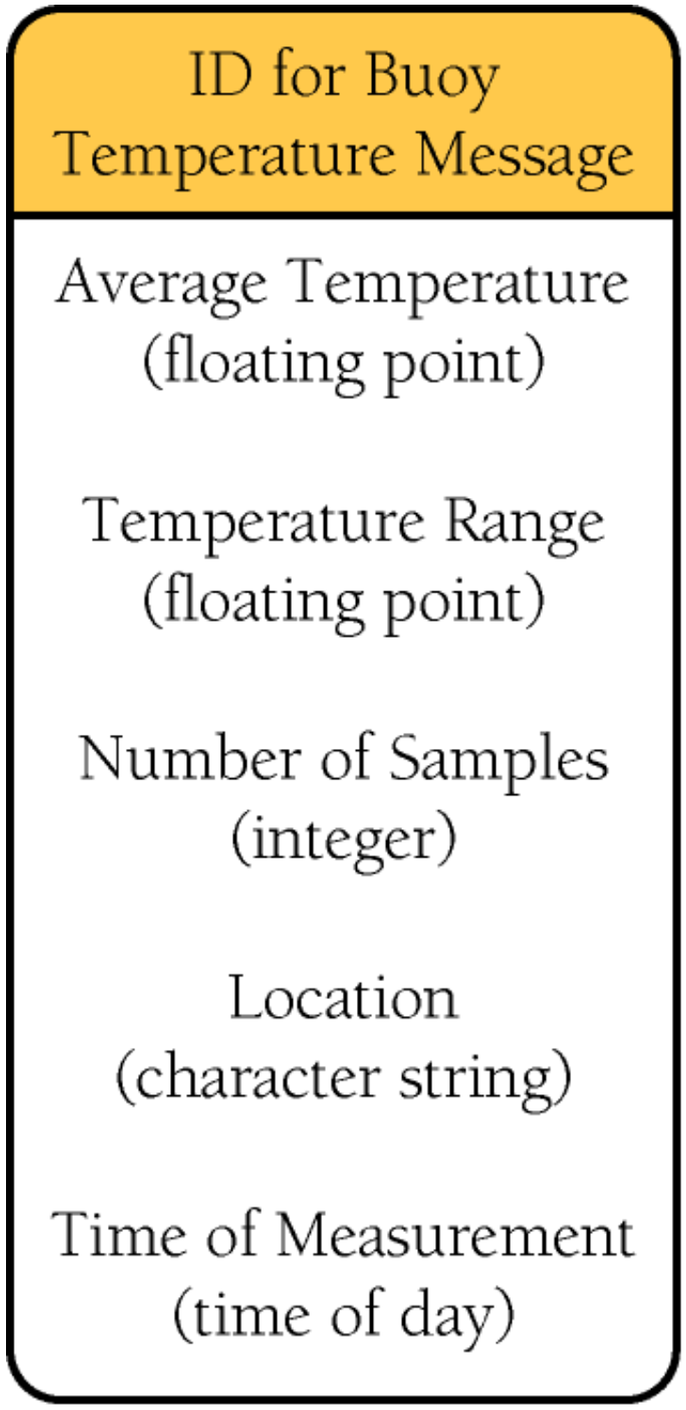
```
enum FieldType {
   FieldType_FloatingPoint,
   FieldType_Integer,
   FieldType_String,
   FieldType_TimeOfDay,
   FieldType_Boolean,
   FieldType_BitField,
   FieldType_Last = FieldType_BitField
};
```

消息的表记录—浮标温度消息

示例：定义浮标温度消息的表记录项

```
Message Begin
   NumFields 5
   MessageName "Buoy Temperature Message"
   Field 1, FloatingPoint, "Average Temperature"
   Field 2, FloatingPoint, "Temperature Range"
   Field 3, Integer, "Number of Samples"
   Field 4, String, "Location"
   Field 5, TimeOfDay, "Time of Measurement"
Message End
```

有多少种类型的消息，就需要定义多少个表项



表驱动法代码

- 表驱动法中最上层循环的伪代码
  - 这部分的代码与前面的方法差别不大

```
While more messages to read
   Read a message header
   Decode the message ID from the message header
   Look up the message description in the message-description table
   Read the message fields and print them based on the message
description
End While
```

消息打印子程序

```
While more fields to print
   Get the field type from the message description
   case ( field type )
      of ( floating point )
         read a floating-point value
         print the field label
         print the floating-point value
      of ( integer )
         read an integer value
         print the field label
         print the integer value
      of ( character string )
         read a character string
         print the field label
         print the character string
      …
   End Case
End While
```

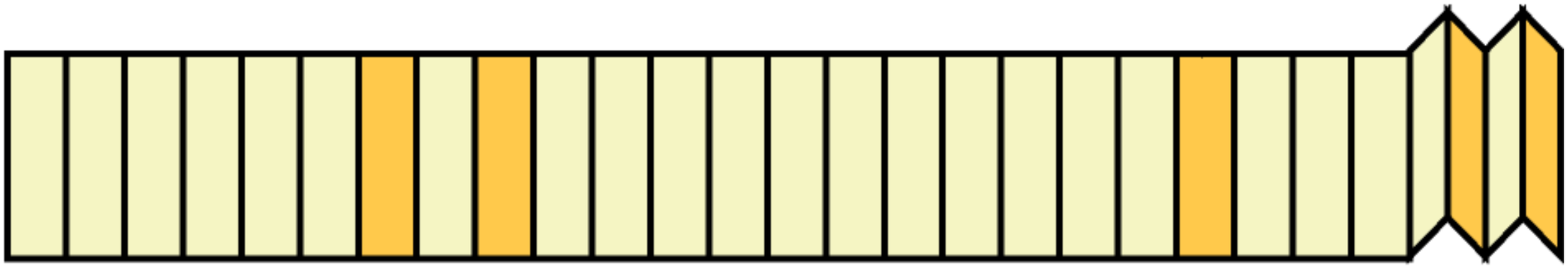
用一个打印子程序取代了**19**种消息的打印子程序。

索引访问表

- 当无法直接从表中查询需要的条目时，就需要借助其他办法先获取表键值
- 索引访问的方法
  - 就是先用一个基本类型的数据从一张索引表中查出一个键值，然后再用这一键值查出你感兴趣的主数据
- 索引表是一种间接访问的技术

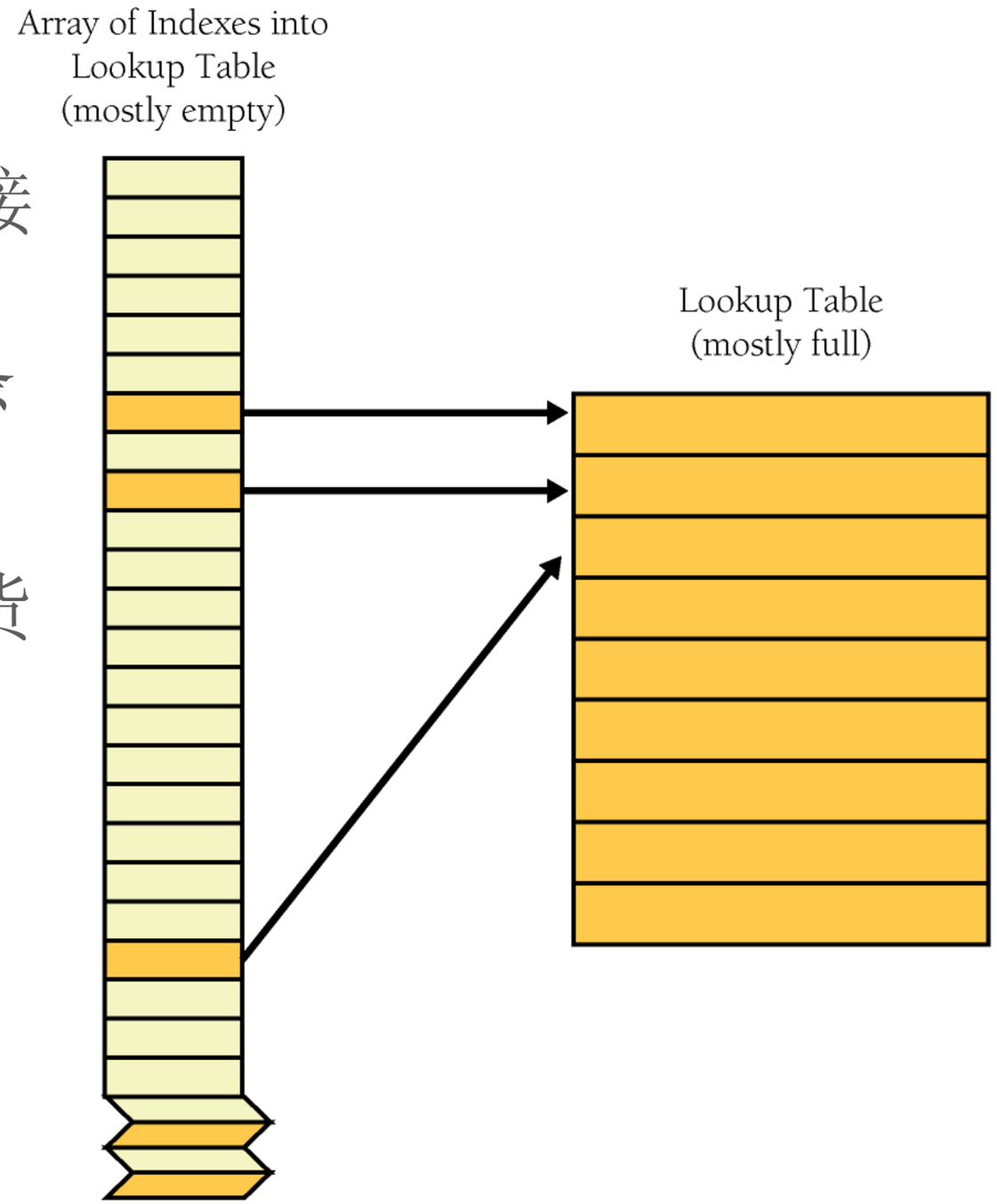
示例

- 假设你经营着一家商店，有大约100 种商品。再假设每种商品都有一个4 位数字的物品编号， 其范围是0000 到9999
  - 如果你想用这个编号作为键值直接查询一张描述商品信息的表，那么就要生成一个具有10000 条记录的访问表



示例

- 利用索引表
  - 如果用这个编号作为键值直接查询一张描述商品信息的表，那么就生成一个具有10000 条记录的索引数组(从0到9999)
  - 该数组中除了与你商店中的货物的标志相对应的100条记录以外，其余记录都是空的

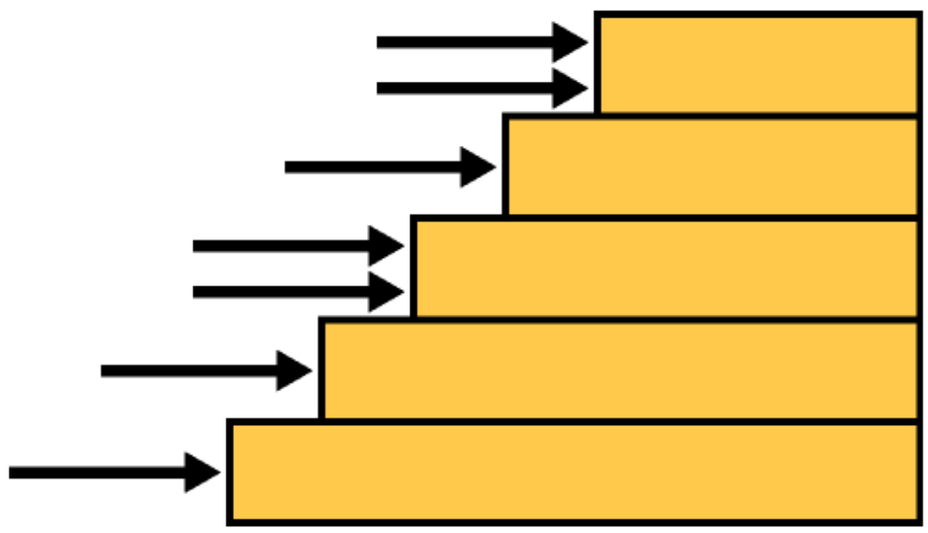


索引访问技术的优点

- 索引访问技术的主要优点：
  - 如果主查询表中的每一条记录都很大，那么索引数组就可以节省很多空间
    - 一般而言索引表中的每条记录需要占用2 ~4字节
  - 即使你用了索引以后没有节省内存空间， 操作位于索引中的记录有时也要比操作位于主表中的记录更方便更廉价
  - 编写到表里面的数据比嵌入代码中的数据更容易维护

阶梯访问表

- 阶梯访问方法不像索引结构那样直接， 但是它要比索引访问方法节省空间
- 阶梯结构的基本想法
  - 通过确定每项命中的阶梯层次确定其归类



示例

- 如果你正在开发一个等级评定的应用程序，按照如下等级区间对分数定级

分值区间	等级
≥90.0%	A
<90.0%	B
<75.0%	C
<65.0%	D
<50.0%	F

- 你不能用简单的数据转换函数来把表键值转换为A 至F 字母所代表的等级
- 用索引也不合适，因为这里用的是浮点数

使用阶梯方法

- 把每一区间的上限写入一张表里， 然后写一个循环，按照各区间的上限来检查分数
  - 当分数第一次超过某个区间的上限时， 你就知道相应的等级了
  - 区间表：{ 50.0, 65.0, 75.0, 90.0, 100.0 }

Visual Basic示例：阶梯表查询

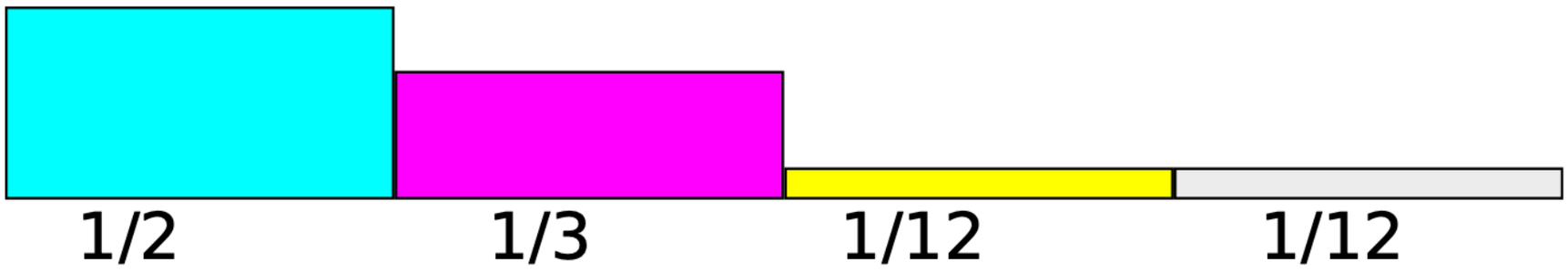
```
' set up data for grading table
Dim rangeLimit() As Double = { 50.0, 65.0, 75.0, 90.0, 100.0 }
Dim grade() As String = { "F", "D", "C", "B", "A" }
maxGradeLevel = grade.Length - 1
...
' assign a grade to a student based on the student's score
gradeLevel = 0
studentGrade = "A"
While (  gradeLevel < maxGradeLevel )
   If ( studentScore < rangeLimit( gradeLevel ) ) Then
      studentGrade = grade( gradeLevel )
   End If
   gradeLevel = gradeLevel + 1
Wend
```

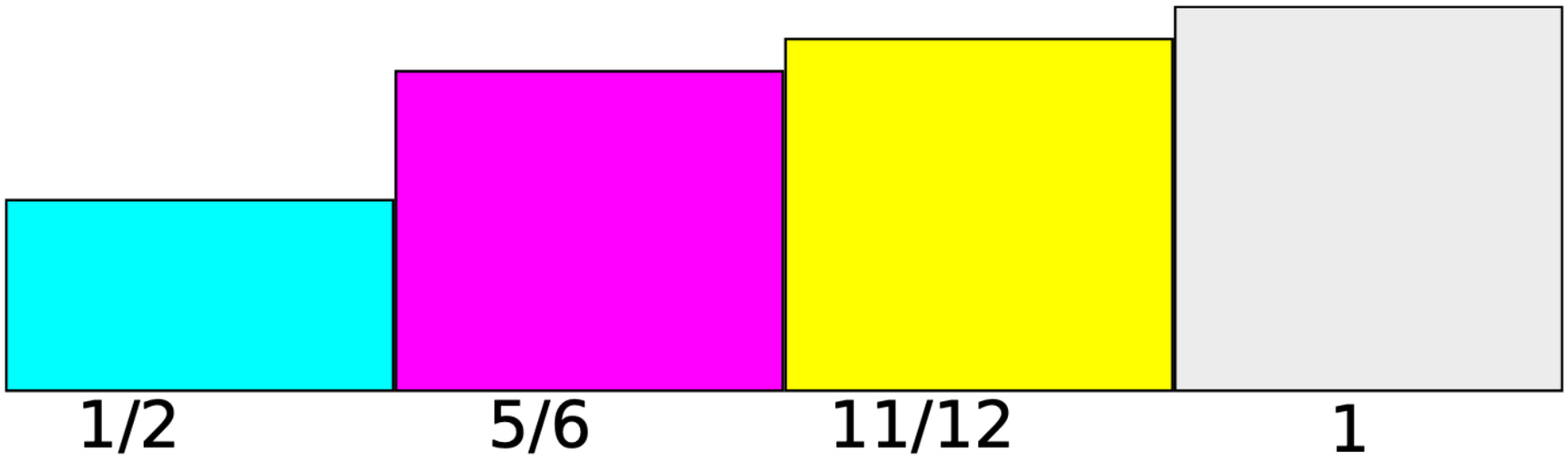
注意事项

- 注意端点

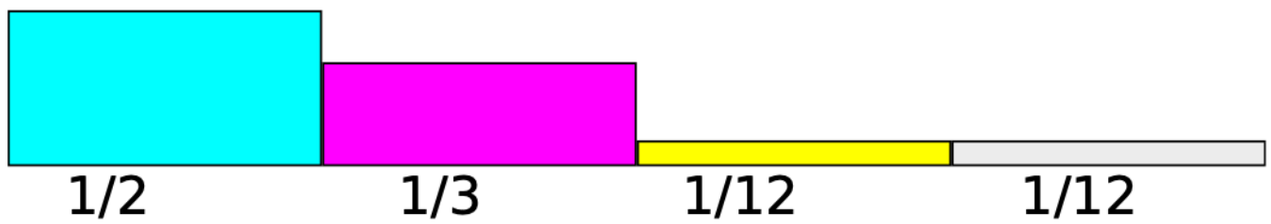
- 二分查找取代顺序查找

示例：CDF



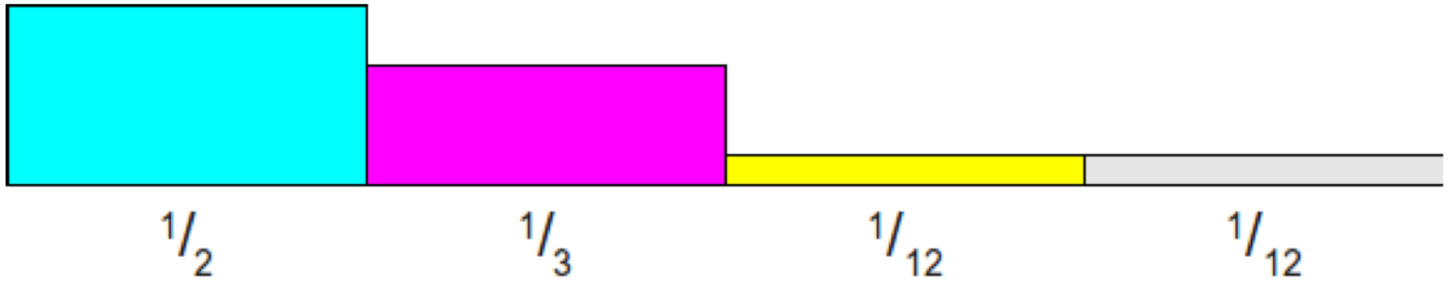


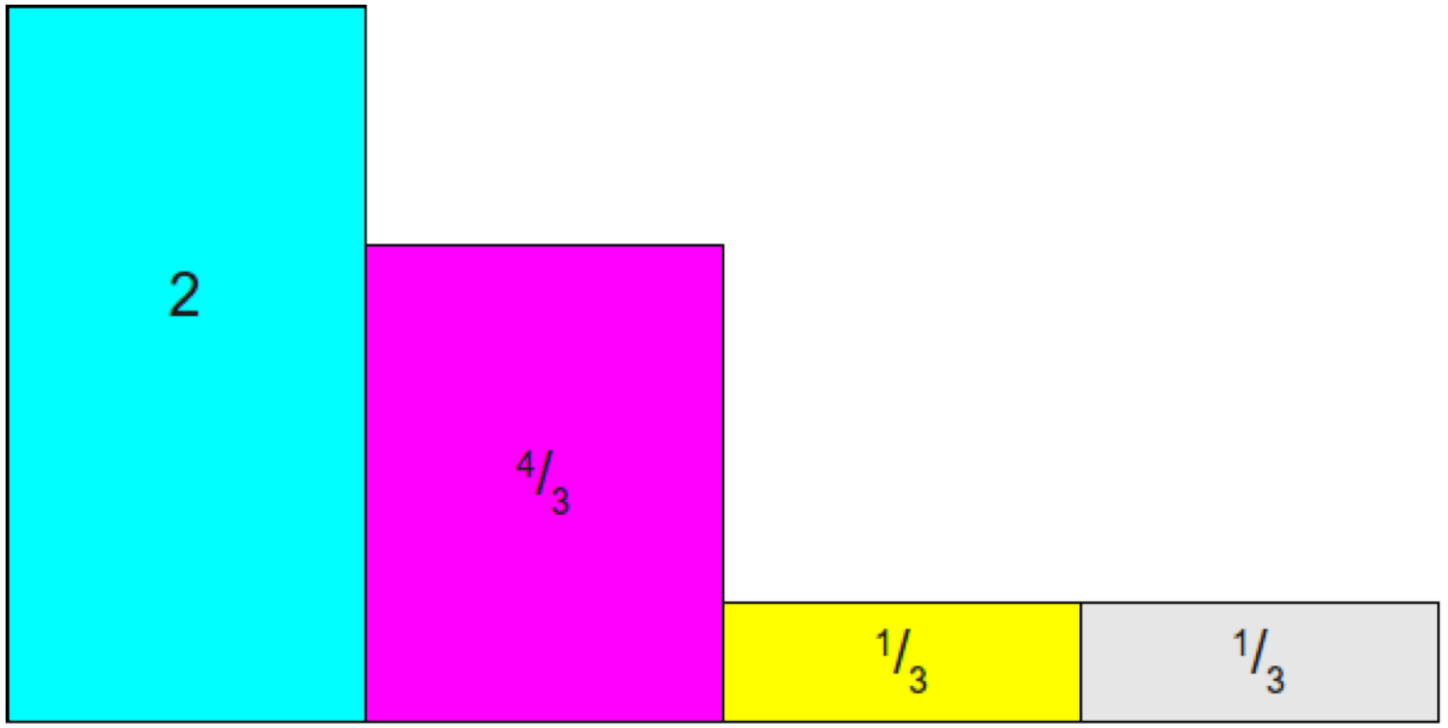
示例：CDF



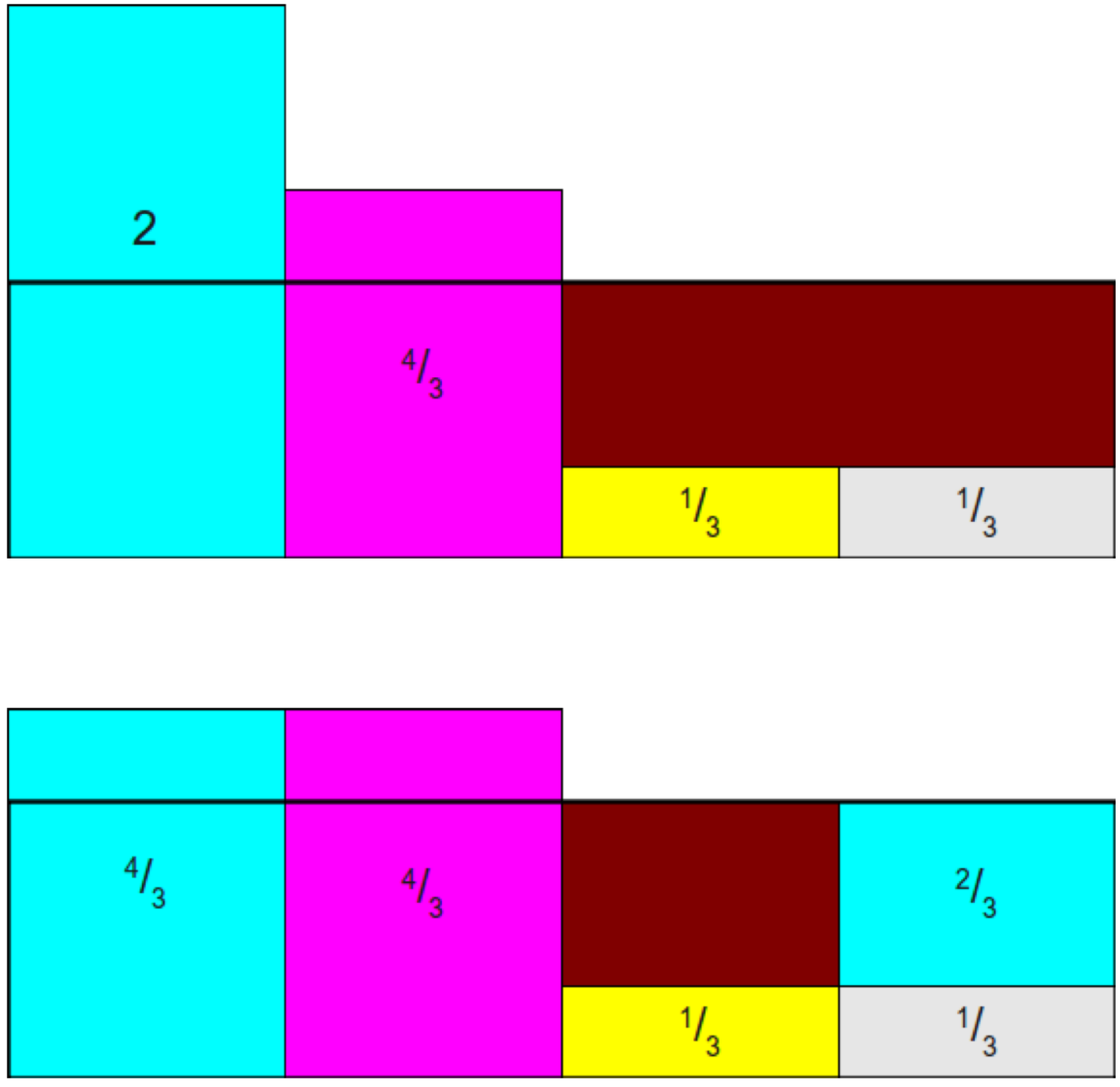
1. Generate a random number A between 0 and 1
2. Compare A with CDF(i) until $A <$ CDF(i) and select i
3. Need n comparisons in the worst case

改进：别名方法

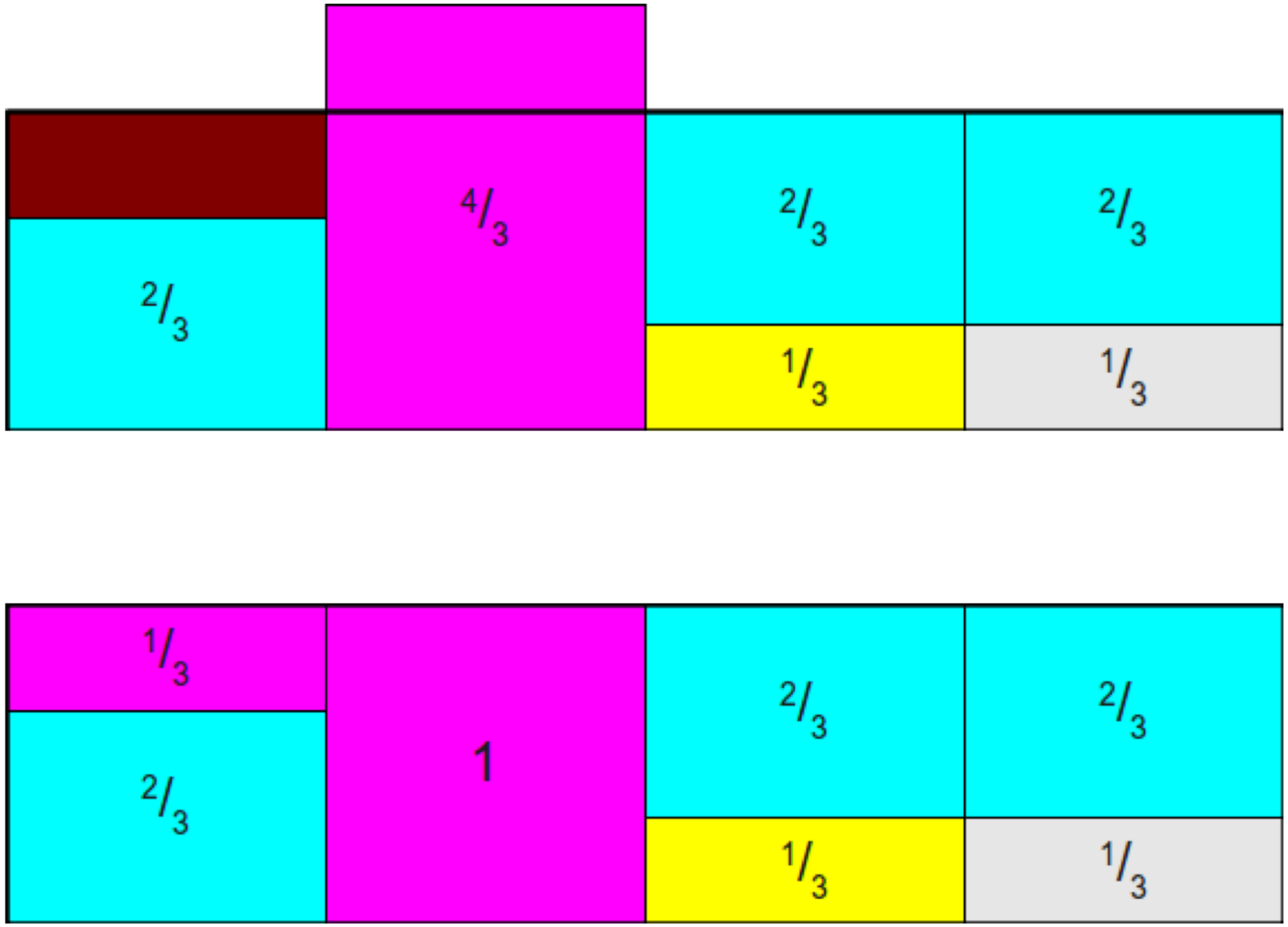




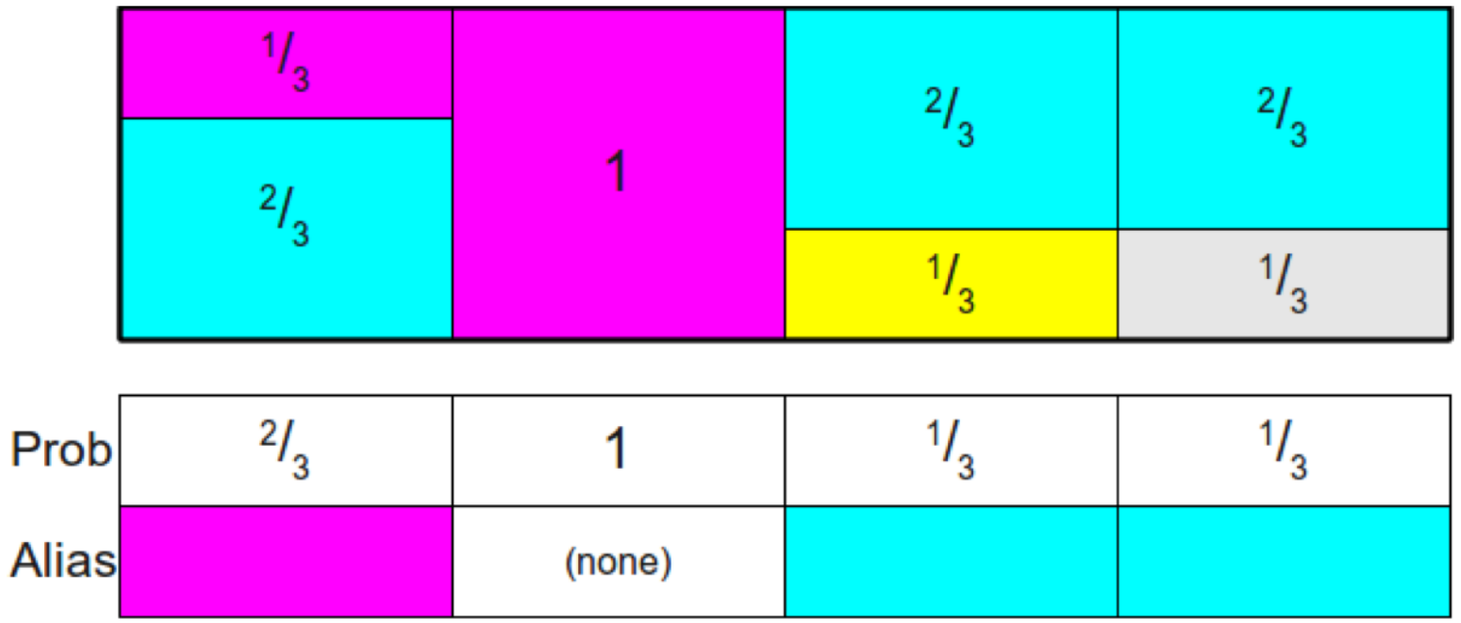
改进：别名方法



改进：别名方法



改进：别名方法



1. Generate a random number A from $\{1, 2, 3, 4\}$
2. Generate a random number B between 0 and 1
3. Compare B with Prob$\{A\}$
4. If $B >$ Prob$\{A\}$, then select Alias$\{A\}$; otherwise, select A.