08讲组合：如何让计算机安排世界杯的赛程

你好，我是黄申。

2018年足球世界杯结束有半年了，当时激烈的赛况你现在还记忆犹新吧？你知道这场足球盛宴的比赛日程是怎么安排的吗？如果现在你是组委会，你会怎么安排比赛日程呢？我们可以用上一节的排列思想，让全部的32支入围球队都和其他球队进行一次主客场的比赛。

自己不可能和自己比赛，因此在这种不可重复的排列中，主场球队有32种选择，而客场球队有31种选择。那么一共要进行多少场比赛呢？很简单，就是32x31=992场！这也太夸张了吧？一天看2场，也要1年多才能看完！即使球迷开心了，可是每队球员要踢主客场共62场，早已累趴下了。

好吧，既然这样，我们是否可以取消主客场制，让任意两个球队之间只要踢1场就好啦？取消主客场，这就意味着原来两队之间的比赛由2场降为1场，那么所有比赛场次就是992/2=496场。还是很多，对吧？

是的，这就是为什么要将所有32支队伍分成8个小组先进行小组赛的原因。一旦分成小组，每个小组的赛事就是(4x3)/2=6场。所有小组赛就是6x8=48场。

再加上在16强阶段开始采取淘汰制，两两淘汰，所以需要8+4+2+2=16场淘汰赛（最后一次加2是因为还有3、4名的决赛），那么整个世界杯决赛阶段就是48+16=64场比赛。

当然，说了这么多，你可能会好奇，这两两配对比赛的场次，我是如何计算出来的？让我引出今天的概念，**组合**（Combination）。

组合可以说是排列的兄弟，两者类似但又有所不同，这两者的区别，不知道你还记得不，上学的时候，老师肯定说过不止一次，那就是，组合是不考虑每个元素出现的顺序的。

从定义上来说，组合是指，从n个不同元素中取出m（1≤m≤n）个不同的元素。例如，我们前面说到的世界杯足球赛的例子，从32支球队里找出任意2支球队进行比赛，就是从32个元素中取出2个元素的组合。如果上一讲中，田忌赛马的规则改一下，

改为从10匹马里挑出3匹比赛，但是并不关心这3匹马的出战顺序，那么也是一个组合的问题。

对于所有m取值的组合之全集合，我们可以叫作**全组合**（All Combination）。例如对于集合{1, 2, 3}而言，全组合就是{空集, {1}, {2}, {3}, {1, 2}, {1,3} {2, 3}, {1, 2, 3}}。

如果我们安排足球比赛时，不考虑主客场，也就是不考虑这两只球队的顺序，两队只要踢一次就行了。那么从n个元素取出m个的组合，有多少种可能呢?

我们假设某种运动需要3支球队一起比赛，那么32支球队就有32x31x30种排列，如果三支球队在一起只要比一场，那么我们要抹除多余的比赛。三支球队按照任意顺序的比赛有3x2x1=6场，所以从32支队伍里取出3支队伍的组合是(32x31x30)/(3x2x1)。基于此，我们可以扩展成以下两种情况。

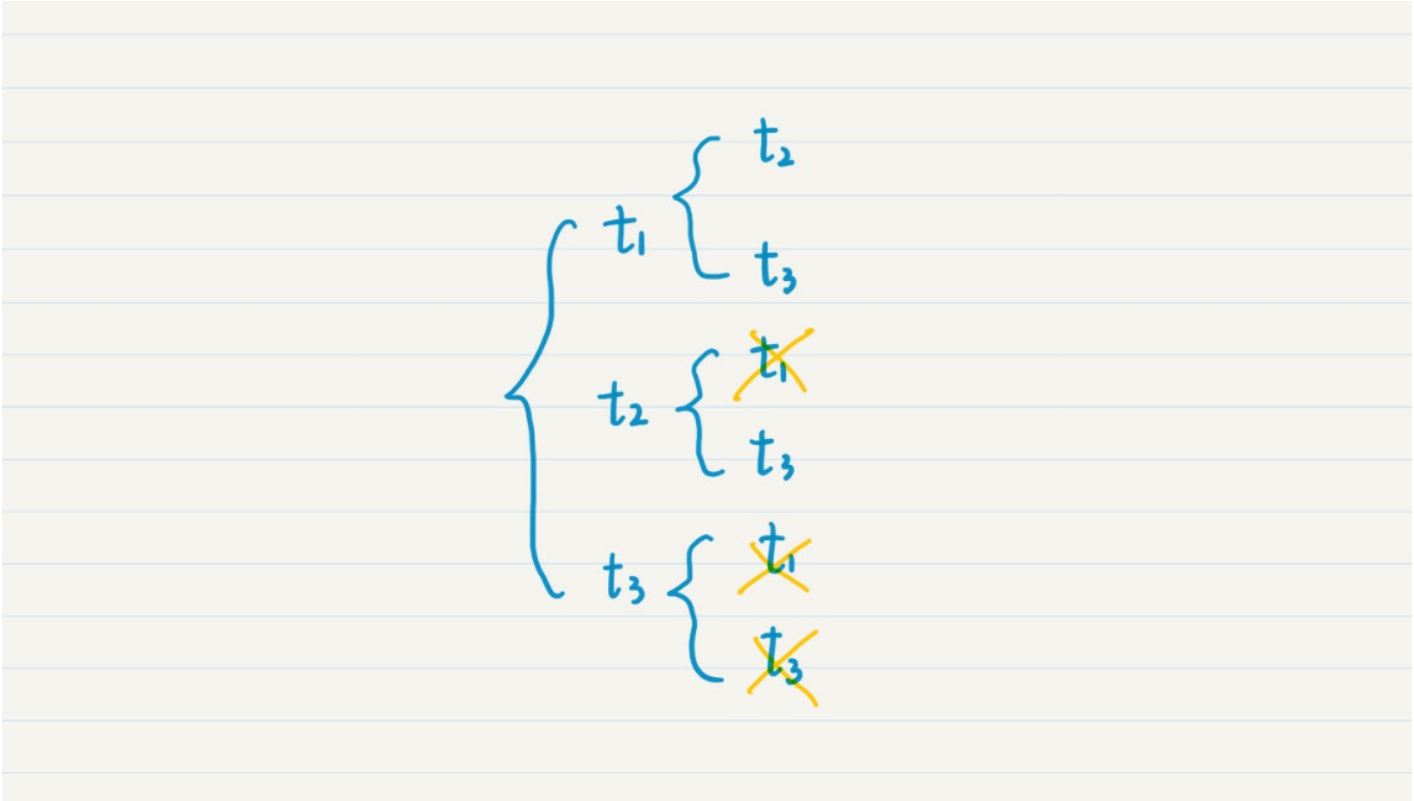
- n个元素里取出m个的组合，可能性数量就是n个里取m个的排列数量，除以m个全排列的数量，也就是(n! / (n-m)!) / m!。
- 对于全组合而言，可能性为2^n种。例如，当n=3的时候，全组合包括了8种情况。

这两点都可以用数学归纳法证明，有兴趣的话你可以自己尝试一下。

如何让计算机来组合队伍?

上一节，我用递归实现了全排列。全组合就是将所有元素列出来，没有太大意义，所以我这里就带你看下，如何使用递归从3个元素中选取2个元素的组合。

我们假设有3个队伍，t1，t2和t3。我还是把递归的选择画成图，这样比较直观，你也好理解。从图中我们可以看出，对于组合而言，由于{t1, t2}已经出现了，因此就无需{t2, t1}。同理，出现{t1, t3}，就无需{t3, t1}等等。对于重复的，我用叉划掉了。这样，最终只有3种组合了。



那么，如何使用代码来实现呢？一种最简单粗暴的做法是：

1. 先实现排列的代码，输出所有的排列。例如{t1, t2}, {t2, t1}；

2. 针对每种排列，对其中的元素按照一定的规则排序。那么上述两种排列经过排序后，就是{t1, t2}, {t1, t2}；

3. 对排序后的排列，去掉重复的那些。上述两种排列最终只保留一个{t1, t2}。

这样做效率就会比较低，很多排列生成之后，最终还是要被当做重复的结果去掉。

显然，还有更好的做法。从图中我们可以看出被划掉的那些，都是那些出现顺序和原有顺序颠倒的元素。

例如，在原有集合中，t1在t2的前面，所以我们划掉了{t2, t1}的组合。这是因为，我们知道t1出现在t2之前，t1的组合中一定已经包含了t2，所以t2的组合就无需再考虑t1了。因此，我只需要在原有的排列代码中，稍作修改，每次传入嵌套函数的剩余元素，不再是所有的未选择元素，而是出现在当前被选元素之后的那些。具体代码是这样的：

```
import java.util.ArrayList;
import java.util.Arrays;

public class Lesson8_1 {

 /**
    * @Description: 使用函数的递归（嵌套）调用，找出所有可能的队伍组合
    * @param teams-目前还剩多少队伍没有参与组合，result-保存当前已经组合的队伍
    * @return void
    */

    public static void combine(ArrayList<String> teams, ArrayList<String> result, int m) {

     // 挑选完了m个元素，输出结果
     if (result.size() == m) {
      System.out.println(result);
      return;
     }

     for (int i = 0; i < teams.size(); i++) {
      // 从剩下的队伍中，选择一队，加入结果
      ArrayList<String> newResult = (ArrayList<String>)(result.clone());
      newResult.add(teams.get(i));

      // 只考虑当前选择之后的所有队伍
      ArrayList<String> rest_teams = new ArrayList<String>(teams.subList(i + 1, teams.size()));

      // 递归调用，对于剩余的队伍继续生成组合
      combine(rest_teams, newResult, m);
     }

    }

}
```

这是一段测试代码，可以帮助我们找到从3个元素中选择2个元素的所有组合。

```
public static void main(String[] args) {

  ArrayList<String> teams = new ArrayList<String>(Arrays.asList("t1", "t2", "t3"));
  Lesson8_1.combine(teams, new ArrayList<String>(), 2);

 }
```

组合的应用：如何高效地处理词组?

组合在计算机领域中也有很多的应用场景。比如大型比赛中赛程的自动安排、多维度的数据分析以及自然语言处理的优化等等。

在我之前的研究工作中，经常要处理一些自然语言，用组合的思想提升系统性能。今天我结合自己亲身的经历，先来说说组合在自然语言处理中的应用。

当时，我们需要将每篇很长的文章，分隔成一个个的单词，然后对每个单词进行索引，便于日后的查询。但是很多时候，光有单个的单词是不够的，还要考虑多个单词所组成的词组。例如，“red bluetooth mouse”这样的词组。

处理词组最常见的一种方式是**多元文法**。什么是多元文法呢？这词看起来很复杂，其实就是把临近的几个单词合并起来，组合一个新的词组。比如我可以把“red”和“bluetooth”合并为“red bluetooth”，还可以把“bluetooth”和“mouse”合并为“bluetooth mouse”。

设计多元文法只是为了方便计算机的处理，而不考虑组合后的词组是不是有正确的语法和语义。例如“red bluetooth”，从人类的角度来看，这个词就很奇怪。但是毕竟它还会生成很多合理的词组，例如“bluetooth mouse”。所以，如果不进行任何深入的语法分析，我们其实没办法区分哪些多元词组是有意义的，哪些是没有意义的，因此最简单的做法就是保留所有词组。

普通的多元文法本身存在一个问题，那就是定死了每个元组内单词出现的顺序。例如，原文中可能出现的是“red bluetooth mouse”，可是用户在查询的时候可能输入的是“bluetooth mouse red”。这么输入肯定不符合语法，但实际上互联网上的用户经常会这么干。

那么，在这种情况下，如果我们只保留原文的“red bluetooth mouse”，就无法将其和用户输入的“bluetooth red mouse”匹配了。所以，如果我们并不要求查询词组中单词所出现的顺序和原文一致，那该怎么办呢?

我当时就在想，可以把每个二元或三元组进行全排列，得到所有的可能。但是这样的话，二元组的数量就会增加1倍，三元组的数量就会增加5倍，一篇文章的数据保存量就会增加3倍左右。我也试过对用户查询做全排列，把原有的二元组查询变为2个不同的二元组查询，把原有的三元组查询变为6个不同的三元组查询，但是事实是，这样会增加实时查询的耗时。

于是，我就想到了组合。多个单词出现时，我并不关心它们的顺序（也就是不关心排列），而只关心它们的组合。因为无需关心顺序，就意味着我可以对多元组内的单词进行某种形式的标准化。即使原来的单词出现顺序有所不同，经过这个标准化过程之后，都会变成唯一的顺序。

例如，“red bluetooth mouse”，这三个词排序后就是“bluetooth,mouse,red”，而“bluetooth red mouse”排序后也是“bluetooth,mouse,red”，自然两者就能匹配上了。我需要做的事情就是在保存文章多元组和处理用户查询这两个阶段分别进行这种排序。这样既可以减少保存的数据量，同时可以减少查询的耗时。这个问题很容易就解决了。怎么样，组合是不是非常神奇?

此外，组合思想还广泛应用在多维度的数据分析中。比如，我们要设计一个连锁店的销售业绩报表。这张报表有若干个属性，

包括分店名称、所在城市、销售品类等等。那么最基本的总结数据包括每个分店的销售额、每个城市的销售额、每个品类的销售额。除了这些最基本的数据，我们还可以利用组合的思想，生成更多的筛选条件。

小结

组合和排列有相似之处，都是从n个元素中取出若干个元素。不过，排列考虑了取出的元素它们之间的顺序，而组合无需考虑这种顺序。这是排列和组合最大的区别。因此，组合适合找到多个元素之间的联系而并不在意它们之间的先后顺序，例如多元文法中的多元组，这有利于避免不必要的数据保存或操作。

具体到编程，组合和排列两者的实现非常类似。区别在于，组合并不考虑挑选出来的元素之间，是如何排序的。所以，在递归的时候，传入下一个嵌套调用函数的剩余元素，只需要包含当前被选元素之后的那些，以避免重复的组合。

今日学习笔记

第8节 组合

1. 组合是指，从n个不同元素中取出m（1≤m≤n）个不同的元素。所有m取值的组合之全集合，我们可以叫作全组合。

2. 在自然语言处理中，我们需要用多元文法把临近的几个单词合并起来，组合成一个新的词组。普通的多元文法定死了每个元组内单词出现的顺序。但是事实上，多个单词出现时，我们可以不用关心它们的顺序，而只关心它们的组合。这样我们就可以对多元组内的单词进行某种形式的标准化。即使原来的单词出现顺序有所不同，经过这个标准化过程之后，都会变成唯一的顺序。

黄申 · 程序员的数学基础课

思考题

假设现在需要设计一个抽奖系统。需要依次从100个人中，抽取三等奖10名，二等奖3名和一等奖1名。请列出所有可能的组合，需要注意的每人最多只能被抽中1次。

欢迎在留言区交作业，并写下你今天的学习笔记。你可以点击“请朋友读”，把今天的内容分享给你的好友，和他一起精进。

精选留言

风轨

从100人中选10人得3等奖，C(100, 10) = 17310309456440
再从剩下90人中选3人的3等奖，C(90, 3) = 117480
再从剩下87人中选1人得1等奖， C(87, 1) = 87
总共有大约有1.8×10^20种可能性，
假设我们的计算机每1ns就能输出1个结果，全部输出来大约要5610年！
假设每个结果占13个字节，把结果保存下来大约要占1995EB，远远大于世界上存储总容量！

当数据量比较小时，还是可以算的：

```
public class Combination {

/**
* 求组合数
*
* @param n
* @param r
* @return
*/
static int c(int n, int r) {
if (r > n) {
return 0;
}
int R = n - r;
int ret = 1;
while (n > R) {
ret *= n--;
```

```
}
while (r > 1) {
ret /= r--;
}
return ret;
}

/**
* 求组合情况
* @param es
* @param r
* @param I 数组es开始取数位置
* @return
*/
static int[][] C(int[] es, int r, int I) {
int[][] rst = new int[c(es.length - I, r)][];
if (r == 1) {
for (int rsti = 0; rsti < rst.length; rsti++, I++) {
rst[rsti] = new int[] { es[I] };
}
} else {
for (int rsti = 0; I < es.length; I++) {
int[][] srst = C(es, r - 1, I + 1);
for (int[] sc : srst) {
int[] t = rst[rsti] = new int[sc.length + 1];
t[0] = es[I];
System.arraycopy(sc, 0, t, 1, sc.length);
rsti++;
}
}
}
return rst;
}

public static void main(String[] args) {
int[][] c = C(new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }, 3, 0);
for (int[] cc : c) {
System.out.println(Arrays.toString(cc));
}
/**
* 输出结果
* [1, 2, 3]
* [1, 2, 4]
* [1, 2, 5]
* [1, 2, 6]
* ··· 省略111行 ···
* [6, 9, 10]
* [7, 8, 9]
* [7, 8, 10]
* [7, 9, 10]
* [8, 9, 10]
```

*

*/

}

}

2018-12-31 16:12

作者回复

确实数字取得太大了

2019-01-01 11:27

行者

案例python实现：

```
comb = ['t1', 't2', 't3', 't4', 't5']
import copy

def combination(n, comb, result):

if n == 0:
print(result)
return
for i in comb:
newResult = copy.copy(result)
newComb = copy.copy(comb)
newResult.append(i)
newComb = list(set(newComb).difference(set(comb[:comb.index(i) + 1])))
combination(n - 1, newComb, newResult)

combination(4, comb, [])
```

2019-01-02 09:07

Wing·三金

思路一：

先运行combine(100, 1)，将所有结果保存。然后用一层迭代对每个结果运行combine(99, 3)，将所有结果append进去。

然后再来一层迭代对上一结果运行combine(96, 10)，同样依次append进去。

此处的关键点在于每个迭代下得将上一结果中的数拿掉，以及得保存临时结果。

此思路也等价于直接上三个嵌套循环+运行递归程序。

思路二：

先运行combine(100, 14)，对每个结果运行combine(14, 10)，再对每个更新的结果运行combine(4, 3)。其实就是思路一逆过来。

理论上二者的复杂度是一样的，用scipy验证了下确实如此。

2018-12-31 17:32

Ricky

```
#include <iostream>
#include <vector>

using namespace std;
void winPrize(int f, int s, int t,
vector<int> &first, vector<int> &second, vector<int> &third, vector<int> &base) {
```

```
if (first.size() == f && second.size() == s && third.size() == t) {
cout << "\nAwards Notification" << endl;
cout << "Prize 1: " << first.back() << endl;
cout << "Prize 2: ";
for (int x: second) {
cout << x << " ";
}
cout << endl;
cout << "Prize 3: ";
for (int y: third) {
cout << y << " ";
}
cout << endl;
return;
}
for (int x: base) {
// 每次仅添加一个成员进奖项圈, 优先级按照一等奖 > 二等奖 > 三等奖
auto f1 = first, s1 = second, t1 = third, left = base;
if (first.size() < f) {
f1.push_back(x);
} else if (second.size() < s) {
s1.push_back(x);
} else if (third.size() < t) {
t1.push_back(x);
}
// 删除成员
auto iter = left.begin();
while (iter != left.end()) {
if (*iter == x) {
iter = left.erase(iter);
} else iter++;
}
winPrize(f, s, t, f1, s1, t1, left);
}
}

void winPrize(int tl, int f, int s, int t) {
vector<int> first, second, third, base;
for (int i = 0; i < tl; ++i) {
base.push_back(i);
}

winPrize(f, s, t, first, second, third, base);
}
int main() {
cout << "Prize Result" << endl;
winPrize(10, 1, 2, 3);
return 0;
}
```

**********结果**********

Awards Notification

Prize 1: 2
Prize 2: 0 6
Prize 3: 8 3 1

Awards Notification
Prize 1: 2
Prize 2: 0 6
Prize 3: 8 3 4

Awards Notification
Prize 1: 2
Prize 2: 0 6
Prize 3: 8 3 5

Awards Notification
Prize 1: 2
Prize 2: 0 6
Prize 3: 8 3 7
2019-01-10 13:24

Youngggg
由于数量过大 设置10个人中 1等奖1名 2等奖2名 3等奖3名

```
import copy
word = []
for i in range(1,11):
word.append(i)
def sort(one_num, two_num, three_num, one_result=[], two_result=[], three_result=[]):
if len(one_result) == one_num and len(two_result) == two_num and len(three_result) == three_num:
print("一等奖", one_result)
print("二等奖", two_result)
print("三等奖", three_result)
return
else:
i = 0
while i < len(word):
if word[i] not in one_result and len(one_result) < one_num:
if len(one_result)>0:
if one_result[-1] > word[i]:
i = i+1
continue
new_one_result = copy.copy(one_result)
new_one_result.append(word[i])
i = i+1
sort(one_num, two_num, three_num, new_one_result, [], [])
elif word[i] not in one_result and word[i] not in two_result and len(two_result) < two_num:
if len(two_result)>0:
if two_result[-1] > word[i]:
i = i + 1
continue
new_two_result = copy.copy(two_result)
new_two_result.append(word[i])
i = i+1
```

sort(one_num, two_num, three_num, one_result, new_two_result, [])
elif word[i] not in one_result and word[i] not in two_result and word[i] not in three_result and len(three_result) < three_num:
if len(three_result)>0:
if three_result[-1] > word[i]:
i = i+1
continue
new_three_result = copy.copy(three_result)
new_three_result.append(word[i])
i = i+1
sort(one_num, two_num, three_num, one_result, two_result, new_three_result)
else:
i = i+1
continue
sort(1, 2, 3, [], [], [])
运行结果：
一等奖 [1]
二等奖 [2, 3]
三等奖 [4, 5, 6]
.....

2019-01-02 17:34

proZhanng
老师能否给个稍微规范点的代码解答，个人基础弱，解题有思路，但是代码实现有点困难，留言区代码稍微有点难以阅读
2019-02-12 23:35

作者回复

我后期会整理好所有的代码和注释，然后上传到github
2019-02-13 05:21

qinggeouye
python (比较粗暴的解法...)
import copy

def lucky_draw_combination(n, m, result=None, all_results=None):
"""
使用函数的递归（嵌套）调用，找出所有可能的中奖者的组合
:param all_results: 中奖者的所有组合
:param n: 参与抽奖的人
:param result: 抽奖结果
:param m: 中奖的人数
:return: None
"""
if result is None:
result = []
if all_results is None:
all_results = []
if len(result) == m:
# print(result)
return all_results.append(result)
for i in range(len(n)):
# 从剩下的人中，抽出一个人加入结果
new_result = copy.copy(result)
new_result.append(n[i])

```
# 每人最多只能被抽中一次，当前被抽中的人之后的人，进入下一次抽奖
rest_n = n[i + 1: len(n)]
# 递归调用 在剩下的人中继续抽奖
lucky_draw_combination(rest_n, m, new_result, all_results)
return all_results


if __name__ == "__main__":
total = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] # 被抽奖人列表
m_ = [3, 2, 1] # 三等奖、二等奖、一等奖的个数
lucky1 = lucky_draw_combination(total, m_[0])
for k1 in lucky1:
total2 = copy.copy(total)
for j1 in k1:
total2.remove(j1)
lucky2 = lucky_draw_combination(total2, m_[1])
for k2 in lucky2:
total3 = copy.copy(total2)
for j2 in k2:
total3.remove(j2)
lucky3 = lucky_draw_combination(total3, m_[2])
for k3 in lucky3:
print(k1, k2, k3)
```

2019-02-10 03:47

夏微凉

黄老师，我这几天一直在纠结思考题，总共10人，一等名1名，二等奖2名，三等3名，还是没有完全理解思路，希望老师方便的时候解答下

2019-01-14 22:45

作者回复

这道题是用到了组合及排列，先看100个人里取1人的数量是C100,1 (格式问题，C100,1表示从100人里取1人的组合数量)，剩下99人里取2人为C99,2，再剩下97人里取3人为C97,3，然后再利用排列，总共可能为C100,1 x C99,2 x C97,3

2019-01-16 01:48

Joe

C++实现

```
#include <cmath>
#include <iostream>
#include <vector>

using namespace std;

class Combination {
private:
const int firstPrize;
const int secondPrize;
const int thirdPrize;

public:
Combination(int x, int y, int z)
: firstPrize(x), secondPrize(y), thirdPrize(z) {
}
```

```
/**
* @description:
* 从10个人中选出三等奖3人，二等奖2人，一等奖1人，不能重复获奖。
* @param {type} rewardNum- 指定赏金数， result- 奖赏方式结果。
* @return: null
*/
void rewardMethods(vector<int> result, vector<int> candidate) {
unsigned int len = thirdPrize + secondPrize + firstPrize;
if (result.size() == len) {
// 输出结果
resultOutput(result);
return;
} else {
for (unsigned int i = 0; i < candidate.size(); i++) {
vector<int> resultNew = result;
resultNew.push_back(candidate[i]);

vector<int> candidateNew;
copyElem(candidate, candidateNew, i + 1);
rewardMethods(resultNew, candidateNew);
}
}
}
// 数据复制函数
void copyElem(vector<int>& input, vector<int>& output, int i) {
vector<int>::iterator it = input.begin() + i;
for (; it < input.end(); it++) {
output.push_back(*it);
}
}
// 数据复制函数
void copyElem(vector<int>& input, vector<int>& output, int i) {
vector<int>::iterator it = input.begin() + i;
for (; it < input.end(); it++) {
output.push_back(*it);
}
}
// 输出结果
void resultOutput(vector<int> res) {
for (unsigned int i = 0; i < res.size(); i++) {
if (i == thirdPrize) cout << '*';
if (i == thirdPrize + secondPrize) cout << '*';
cout << res[i] << ' ';
}
cout << endl;
}
};

// test
int main(void) {
Combination test(1, 2, 3);
```

```
vector<int> res;
vector<int> candidate;
// 输入
for (unsigned int i = 0; i < 10; i++) {
candidate.push_back(i + 1);
}
test.rewardMethods(res, candidate);
}
```

2019-01-10 22:46

神蛊温皇

100人中抽取14个中奖者，每人只能一次，可能的组合有 100!/(100-14)! 种。

2019-01-09 15:17

作者回复

这个是排列的数量

2019-01-14 02:15

曹宇

思路：先抽一等奖，然后在剩余的人中抽二等奖，最后然后在剩余的人中抽三等奖。

```
public static void recGenComb2(final ArrayList<String> originalList, ArrayList<String> remainderList, ArrayList<String>
resultList, int remainderNum, int selectNum){

if (remainderNum == 1 && selectNum == 1){

logger.debug( "一等奖："+resultList );

ArrayList<String> remainOriginalList = cloneExt(originalList, resultList);

recGenComb2(remainOriginalList,remainOriginalList,new ArrayList<>(),0,2);

return;

}

if (remainderNum == 3 && selectNum == 2){

logger.debug( "二等奖：" + resultList );

ArrayList<String> remainOriginalList = cloneExt(originalList, resultList);

recGenComb2(remainOriginalList,remainOriginalList,new ArrayList<>(),0,3);

return;

}

if (remainderNum == 10 && selectNum == 3){

logger.debug( "三等奖：" + resultList );
```

```
return;

}

for (int i=0;i<remainderList.size();i++){

ArrayList<String> copyResultList = (ArrayList<String>)resultList.clone();

copyResultList.add( remainderList.get( i ) );

ArrayList<String> copyRemainderList = new ArrayList<>(remainderList.subList( i+1, remainderList.size() ));

recGenComb2(originalList, copyRemainderList,copyResultList,remainderNum+1,selectNum);

}

}
```

2019-01-09 10:16

作者回复

思路是对的

2019-01-14 02:12

Geek_477c02

老师，可以说下文章里的“词组”是如何总结出来的么？

2019-01-08 09:56

作者回复

你是指词组如何生成的吗？基本的方法是使用n元文法，把相邻的2个或3个单词（或者中文词）合并起来，比如句子"a red blue tooth mouse"，二元文法就可以得到"a red", "red bluetooth", "bluetooth mouse"。我们可以把这个过程，想做从n个元素里跳出2个元素的组合，然后利用组合内不区别元素顺序的特点，对这些单词进行统一的排序，这样"red bluetooth"和"bluetooth red"都能对应的上

2019-01-08 23:34

文刂氵共超

使用C++实现组合问题-从n个数中取出m个不同的元素，不考虑顺序

```
#include <iostream>
#include <vector>

using namespace std;

template <class T>
void PrintVector(vector<T> & DataVec)
{
for (size_t i = 0; i < DataVec.size(); ++i)
{
cout << DataVec[i] << " ";
}
cout << endl;
}

template <class T>
```

```
void Combination(vector<T> &DataVec, int m, vector<T> &resultVec)
{
if (m <= 0 && m > DataVec.size())
{
return;
}

if (resultVec.size() == m)
{
PrintVector(resultVec);
return;
}

for (size_t i = 0; i < DataVec.size(); ++i)
{
vector<T> tempResultVec = resultVec;
tempResultVec.push_back(DataVec[i]);

vector<T> tempDataVec(DataVec.begin()+i+1, DataVec.end());

Combination(tempDataVec, m, tempResultVec);
}
}

int _tmain(int argc, _TCHAR* argv[])
{
vector<int> resultV;
int dataList[] = {2,6,8,23,78,45,32,64};
vector<int> dataV(dataList, dataList+8);

Combination(dataV, 5, resultV);

PrintVector(resultV);

return 0;
}
```

2019-01-04 15:57

作者回复

使用栈来实现递归过程的想法很棒

2019-01-14 01:45

焦太郎

晕了已经，为什么共32X31场比赛，每队却只踢62场?

2019-01-03 10:43

作者回复

因为每场球有两个球队参与，或者换个角度想，每对需要和另外31队踢，主客场制，所以31*2

2019-01-04 02:33

Ryoma

在组合的定义中，老师你说m的范围是1≤m≤n，那全组合应该是不包括的空集的。

不知我说的对不对?

2019-01-02 20:47

作者回复

全组合应该是包括空集的，组合中没有考虑m=0是为了计算组合数量

2019-01-02 23:16

樊少皇

```
public static void lottery(ArrayList<Integer> resultF, ArrayList<Integer> resultS, ArrayList<Integer> resultT,
ArrayList<Integer> remain,ArrayList<Integer> remainList, int f, int s, int t) {
if(resultT.size() == t){
remain = (ArrayList<Integer>) remainList.clone();
remain.removeAll(resultT);
if(resultS.size() == s){
remain = (ArrayList<Integer>) remainList.clone();
remain.removeAll(resultT);
remain.removeAll(resultS);
if(resultF.size() == f){
System.out.print("三等奖:" + resultT + ";");
System.out.print("二等奖:" + resultS+ ";");
System.out.println("一等奖:" + resultF);
return;
} else {
for (int i = 0; i < remain.size(); i++) {
ArrayList<Integer> newResult = (ArrayList<Integer>) resultF.clone();
newResult.add(remain.get(i));
ArrayList<Integer> newRemain = new ArrayList<Integer>(remain.subList(i + 1, remain.size()));
lottery(newResult, resultS, resultT, newRemain,remainList, f, s, t);
}
}
} else {
for (int i = 0; i < remain.size(); i++) {
ArrayList<Integer> newResult = (ArrayList<Integer>) resultS.clone();
newResult.add(remain.get(i));
ArrayList<Integer> newRemain = new ArrayList<Integer>(remain.subList(i + 1, remain.size()));
lottery(resultF, newResult, resultT, newRemain,remainList, f, s, t);
}
}
} else {
for (int i = 0; i < remain.size(); i++) {
ArrayList<Integer> newResult = (ArrayList<Integer>) resultT.clone();
newResult.add(remain.get(i));
ArrayList<Integer> newRemain = new ArrayList<Integer>(remain.subList(i + 1, remain.size()));
lottery(resultF, resultS, newResult, newRemain,remainList, f, s, t);
}
}
}
```

2019-01-02 17:56

菩提

思考题，我对题目的理解是从100个选14个排列，C100,4。程序实现，将老师文章中的示例代码从3个选2排列修改，t1,t2,t3改为 t1,t2.....t99,t100,排列方法调用第三个参数2改为14。

100x99x.....87/(14x13x.....x2x1)种结果。

2019-01-01 10:06

Being

老师，这个数是不是特别大呀，用排列的公式 P(14,100) /P(1,1)/P(3,3)/P(10,10) 对吗？思路就是一共有1+3+10=14个人中奖，即在100个人中取14个人进行排序，然后去掉一、二、三等奖各自的重复情况，就得到了最终中奖情况的组合。

我用Python实现的，但是数值太大，输出不了，我就改用小的数进行了验证

```
rewordMap = {'一等奖':1,'二等奖':2,'三等奖':3}
peopleSize = 7
peopleList = []

for i in range(peopleSize):
peopleList.append("People" + str(i + 1))

def CombinationReword(rewordList, rewordSet, peopleList, rewordSize):
if len(rewordSet) == rewordSize:
rewordList.append(rewordSet.copy())
return

for people in peopleList:
newRewordSet = rewordSet.copy()
newRewordSet.add(people)

newPeopleList = peopleList[peopleList.index(people) + 1:]
CombinationReword(rewordList, newRewordSet, newPeopleList, rewordSize)

#===== 计算各个奖项的组合情况
rewordFirstList = []
CombinationReword(rewordFirstList,set(), peopleList, rewordMap['一等奖'])

rewordSecondList = []
CombinationReword(rewordSecondList,set(), peopleList, rewordMap['二等奖'])

rewordThridList = []
CombinationReword(rewordThridList,set(), peopleList,rewordMap['三等奖'])
#=====

#===== 将各个奖项的的set取交集，若有重复，去除，未有重复，则为一种情况，输出
count = 0
for thirdSet in rewordThridList:
for secondSet in rewordSecondList:
for firstSet in rewordFirstList:
set1 = thirdSet & secondSet
if len(set1) != 0:
continue
set2 = thirdSet & firstSet
if len(set2) != 0:
continue
set3 = secondSet & firstSet
if len(set3) != 0:
continue

count += 1
print("First")
print(firstSet)
print("Second")
print(secondSet)
```

```
print("Third")
print(thirdSet)
print("======================")

print(count)
```

2018-12-31 16:27

作者回复

确实是很大的数字，可以用小的数字来测试

2019-01-01 11:31

JonathanShi

依次抽奖的话，三等奖为 C100 10 (100! ÷90!)÷10!

二等奖。C90 3；(90!÷87!)÷3!

一等奖C 87 1

总的可能应该是三项的和，不知道是不是这个结果，请老师指正

2018-12-31 16:18

作者回复

总的可能是三者的乘积，是个非常大的数字

2019-01-01 11:30