题目：assignment 3 T3

给定一个句子，求出所有包含词汇大小写的情况（求子集）

其中，符号与空格不变

题意就是说给一句话，这句话里面有n个英语单词，每个单词有可能小写也可能大写，给出所有情况组成的集合。抽象成数学就是一个枚举所有子集的问题。

主要思路：

设计两个set，其中一个为原来的set，另一个set为替换后的结果。由于提供了将句子变为vector元素的函数，于是可以直接将vector中元素替换成大写，使用point来确定位置。

由于求子集对于同一个元素有两个不同的选择（变大写，不变大写），两者应同时进行，同时加入递归操作。在每个词操作完后，进行递归ans+=allEmphasesOfhelp(elems,sofar,point+1);

直至point==size，返回sofar里的所有句子。



注意：

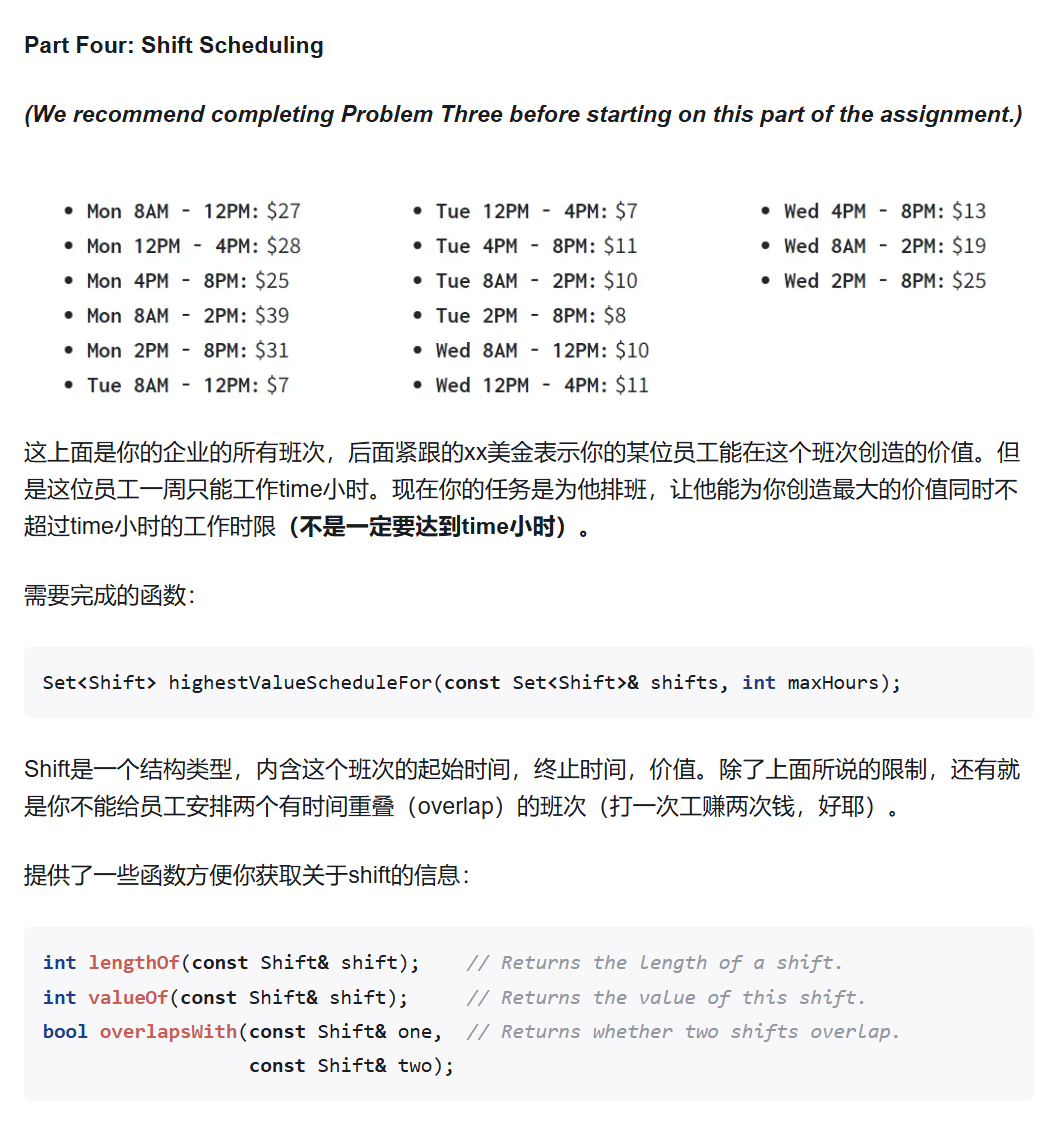
for(string &s:elems)

s=toLowerCase(s);

必须使用引用，用：进行循环是只是提供了一个副本，并不会真正改变elems里的元素。

传参数记得应用！

题目：assignment 3 T4



与上一题类似，这也是一道运用递归来确定答案集合的题，不一样的是这道题并不是在原来的基础上进行查找修改，而是需要比较来得出最优值。

主要思路：

由于并不知道是否在下一次shift后超时，所以本题函数中的三个set应该设为remain，ifchoose与sofar，用于获取set中的元素（每次递归后减去首元素，再读出首元素）；如果选择了这个元素（是否超时？是否最优？）；目前已选择的元素（在递归过程中与ifchoose相比较，得出最优解）。

，图形用户界面, 文本, 应用程序

描述已自动生成

若均不超时，遍历两种情况（由于ifchoose可能在后续选择中超时，所以不一定比sofar大）

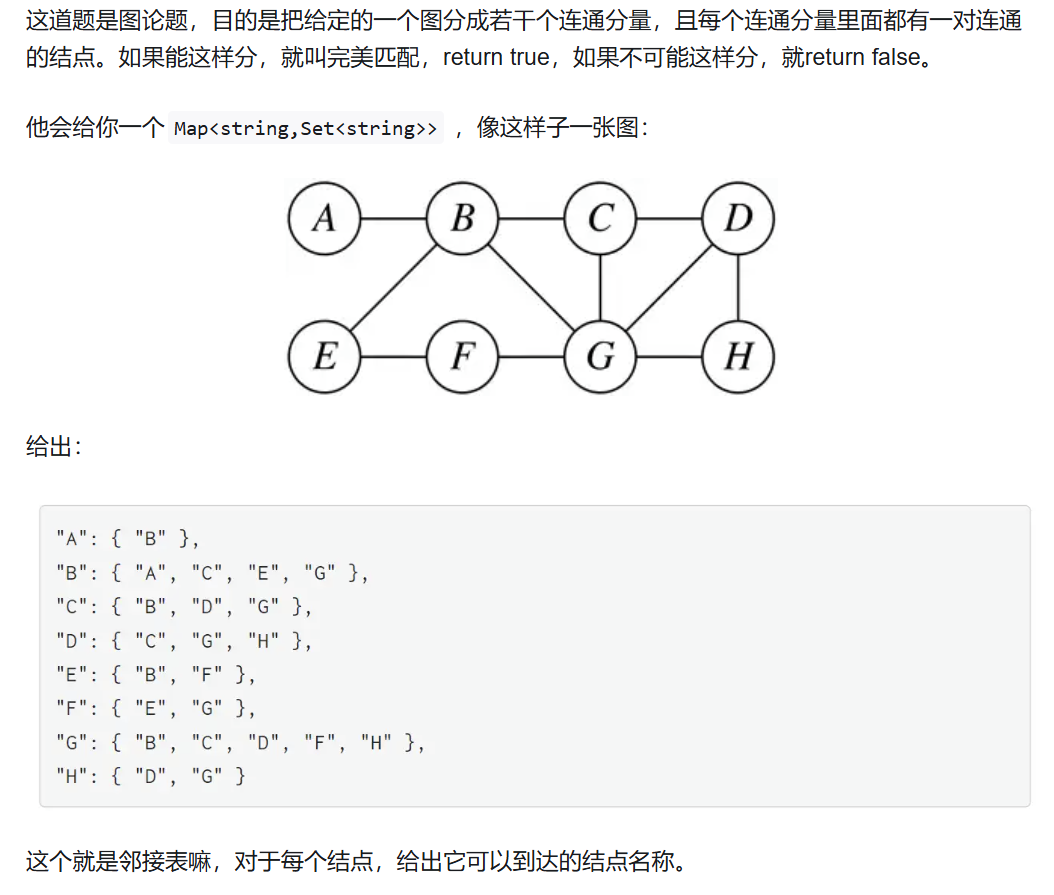
取大者返回，并且在超时情况中返回未choose，也就是sofar的递归返回值，相当于不选这个元素

注意：

传参需要经过const，保证原来的值不变。值得注意的是，两个值都是通过参数的拷贝进行修改的，一个是remain拷贝了shifts以及ifchoose拷贝了sofar。

第三单元主要在讲如何think recursively，关键点主要是：不要太关注递归的过程，主要关注每个函数能产生什么结果。思考时要考虑参数以及最后的返回值是什么，来构思中间的代码。

题目 assignment4 T3



Part 1 实现bool函数

bool hasPerfectMatching(**const** Map<string, Set<string>>& possibleLinks,

Set<Pair>& matching);

用于判断给出的map是否存在perfectmatching，即没有不与人连接的节点。

思路：

用递归，判断在取出两个人之后剩下情况能否match，若能则返回true，需先判断是否为已选的组合（存在ab，ba）

文本

描述已自动生成

比较简单，但是对于这一类问题非常典型，即通过键值对来进行遍历，递归。

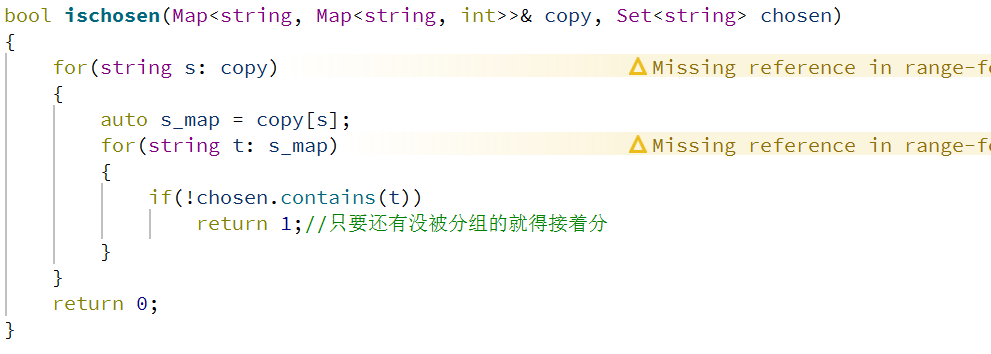
PART2

在第一问基础上添加配对权重值，要求找出权重最大的配对（不一定要是perfectmatch）

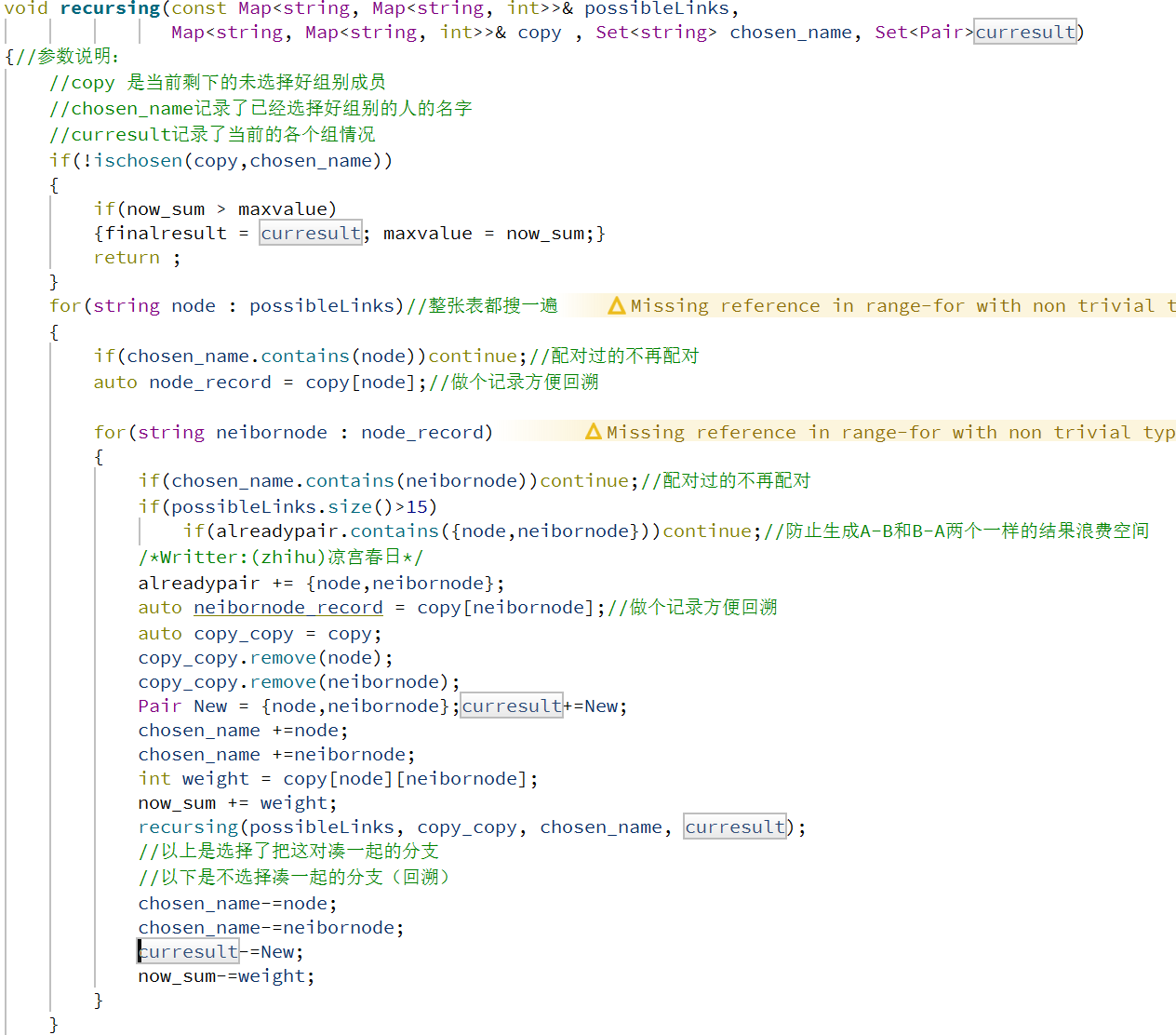
在配对的基础上引入了权值

思路：

进行递归是使用回溯功能，若大于目前权重最大值则使用，若小于则不用。在离散数学里，这应该是图的某种分支，利用递归来求最大值。



首先需要一个函数来判断是否被都选中过



中心函数部分：

首先是判断是否每个元素都被选过了，若还有剩下的，则将当前最大数记下来，继续递归。

后续就跟上方一样的取出数据进行遍历，由于需要检测是否是最大取法，所有需要给copy创建一个副本copy——copy，来进行替代减少选取数的情况，进行递归。在递归后，若不是最大情况，则在基础上减去加入的元素，再次递归。

题目 assignment4 T4



思路：

与前几题类似，这也是一道用图来进行递归的题目，不同的是，这道题如果单纯枚举将会产生很多大case无法通过，所以不能利用for来一个个枚举递归。把每个城市与他周边的城市划分成一个小group，对于小group进行判断是否成功。



首先是辅助函数，要验证全部是否都能领到就要先验证单个中心城是否符合，所以先checksingle，再checkall，同时，由于现存城市需要一个全城市的set，则构造一个init函数进行。



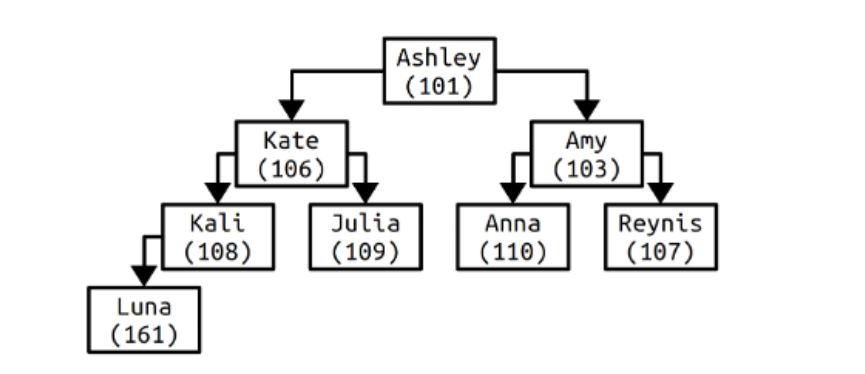
对于主函数，首先只要不是全部覆盖，就一直进行递归，直到放置城市数量为0（即无城市可放且无法全覆盖）。然后又是熟悉的读取set中数据，先假设某城市为物资存放地，再对其小组做隔离，对剩下的城市递归。若不行，则再对该小组的其他成员进行同样操作。注意，在remain去掉的时候，要减掉一整个组，而不是单个元素。

总结：

Assignment4也还是主要讲递归，3主要是树的递归，用于处理选择，子集问题比较多，4主要是图的递归，同类问题的方法也比较公式，主要难度在于取辅助函数来达到对应的功能。还有一些应对大case的技巧，用全局变量，划分递归等，总体难度比较大。

ASSIGNMENT 6

本节课主要是讲优先队列也就是堆的实现。

堆就是一个有序的二叉树，

将值最小的元素放在最上面，可以实现多种操作。比较常见的就是enqueue与dequeue。

先说enqueue，新元素加入堆，需要重新进行排序，也就是要逐级进行大小比较。

完全二叉树有一个比较重要的性质，就是叶子等于父节点的两倍或两倍加一，运用这个性质，如果用数组表示一个堆，我们可以从下往上做比较。

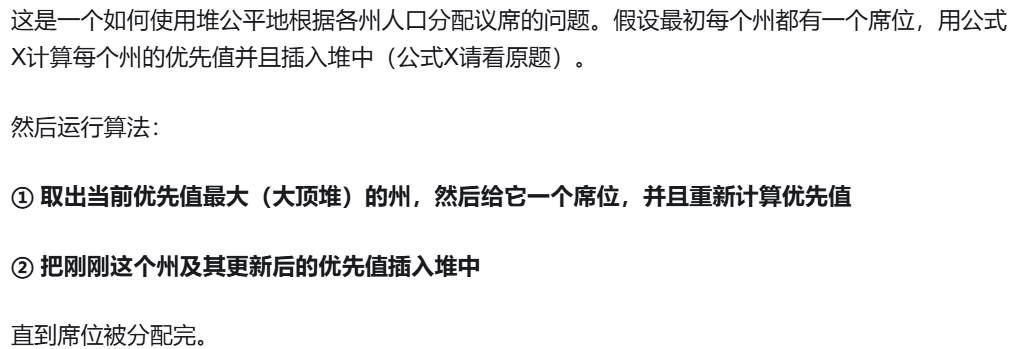
每级比较完，若子小于父，则进行交换，将子与父的位置互换，同时将父节点变为再上一级的父节点，也就是此时的子节点/2. 由于int型是向下取整，所有不用担心舍入的问题。

然后就是dequeue，也就是删除堆顶。这里我们用特殊的最简单方法，将顶部数与最后的数互换，此时最小的数可以自由的删去，并不会出现空缺的情况。在互换之后，再进行自上而下的冒泡排序操作。也就是并反复交换该项和它***权重较低的***子项，直到它没有子项或权重低于它的所有子项



若左分支大于右分支，则用左分支的值进行比较，否则赋值，节省了运行时间。同时进行位置互换，并再一次进行向下冒泡。注意为了防止超出内存，应该限制最后一次向下时处于倒数第二行。

T3



问题的最大难点就是如何让一个堆便利的返回最大值而不是最小值。有一个非常简单的方法：让所有权值都乘-1，这样，每次返回的项就是原来数值最大的项。

ASSIGNMENT 7

