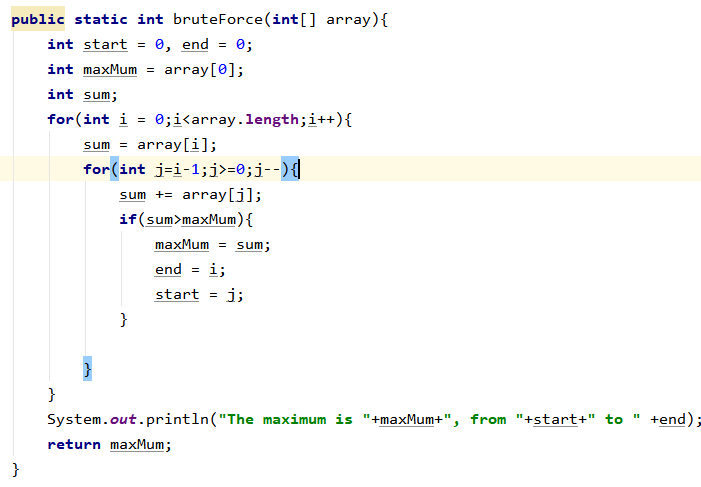
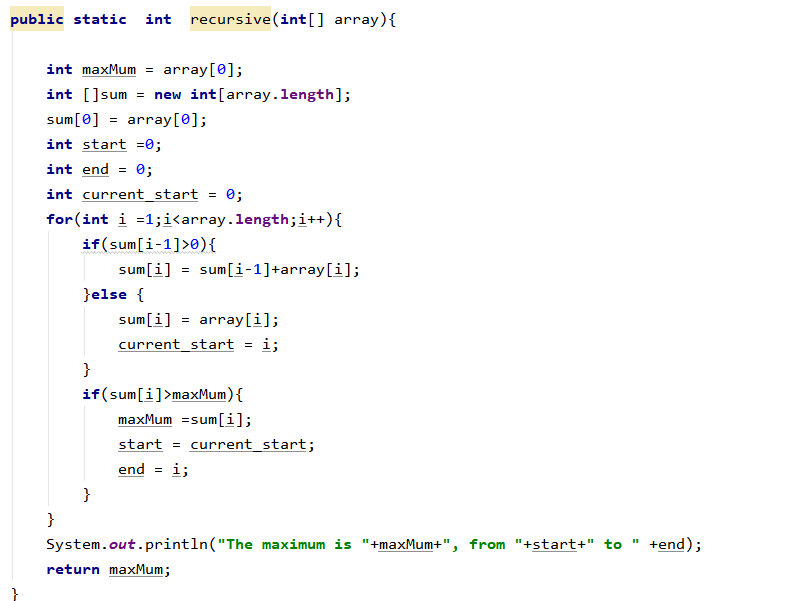
pj2实验文档

1. The maximum-subarray problem
2. 暴力算法和递归的算法实现：



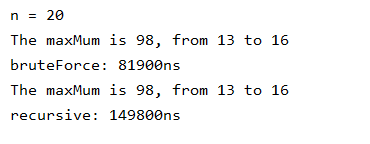
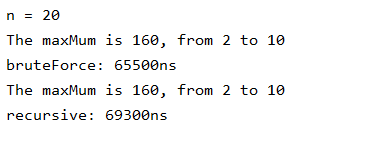


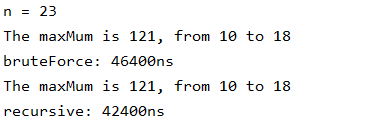
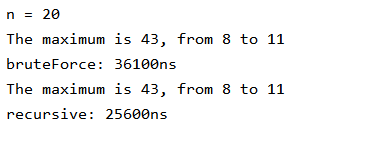
传入参数为每一天相对于前一天的价格变化的数组（a0=0），运行结果截图如下：

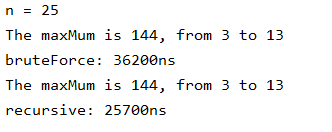
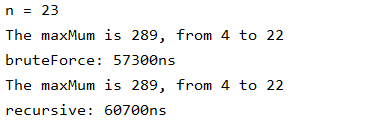


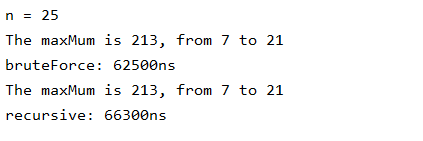
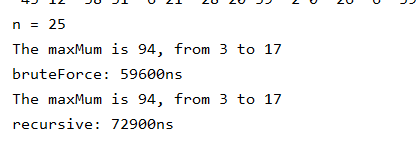
得到的最大收益为43，第8天买入，第11天卖出；

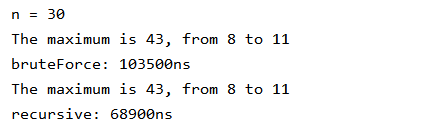
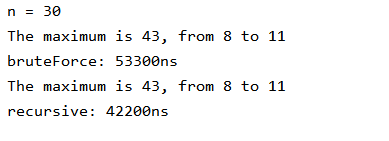
通过改变传入的数组的大小，可以发现，当数组较小的时候，暴力破解的用时比动态规划长，当数组大小**n大于25**的时候，动态规划击败了暴力破解，运行截图如下：



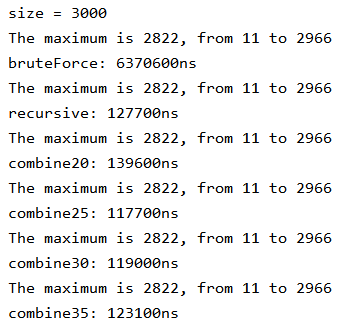
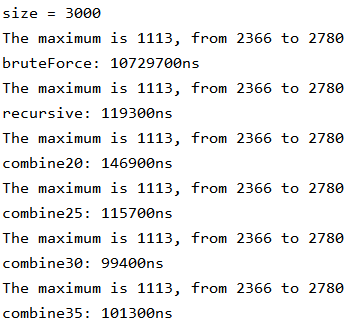


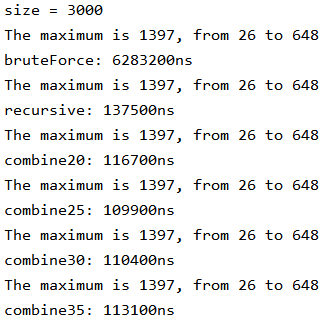
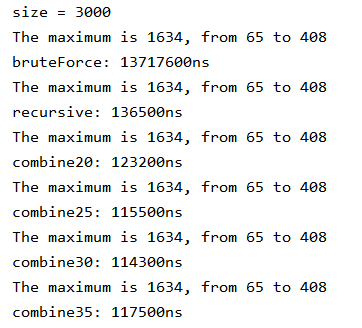




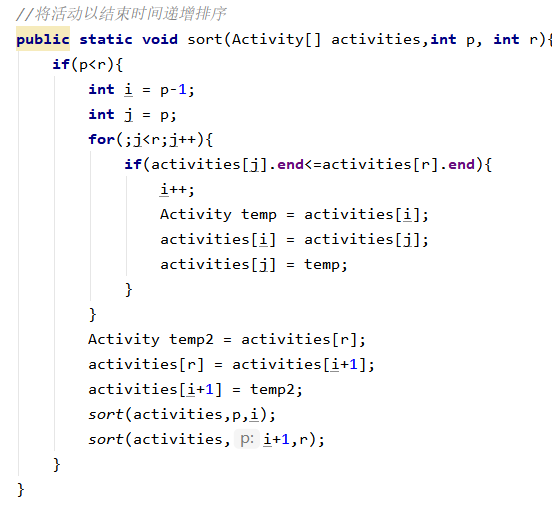


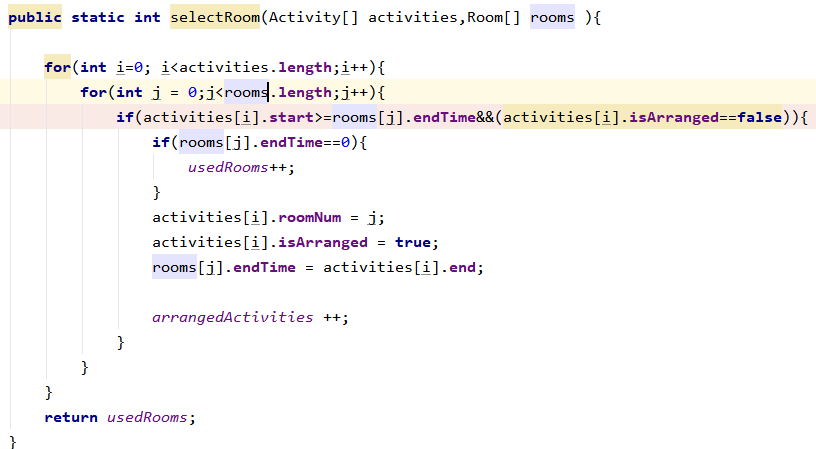
然后编写组合算法，当**n为30**左右组合算法更快。

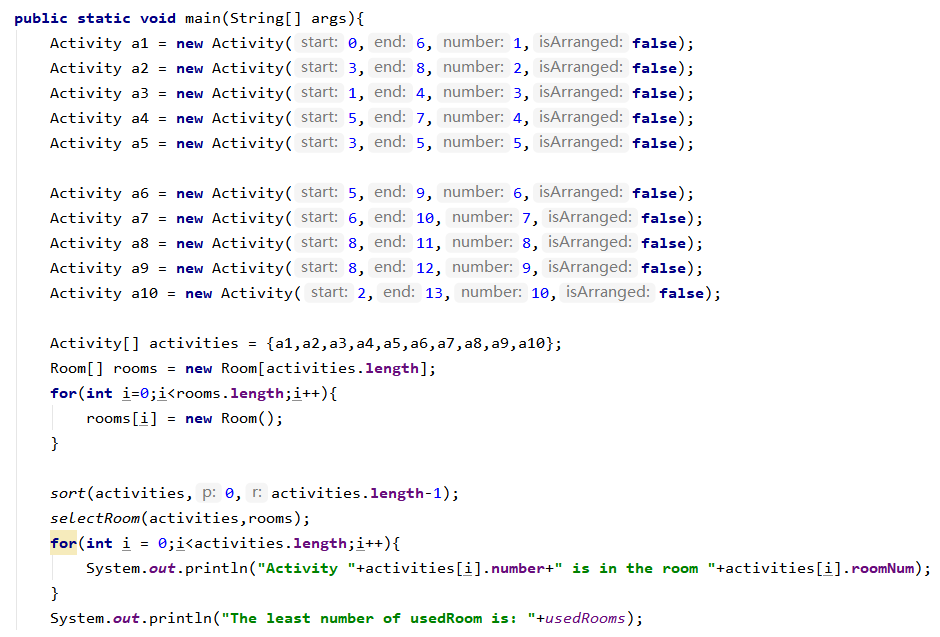
1. 可以令一开始的maximum为0，这样如果不存在和大于零的子序列，即空序列，则返回0。
2. Interval-graph coloring problem
3. 代码部分截图如下，先按照活动结束时间用快排排序，时间复杂度为O(nlgn)，之后是为每个活动安排房间，有两个for循环嵌套，复杂度为O(n²)，n为活动数目，给出的最大房间数目也为n，所以算法总的时间复杂度为O(n²)。

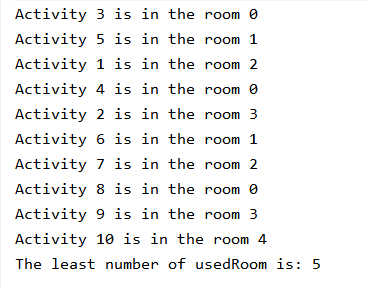




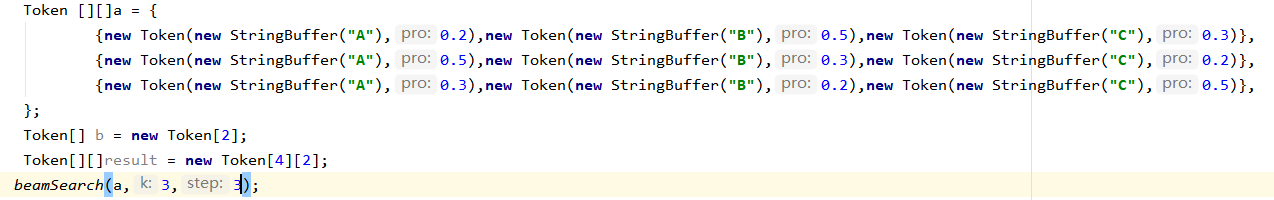
1. 构建的测试实例和运行截图如下：

每个活动都是一个对象，有开始时间、结束时间、活动编号和是否被安排四个属性，所有需要被安排的活动构成一个数组；还有一个Room数组，每个元素都是一个room对象，room对象有活动总结束时间的属性。然后将活动数组和room数组传入函数selectRoom，即可得到结果。

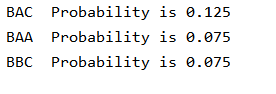




1. Beam Search
2. 示例和k=3和2运行结果如下，



k=3：



k=2：



1. 贪婪算法、Beam Search、暴力求解的选择：
2. 当数据较少的时候，三者速度没有较大差别，均可选择，其中暴力求解的结果更多，可供选择的范围更广泛，此时暴力求解可能更加占优势；
3. 当数据较多时，暴力求解可能耗时较长，需要的内存也更多，此时暴力求解是最次的选择。如果确保其他的资源和消耗足够满足最优解，就可以选择用贪婪算法直接得到最优解。但是有时候可能其他方面的资源不够，或者最优解需要的消耗过高，就可以考虑用Beam Search找到相对较优的多个解，然后综合其他资源等方面综合考虑选择。