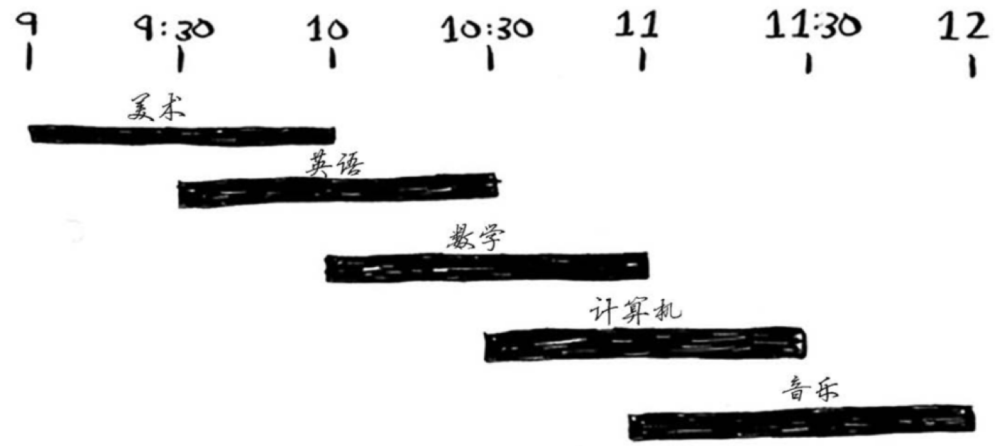
贪心算法

教室调度问题

假设有这样一张课程表，你需要将尽可能多的课程安排在某一间教室！



课程	开始时间	结束时间
美术	9 AM	10 AM
英语	9:30 AM	10:30 AM
数学	10 AM	11 AM
计算机	10:30 AM	11:30 AM
音乐	11 AM	12 PM

如何做？

- 首先，选出最早结束的课程作为第一节课
- 其次，以上一节课程的结束时间为起点，选择第二个最早结束的课程
- 重复如上

美术	9AM	10AM	✓
英语	9:30AM	10:30AM	
数学	10AM	11AM	
计算机	10:30AM	11:30AM	
音乐	11AM	12PM	

美术	9AM	10AM	✓
英语	9:30AM	10:30AM	✗
数学	10AM	11AM	✓
计算机	10:30AM	11:30AM	
音乐	11AM	12PM	

美术	9AM	10AM	✓
英语	9:30AM	10:30AM	✗
数学	10AM	11AM	✓
计算机	10:30AM	11:30AM	✗
音乐	11AM	12PM	✓

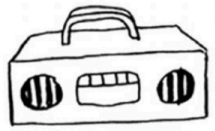
贪心算法的核心思想

呐，其实很简单！

每步都采取最优的做法

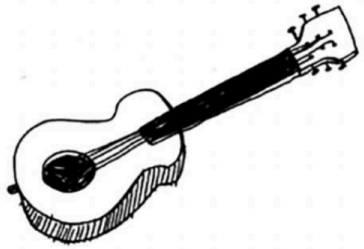
背包问题

假设你是一个贪婪的小偷，背着一个可以装35KG重的背包，在商场伺机行盗，你力图使此行收益最大，你该如何做？



音响
3000元
30KG

笔记本电脑
2000元
20KG



吉他
1500元
15KG

完美是优秀的敌人

在有些情况下，完美是优秀的敌人。有时候你只需找到一个能够大致解决问题的算法，此时贪婪算法正好可派上用场，因为它们实现起来很容易，得到的结果又与正确结果相当接近。

集合覆盖问题

假设你需要投放一个广告，要覆盖全国所有城市。为此，你需要决定在哪些平台上投放，每个平台投放都需要支付费用，你需要尽可能减少平台的数量。这些平台如下：

A	{北京,上海,广州,深圳,成都,重庆}
B	{北京,广州,河北,山东,青岛,贵州}
C	{广元,平川,江西,温州,宜宾}
D	{上海,杭州,宁波,江西}
E	{江西,平川,河北}

首先，创建一个数组用来存储所有的城市

```
String[] cities = {"北京","上海","广州","深圳","成都","重庆","山东","青岛","贵州","江西","宜宾","杭州","宁波"};
```

然后，我们模拟平台并初始化对应的数据

```
Map<String,Set<String>> platforms = new HashMap<>();
platforms.put("A",new HashSet(Arrays.asList(new String[]{"北京","上海","广州","深圳","成都","重庆"})));
platforms.put("B",new HashSet(Arrays.asList(new String[]{"北京","广州","山东","青岛","贵州"})));
platforms.put("C",new HashSet(Arrays.asList(new String[]{"广元","江西","温州","宜宾"})));
platforms.put("D",new HashSet(Arrays.asList(new String[]{"上海","杭州","宁波","江西"})));
platforms.put("E",new HashSet(Arrays.asList(new String[]{"江西","河北"})));
```

创建一个用于保存结果的MAP字典

```
Map<String,Set<String>> result = new HashMap<>();
```

然后，我们模拟平台并初始化对应的数据

```
Set<String> citySet = new HashSet<>();
while (true){
    String key = null;
    int countUnion = 0;
    for (String pName : platforms.keySet()){
        Set<String> platform = platforms.get(pName);
        Set temp = Sets.union(citySet,platform);
        if (temp.size()>countUnion){
            key = pName;
            countUnion = temp.size();
        }
    }
    citySet.addAll(platforms.get(key));
    result.put(key,platforms.get(key));
    //如果Set集合中的城市数目大于等于了citys表示已经找到结果了，终止while循环
    if (citySet.size()>=cities.length)break;
}
```

结 果 输 出

```
System.out.println(Arrays.toString(result.keySet().toArray()));
```

Thanks for your time