六、典型算法

6.1 贪心算法

6.1.1 贪心思想

6.1.2 活动选择案例

6.1.3 装载问题案例

6.1.4 其他案例

六、典型算法

6.1.4 其他案例

1. 分发饼干问题
2. 柠檬水找零问题
3. 最大子数组和问题
4. 跳跃游戏及其增强版
5. 弓箭射气球
6. 迷离傍地走
7. 作业加工问题

分发饼干问题（leetcode/455）

假设你是一位很棒的家长，想要给孩子们分一些小饼干。但是，每个孩子最多只能给一块饼干。

对每个孩子 i，都有一个胃口值 g[i]，这是能让孩子们满足胃口的饼干的最小尺寸；并且每块饼干 j，都有一个尺寸 s[j] 。如果 s[j] >= g[i]，我们可以将这个饼干 j 分配给孩子 i ，这个孩子会得到满足。你的目标是尽可能满足越多数量的孩子，并输出这个最大数值。

胃口值：1, 2, 7, 10

饼干尺寸：1, 3, 5, 9

贪心思路：充分利用饼干尺寸，喂给胃口尽可能大的小朋友

分发饼干问题（leetcode/455）

```
int findContentChildren(vector<int>& g, vector<int>& s) {

}
```

六、典型算法

6.1.4 其他案例

1. 分发饼干问题
2. 柠檬水找零问题
3. 最大子数组和问题
4. 跳跃游戏及其增强版
5. 弓箭射气球
6. 迷离傍地走
7. 作业加工问题

柠檬水找零问题（leetcode/860）

每一杯柠檬水的售价为 5 元。顾客依次排队购买，一次一杯。客户可能交给付5 元、10 元或 20 元。你必须给每个顾客正确找零，以使得每位顾客最终支付5元。一开始你手头没有任何零钱。

给你一个整数数组 bills ，其中 bills[i] 是第 i 位顾客付的账。如果你能给每位顾客正确找零，返回 true ，否则返回 false 。

如果给5元，怎么办？ 收下即可 bills=[5,5,5,10,20]

如果给10元，怎么办？找5元 bills=[10,5]

如果给20元，怎么办？使用10+5或3*5找零 bills=[5,5,10,10,20]

5的作用更大，不仅可用于20，还可用于10，因此尽可能留5

贪心思路：找零时，使用最大面额优先

柠檬水找零问题（leetcode/860）

```
bool lemonadeChange(vector<int>& bills) {

}
```

六、典型算法

6.1.4 其他案例

1. 分发饼干问题
2. 柠檬水找零问题
3. 最大子数组和问题
4. 跳跃游戏及其增强版
5. 弓箭射气球
6. 迷离傍地走
7. 作业加工问题

最大子数组和（leetcode/53）

给你一个整数数组 nums ，请你找出一个具有最大和的连续子数组（子数组最少包含一个元素），返回其最大和。

子数组是指数组中的一个连续部分。

nums = [-2,1,-3,4,-1,2,1,-5,4]

连续子数组 [4,-1,2,1] 的和最大，为 6

nums = [5,4,-1,7,8]

连续子数组 [5,4,-1,7,8] 的和最大，为23

普通思路：暴力求解，外层控制起始，内层控制长度 $O(n^2)$

贪心依据： 遇到负数就停止吗？ 不对！

当前子序列的和小于0即应舍弃

最大子数组和（leetcode/53）

```
int maxSubArray(int* nums, int numsSize) {

}
```

六、典型算法

6.1.4 其他案例

1. 分发饼干问题
2. 柠檬水找零问题
3. 最大子数组和问题
4. 跳跃游戏及其增强版
5. 弓箭射气球
6. 迷离傍地走
7. 作业加工问题

跳跃游戏（leetcode/55）

给你一个非负整数数组 nums ，你最初位于数组的 第一个下标 。数组中的每个元素代表你在该位置可以跳跃的最大长度。

判断你是否能够到达最后一个下标，如果可以，返回 true ；否则，返回 false 。

nums = [2,3,1,1,4]

先跳1步，再跳3步

nums = [3,2,1,0,4]

到达下标3后无法继续跳

常见误区：纠结于跳几步，用回溯法

贪心依据： 追求最大覆盖范围

跳跃游戏（leetcode/55）

```
bool canJump(int* nums, int numsSize) {



}
```

跳跃游戏2（leetcode/45）

给你一个非负整数数组 nums ，你最初位于数组的 第一个下标 。数组中的每个元素代表你在该位置可以跳跃的最大长度。

判断最少需要跳几步可以达到数组末尾位置

nums = [2,3,1,1,4]

先跳1步，再跳3步，共跳3步

nums = [3,2,1,0,4]

到达下标3后无法继续跳

常见误区：选择每一步尽可能跳最远的贪心策略

贪心依据： 用尽可能少的步数追求更大的覆盖范围

跳跃游戏（leetcode/45）

```
int jump(int* nums, int numsSize) {

}
```

六、典型算法

6.1.4 其他案例

1. 分发饼干问题
2. 柠檬水找零问题
3. 最大子数组和问题
4. 跳跃游戏及其增强版
5. 弓箭射气球
6. 迷离傍地走
7. 作业加工问题

弓箭射气球（leetcode/452）

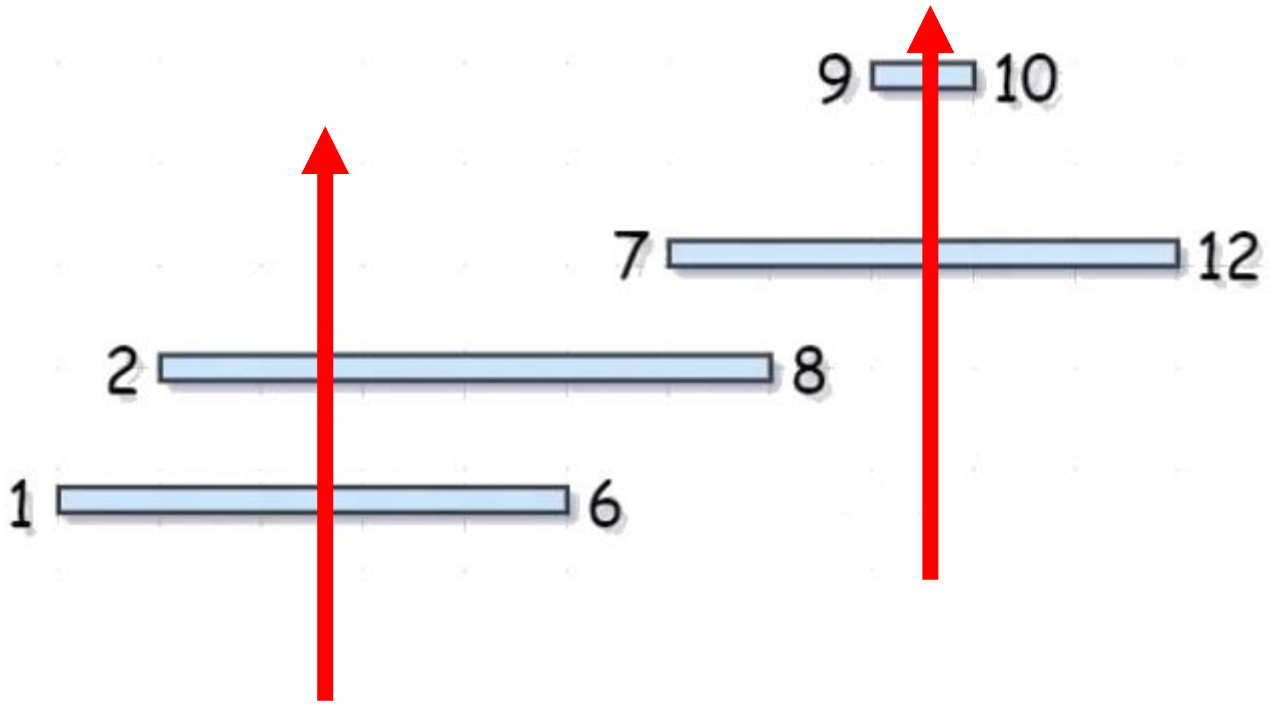
有一些球形气球贴在一堵用 XY 平面表示的墙面上。墙面上的气球记录在整数数组 points，其中points[i] = [xstart, xend] 表示水平直径在 xstart 和 xend之间的气球。你不知道气球的确切 y 坐标。

一支弓箭可以沿着 x 轴从不同点 完全垂直 地射出。在坐标 x 处射出一支箭，若有一个气球的直径的开始和结束坐标为 xstart，xend，且满足 xstart ≤ x ≤ xend，则该气球会被 引爆 。可以射出的弓箭的数量 没有限制 。 弓箭一旦被射出之后，可以无限地前进。

给你一个数组 points ，返回引爆所有气球所必须射出的 最小 弓箭数 。

弓箭射气球（leetcode/452)

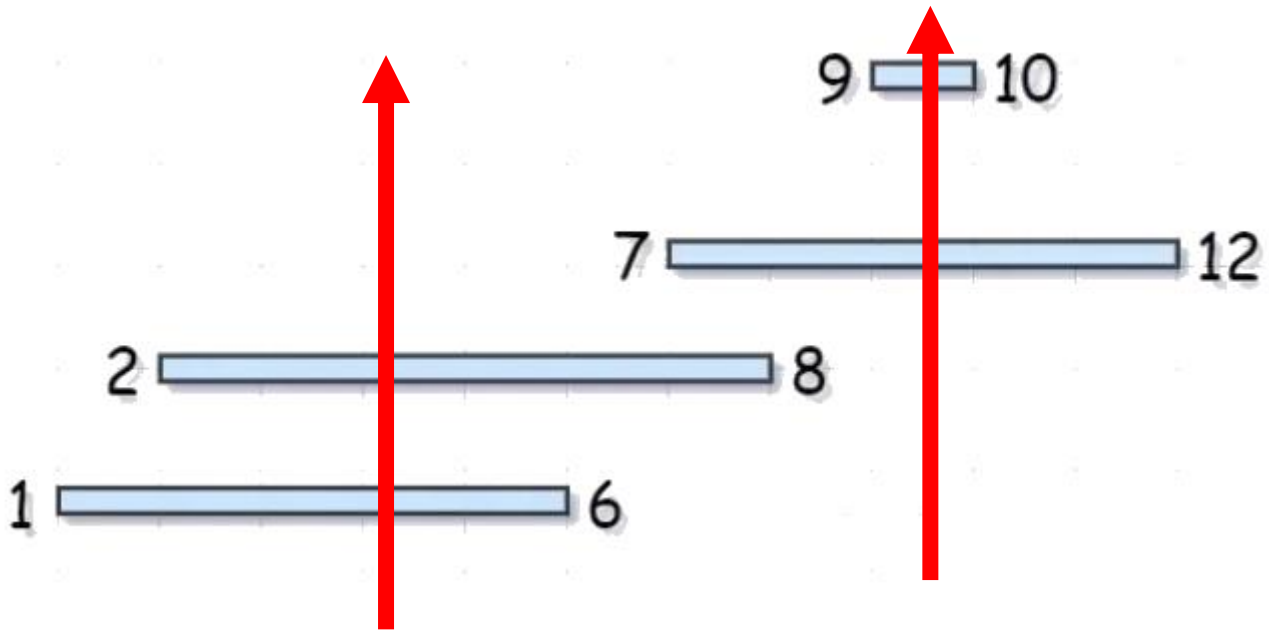
例如：气球坐标数组分别为：[9, 10]，[2, 8]，[1, 6]，[7, 12]



怎么设置贪心策略?

弓箭射气球（leetcode/452）

例如：气球坐标数组分别为：[9, 10]，[2, 8]，[1, 6]，[7, 12]



先将气球按左端点位置排序，判断相邻气球右端点是否重合

贪心策略：一根箭尽可能射中相邻且重叠的气球

如何判断相邻?

弓箭射气球（leetcode/452）

```
class Solution {
public:
    int findMinArrowShots(vector<vector<int>> &points) {



    }
};
```

六、典型算法

6.1.4 其他案例

1. 分发饼干问题
2. 柠檬水找零问题
3. 最大子数组和问题
4. 跳跃游戏及其增强版
5. 弓箭射气球
6. 迷离傍地走
7. 作业加工问题

迷离傍地走

为了庆祝汉朝盛世，武后决定拜孙武和王翦对春夏秋冬4人进行军事训练。

孙武和王翳分别负责四人的站军姿和踢正步科目；根据军训要求，只有先学会站军姿之后才能进行踢正步训练。不同人在不同科目上有差别。每次教官只能教1个人，学习时间如下表：

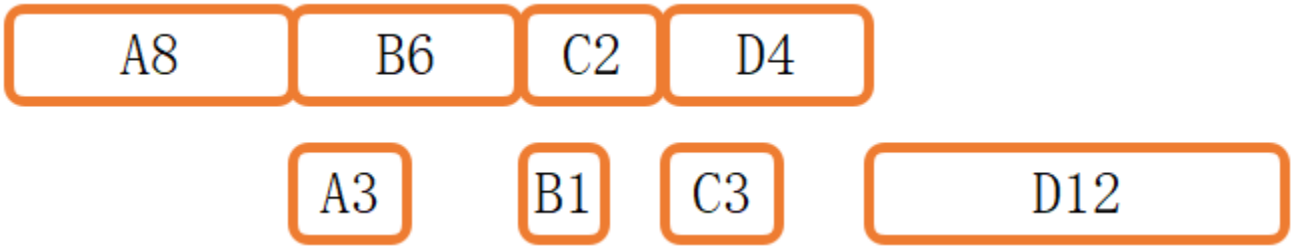
教官	春	夏	秋	冬
孙武	8	6	2	4
王翦	3	1	3	12

请问，如何安排四人的学习次序，使得孙武和王翦能够在最短时间使得所有人完成训练?

迷离傍地走

如按原始次序安排训练：

科目	A	B	C	D
站军姿	8	6	2	4
踢正步	3	1	3	12



原始次序进行训练，所有人完成训练一共需要32小时

迷离傍地走

如按原始次序安排训练：

科目	A	B	C	D
站军姿	8	6	2	4
踢正步	3	1	3	12

贪心策略：第一步开始和第二步结束时空闲时间最少

正步用时少的第二步靠后，军姿用时少的第一步靠前

所有用时最短者：B踢正步，因此B最后

第二短者为C站军姿，因此C最先

A和D中最短为A踢正步，因此A倒数第二

D站军姿4小时，正数第二

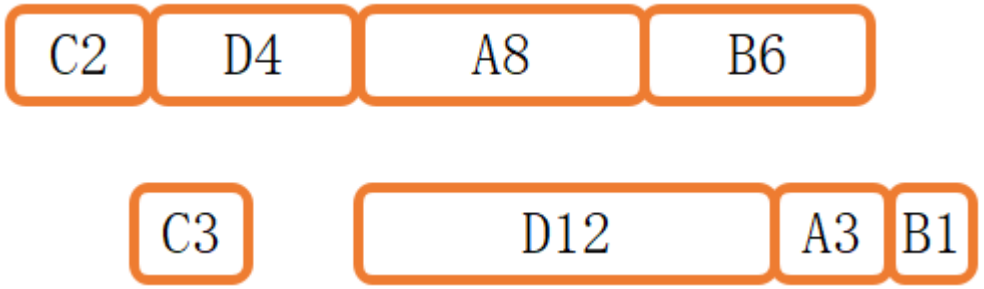
迷离傍地走

如按原始次序安排训练：

科目	A	B	C	D
站军姿	8	6	2	4
踢正步	3	1	3	12

贪心策略：第一步开始和第二步结束时空闲时间最少

次序：C->D->A->B



六、典型算法

6.1.4 其他案例

1. 分发饼干问题
2. 柠檬水找零问题
3. 最大子数组和问题
4. 跳跃游戏及其增强版
5. 弓箭射气球
6. 迷离傍地走
7. 作业加工问题

作业加工问题

设有n个独立的作业{1,2,3，…,n},由m台相同的机器进行加工处理。作业i所需时间为Ti，现约定，任何作业可以在任何一台机器上加工处理，但未完工前不允许中断处理过程，任何作业不能拆分成更小的子作业。

要求用贪心算法设计思想设计程序，为该调度问题设计出一种作业调度方案，使得所给n个作业在尽量少的时间内由m台机器加工处理完成。

例：有10个任务，3台机器，各任务的完成时间分别为下表所示，计算最短完成时间

1	2	3	4	5	6	7	8	9	10
4	7	12	10	2	3	16	8	6	5

作业加工问题

问题分析

机器相同：不同机器完成同一任务的时间相同

任务独立：不用考虑任务间的衔接完成

如何选择合适的贪心策略?

最短任务优先，让机器尽量并行处理?

最长任务优先，让机器尽量处于工作状态?

作业加工问题

1	2	3	4	5	6	7	8	9	10
4	7	12	10	2	3	16	8	6	5

短作业优先：

序号	第一台机器	第2台机器	第3台机器
1	2	3	4
2	5	6	7
3	8	10	12
4	16		
总时间	31	19	23

作业加工问题

1	2	3	4	5	6	7	8	9	10
4	7	12	10	2	3	16	8	6	5

长作业优先：

序号	第一台机器	第2台机器	第3台机器
1	16	12	10
2	6	7	8
3	3	4	5
4		1	2
总时间	25	24	25

作业加工问题

设有n个独立的作业{1,2,3，…,n}：

n<=m，则所有作业同时处理，最长作业的处理时间是所有作业的处理时间。

n>m，先将作业按照长度非递增排序，然后将前m个作业分配给m台机器，当机器完成当前作业，则从最长队列中取出最长作业分配给该机器，直到所有作业处理完成。

作业加工问题

最长作业优先的贪心选择能够得到全局最优解吗?

2台机器，6个作业分别运行时间：{13,12,7,5,4,1}

根据长作业优先，应该分别安排

13+5+4=22

12+7+1=20

只能得到近似最优解，不能得到全局最优解!

实际最优解:

13+7+1=21

12+5+4=21