# 贪心题目选讲

## 预备知识

- 排序，尤其是多关键字排序(`sort`函数)。
- 优先队列(`priority_queue`)，本质上是一个堆。

```
priority_queue<int> pq; // 按照从大到小的顺序  最大堆
priority_queue<int, vector<int>, greater<int> > pq; // 按照从小到大的顺序  最小堆
priority_queue<pair<int, int> > pq; // 按照first从大到小 second从大到小
priority_queue<pair<int, int>, vector<pair<int,int> >, greater<pair<int, int> > > pq; // 按照first从小到大 second从小到大
```

# 正确性证明

- 能否举出反例
- 最优子结构性质
- 决策包容性
- 反证法
- 微扰(领项交换)
- 范围缩放

# 例题

## 删除问题(COGS 2373)

给定一个正整数 $n$，去掉其中任意 $k$个数字后剩下的数字按原左右次序组成一个新的正整数。请你寻找一种方案使得剩下的数字组成的新数最小。

例如，数字为$175438$，允许删除$4$个数字，删除后组成的新数最小为$13$；如果允许删除$2$个数字，删除后组成的新数最小为$1438$。

- 贪心策略一：最大的数字优先删除(错误)。例如：数字$12324$删除最大的位$4$，结果为$1232$，而实际上应该删除$3$，结果是$1224$。
- 贪心策略二：每一步总是选择一个使剩下的数最小的数字删去，即按高位到低位的顺序枚举，若各位数字递增，则删除最后一个数字；否则删除第一个递减区间的首字符。例如：数字$175438$删除数$7$得到的结果为$15438$，接着删除数$5$得到的结果为$1438$；数字$1234$删除$4$得到的结果为$123$，接着删除数$3$得到的结果为$12$。

## 活动选择(COGS 1151)

学校报告厅是比较繁忙的场所，通常会有$n$个活动要求使用它，而在同一时间内只能有一个活动能使用报告厅。每个活动$i$都有一个使用的开始时间$s_i$和结束时间$f_i$，且$s_i < f_i$，如果选择了活动 $i$，则它在时间区间$[s_i, f_i)$ 内占用该资源。

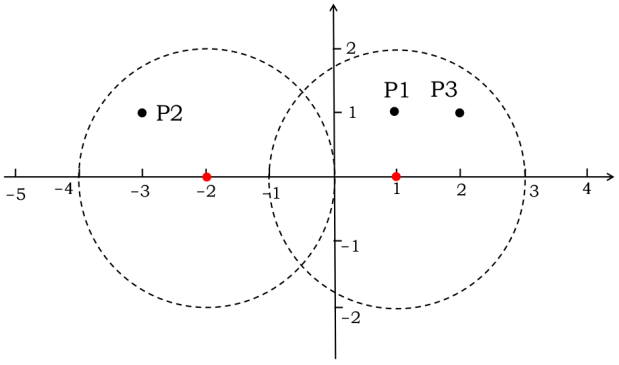
请你设计一个活动安排方案，使得安排的活动数目最多。例如，下表是活动的时间安排，它的最优解为2，可以选择活动1和活动3、活动1和活动4、活动2和活动4等等。

| 活动 | 开始时间 | 结束时间 |
| --- | --- | --- |
| 1 | 1 | 3 |
| 2 | 2 | 5 |
| 3 | 3 | 9 |
| 4 | 6 | 8 |

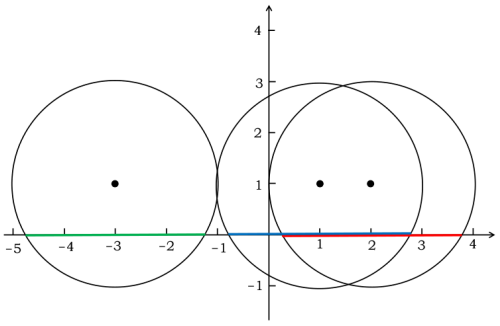
- 贪心策略一：尽可能的选择占用时间短的活动。
- 贪心策略二：尽可能选择开始时间早的活动。
- 贪心策略三：尽可能选择结束时间早的活动。
  -贪心策略三是一个正确的贪心策略。使用该策略每次都选择结束时间早的活动，这样会为未安排的剩余活动留下尽可能多的时间(决策包容性)。算法如下：
  - 将所有活动按照结束时间升序排列；
  - 选择结束时间最早的活动$1$，当前活动$cur$为$1$；
  - 依次扫描后续的每个活动$i$，如果第$i$个活动的开始时间晚于第$cur$个活动的结束时间，则选择第$i$个活动，并将当前活动$cur$设置为$i$，否则放弃选择第$i$个活动。
- 时间复杂度：O($N \log N$)。

# 监控安装(COGS 3187)

校长想通过监控设备覆盖学校内的$n$座建筑物，每座建筑物被视为一个点，在坐标系中给出坐标为$(x, y)$，并且所有的建筑物都在$x$轴的上方。因为学校的供电和传输线路均沿$x$轴，所以监控设备只被允许建立在$x$轴上。每台监控设备的监控范围均为一个半径为$R$的圆，圆心即为监控设备。现在给出$n$座建筑物的坐标，问最少需要几台这样的设备可以实现对所有建筑物的监控?




- 此题从正面想很困难，那么我们换个角度：如果一个建筑物想被监控，那么监控设备必然在以建筑物为圆心半径为$R$的圆内，又因监控设备必须在$x$轴上，所以监控应该在圆与$x$轴相交的弦上。



- 所以，对于每一个建筑物，我们计算出$x$轴上能管辖它的区间$[l, r]$，则原问题转换为：给定$n$个区间，在$x$轴上放置最少的点，使得每个区间包含至少一个点。这就是典型的**区间选点**问题。
- 按照每个区间的右端点$r[i]$从小到大排序($r[i]$相同，则按照$l[i]$从大到小排)，用一个变量$pos$维护最后一台监控设备的位置(初始值为$-10^9$)，依次考虑区间$[l[i], r[i]]$：
  - 如果当前区间左端点$l[i]$大于最后一台监控设备的位置$pos$，则新增一台设备，令$pos = r[i]$。
  - 否则就让一台已经安放过的监控设备来监控，监控数量无需调整。
- 可以用决策包容性来证明。对于每一个区间$[l[i], r[i]]$，我们有两个选择：
  1. 使用已经设置的监控设备。
  2. 新增一台监控设备。
  - 当选择1可行时，不会选择2。选择1之后，未来可以在任意一个位置新增监控设备；而如果选择2，则需要在区间$[l[i], r[i]]$之间新建设备，也就是说第1种选择比第2种选择的坐标范围更广。
  - 其次，如果不得不进行选择2，那么将监控设备放在$r[i]$，未来可能能覆盖后续区间

## 日光浴(COGS 2308)

有$C$头奶牛进行日光浴，第$i$头奶牛需要$minSPF[i]$到$maxSPF[i]$单位强度之间的阳光。

每头奶牛在日光浴前必须涂防晒霜，防晒霜有$L$种，涂上第$i$种之后，身体接收到的阳光强度就会稳定为$SPF[i]$，第$i$种防晒霜有$cover[i]$瓶。

求最多可以满足多少头奶牛进行日光浴。

- 按照 $maxSPF$ 递增的顺序把奶牛排序，依次考虑每头奶牛。
- 对于每头奶牛，扫描一遍所有的防晒霜，在这头奶牛能用(能用指的是该防晒霜的强度符合这头奶牛的范围，并且瓶数还有剩余) 的防晒霜里找 $SPF$ 值最小的使用。
- 以上算法的贪心策略是在满足条件的前提下每次选 $SPF$ 最小的防晒霜。这个策略为什么是正确的呢？每瓶防晒霜是否可用，会被 $minSPF$ 与 $maxSPF$ 两个条件限制。因为奶牛已被按照 $maxSPF$ 递增排序，所以留下较大的防晒霜可能可以让后续奶牛使用(决策包容性)。
- 另外，每头奶牛对答案的贡献至多是1。即使让当前这头奶牛放弃日光浴，留下防晒霜给后面的某一头奶牛用，对答案的贡献也不会变得更大。综上所述，尽量满足当前的奶牛，并选择 SPF 值尽量小的防晒霜是一个正确的贪心策略。

# 合并果子(COGS 75)

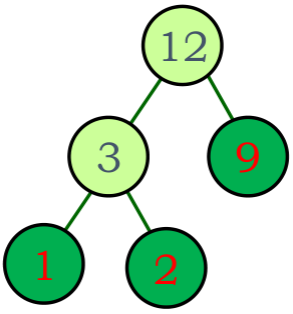
在一个果园里，多多已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。多多决定把所有的果子合成一堆。

每一次合并，多多可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过 $n-1$ 次合并之后，就只剩下一堆了。多多在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以多多在合并果子时要尽可能地节省体力。假定每个果子重量都为 $1$，并且已知果子的种类数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

例如有 $3$ 种果子，数目依次为 $1,2,9$。可以先将 $1,2$ 堆合并，新堆数目为 $3$，耗费体力为 $3$。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 $12$，耗费体力为 $12$。所以多多总共耗费体力为 $3+12=15$。可以证明 $15$ 为最小的体力耗费值。

- 每次合并消耗的体力等于两堆果子的重量之和，所以最终消耗的体力总和就是每堆果子的重量乘上它参与合并的次数。




- 所以想让体力消费少，需要先让重量小先合并，利用果子的重量建立一个最小堆，不断取出堆中最小的两个值，将这两个值累加到答案(消耗的体力)，同时将它们的和插入堆中。
- 当堆的大小为1时，算法结束，输出累加的答案即可。

## 畜栏预定(COGS 3448)

有$N$头牛在畜栏中吃草。每个畜栏在同一时间段只能提供给一头牛吃草，所以可能会需要多个畜栏。给定$N$头牛和每头牛开始吃草的时间A以及结束吃草的时间B，每头牛在[A,B]这一时间段内都会一直吃草。当两头牛的吃草区间存在交集时（包括端点），这两头牛不能被安排在同一个畜栏吃草。求需要的最小畜栏数目和每头牛对应的畜栏方案。

- 首先，按照牛的吃草开始时间从小到大排序。
- 第1头牛肯定要安排一个畜栏，但是对于后续的第$i$头牛，如果前面有畜栏的奶牛吃草结束时间早于这头牛，那么肯定就把这头安排到对应畜栏，否则就需要新开畜栏。
- 所以需要维护当前已经安排好的畜栏，每次需要选择结束时间最早的畜栏，需要使用优先队列维护。

## 超市(COGS 3097)

超市里有$n$件商品，每个商品都有利润$p_i$和过期时间$d_i$,每天只能卖一件商品，过期商品(即当天$d_i \leq 0$)不能再卖。求合理安排每天卖的商品的情况下，可以得到的最大收益是多少。

- 对于所有商品，在商品过期前能卖尽量卖，但是由于每天只能卖一件，而商品又特别多，所以第$t$天时，可能会有很多商品过期不能卖掉。
- 所以，第$t$天时，在保证不卖出过期商品的前提下，尽量卖出利润前$t$大的的商品。
- 把商品按照过期时间从小到大排序，然后依次处理每个商品。
- 为了始终维护利润前若干大的商品，建立一个最小堆来存储售卖的商品(结点为利润)。
- 按照过期时间从小到大的依次处理每个商品：
    - 如果当前商品过期时间$d_i$等于当前堆大小，说明在目前方案中，前$d_i$天已经安排了$d_i$个商品卖出。此时，如果当前商品利润大于堆顶值(已经安排的$d_i$个商品中的最低利润)，那么用当前商品的利润替换堆顶(用当前商品替换掉原方案中利润最低的商品)。
    - 如果当前商品过期时间$d_i$大于当前堆大小，直接将该商品利润入堆(过期前肯定有时间可以卖)。
- 堆内存储的即为要卖出商品的利润，堆中所有元素的和就是答案。

# 国王游戏(COGS 1263)

恰逢 H 国国庆，国王邀请 n 位大臣来玩一个有奖游戏。首先，他让每个大臣在左、右手上面分别写下一个整数，国王自己也在左、右手上各写一个整数。然后，让这 n位大臣排成一排，国王站在队伍的最前面。排好队后，所有的大臣都会获得国王奖赏的若干金币，每位大臣获得的金币数分别是：排在该大臣前面的所有人的左手上的数的乘积除以他自己右手上的数，然后向下取整得到的结果。

国王不希望某一个大臣获得特别多的奖赏，所以他想请你帮他重新安排一下队伍的顺序，使得获得奖赏最多的大臣，所获奖赏尽可能的少。注意，国王的位置始终在队伍的最前面。

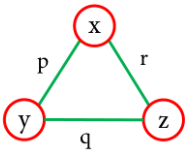
- 按照左右手上的数的乘积升序排序，就是最优排队方案。
- 假设我们已经找到一个最有的顺序，交换两个相邻的大臣$i$和$i+1$，交换之前者两位大臣的奖励分别为：$\frac{1}{b[i]} \times \prod_{j=1}^{i-1} a[j]$　　$\frac{1}{b[i+1]} \times \prod_{j=1}^{i} a[j]$;
- 交换之后两位大臣的奖励分别为：$\frac{1}{b[i+1]} \times \prod_{j=1}^{i-1} a[j]$　　$\frac{a[i+1]}{b[i]} \times \prod_{j=1}^{i-1} a[j]$;
- 其他大臣的奖励显然都不变，因此我们只需要比较上面两组式子的最大值的大小，我们希望交换之前的值较小，即
  $\max\left\{\frac{1}{b[i]} \times \prod_{j=1}^{i-1} a[j], \frac{1}{b[i+1]} \times \prod_{j=1}^{i} a[j]\right\} \leq \max\left\{\frac{1}{b[i+1]} \times \prod_{j=1}^{i-1} a[j], \frac{a[i+1]}{b[i]} \times \prod_{j=1}^{i-1} a[j]\right\}$;
- 上式两边同时消去公因式$\prod_{j=1}^{i-1} a[j]$得到：$\max\left\{\frac{1}{b[i]}, \frac{a[i]}{b[i+1]}\right\} \leq max\left\{\frac{1}{b[i+1]}, \frac{a[i+1]}{b[i]}\right\}$;
- 两边同时乘以$b[i]b[i+1]$得到：$\max\{b[i+1], a[i]b[i]\} \leq \max\{b[i], a[i+1]b[i+1]\}$;
- 因为大臣手上的都是正整数，故$b[i+1] \leq a[i+1]b[i+1]$且$a[i]b[i] \geq b[i]$，所以若想让上述不等式成立则必须要求$a[i]b[i] \leq a[i+1]b[i+1]$。
  - 假设$a[i]b[i] > a[i+1]b[i+1]$，则：$a[i]b[i] \geq b[i] \Rightarrow a[i]b[i] > \max\{b[i], a[i+1]b[i+1]\} \Rightarrow \max\{b[i+1], a[i]b[i]\} \geq a[i]b[i] \Rightarrow \max\{b[i+1], a[i]b[i]\} > \max\{b[i], a[i+1]b[i+1]\}$
  - 上述结论与已知$\max\{b[i+1], a[i]b[i]\} \leq \max\{b[i], a[i+1]b[i+1]\}$矛盾，故而假设错误，即$a[i]b[i] \leq a[i+1]b[i+1]$。

# DZY Loves Physics(COGS 3201)

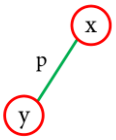
给定一个$n$个点$m$条边的无向图，其中每一个点和边都有权值，求一个点数大于等于2的导出连通子图，使得该子图中点权和除以边权和最大。求这个最大的边权和。

- 我们首先来看一个由3个点3条边组成的图，除了它本身作为一个连通子图，还有6个连通子图：

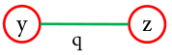
  图，还有 6 个连通子图：

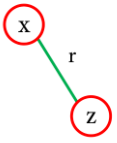
  图：$\frac{x+y+z}{p+q+r}$

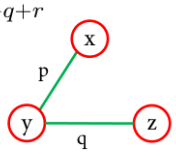
  图：$\frac{x+y}{p}$

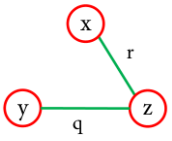
  图：$\frac{y+z}{q}$

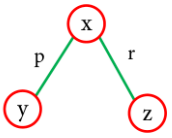
  图：$\frac{x+z}{r}$

  图：$\frac{x+y+z}{p+q}$

  图：$\frac{x+y+z}{q+r}$

  图：$\frac{x+y+z}{p+r}$

- 这几个连通子图，哪一个的值最大？
  - 很容易得出$\frac{x+y+z}{p+q+r}$不是最大的。
  - 经过证明$\frac{x+y+z}{p+q}$、$\frac{x+y+z}{q+r}$、$\frac{x+y+z}{p+r}$不是最大的。
  - 所以最大的在$\frac{x+y}{p}$、$\frac{y+z}{q}$、$\frac{x+z}{r}$中，即所求的子图必然由两个点和一条边构成。
  - 从上面可以得出，子图中有三个点不如两个点的值大。推广到一般情况，子图中有$k$个点不如有2个点的值大，所以对于一般情况，所求的子图必然由两个点和一条边构成。
- 若要证明$\frac{x+y+z}{p+q}$、$\frac{x+y+z}{q+r}$、$\frac{x+y+z}{p+r}$不是最大的，我们只需要证明：$\max\{\frac{x+y}{p},\frac{y+z}{q}\}>\frac{x+y+z}{p+q}$，即可证明$\frac{x+y+z}{p+q}$肯定不是最大的，同理可证$\frac{x+y+z}{q+r}$、$\frac{x+y+z}{p+r}$不是最大的。假设$\frac{x+y}{p}\leq\frac{x+y+z}{p+q},\frac{y+z}{q}\leq\frac{x+y+z}{p+q}$，则
  - $(x+y)(p+q)\leq p(x+y+z),(y+z)(p+q)\leq q(x+y+z)$
  - $qx+qy\leq pz,py+pz\leq qx$
  - $py+qx+qy\leq py+pz<qx$
  - $py+qy\leq 0$
    上述结论与点和边的权值均为正整数矛盾，故而假设错误，即$\max\{\frac{x+y}{p},\frac{y+x}{q}\}>\frac{x+y+z}{p+q}$是正确的，即$\frac{x+y+z}{p+q}$不是最大的。

# 树染色(COGS 3429)

一颗树有 $n$ 个结点，这些结点被标号为：$1, 2, 3 \dots n$，每个结点 $i$ 都有一个权值 $A[i]$。

现在要把这棵树的结点全部染色，染色的规则是：

- 根结点$R$可以随时被染色；对于其他结点，在被染色之前它的父亲结点必须已经染上了色。
- 每次染色的代价为$T * A[i]$，其中$T$代表当前是第几次染色。

求把这棵树染色的最小总代价。

- 贪心策略1：每一步在可以被染色的点里选权值最大的染色(错误)。
- 反例：构造一棵树，让一个权值很小的结点下边有很多权值巨大的结点，另一个权值较大的结点却没有子结点。
- 不过从这个错误的贪心算法的思考中，我们可以提取出一个正确的性质：树中除根结点外权值最大的点，一定会在它的父结点被染色后立即染色。
- 于是我们可以确定的是，树中权值最大的点及其父结点的染色操作是连续进行的，我们可以把这两个点“合并起来”。合并得到的新点的权值设为这两个点的权值的平均值。
- 例如有权值为 $x, y, z$ 的三个点，我们已知$x$和 $y$ 的染色操作是连续进行的，那么就有两种可能的染色方案：
  - 先染$x, y$，再染$z$，代价是$x + 2y + 3z$。
  - 先染$z$，再染$x, y$，代价是$z + 2x + 3y$。
- 我们主要关心这两个代价之间的大小关系，所以不妨把两个式子同时加上 (z-y)
  再除以 2, 分别得到：
  - $\frac{x+y}{2} + 2z$
  - $z + 2\frac{x+y}{2}$
- 这恰好就相当于有权值为 $\frac{x+y}{2}$ 和 $z$ 的两个点的两种染色次序。换言之，下列两种情况的“最优染色次序”可以互相转化：
  - 权值为 $x, y, z$ 的三个点。
  - 权值为 $\frac{x+y}{2}$ 和 z 的两个点。
- 进一步推进，我们可以得到一种“等效权值”的算法：记录每个点是由多少个点合并而成的，一个点的“等效权值”定义为：

$$\text{该点包含的原始权值总和} \div \text{该点包含的原始点数}$$

- 根据一开始提到的性质，我们不断在树中取“等效权值”最大的点 $p$, 与其父结点 $fa$ 合并。合并之前 $p$ 与$fa$各自包含的点的染色顺序是已知的，我们就让 $p$ 中第一个点排在 $fa$ 中最后一个点之后紧接着被染色，把这个顺序保存在 $p$ 与 $fa$ 合并以后的点上。最终整棵树合并成一个点后，我们就按照这个点内保存的顺序在原始的树上把各个结点依次染色，计算出花费的总代价，即为所求。

# 练习题

- 删数问题【NOIP1994】(SYOI)(COGS 2373)
- 活动选择(COGS 1151)
- 拦截导弹(弱化版)(SYOI)(COGS 3191)
- 拼数【NOIP1998】(COGS 3767)
- 旅行家的预算【NOIP1999】(COGS 1160)
- 监控安装(COGS 3187)
- 超速检测【CSP2024S】(COGS 4054)
- 种树(COGS 3193)
- 喷水装置(COGS 3194)
- 日光浴(COGS 2380)
- 合并果子(COGS 75)
- 畜栏预定(COGS 3448)
- 超市(COGS 3097)
- 廊桥分配【CSP2021S】(COGS 3619)
- 种树【国家集训 队2011】(COGS 1862)
- 任务(COGS 2231)
- 国王游戏【NOIP 2012】(COGS 1263)
- 叠罗汉(COGS 3202)
- DZY Loves Physics(COGS 3201)
- 树染色(COGS 3429)