骗分导论

diversion

雅礼中学

October 31, 2018

写在前面的话

本篇提供的方法仅限于在实在不会做的情况下使用，如果你会写正解或者部分分，请认真拿分。

初级

该题目有无解情况，需要输出-1或No，直接输出，一般有10分，可能更多。

该题目有无解情况，需要输出-1或No，直接输出，一般有10分，可能更多。

比如noip2012文化之旅,该题目需要你输出在满足题目限制条件下的最短路径，由于存在无解情况，输出-1即可得到10分。

如何拿到题目所给的暴力分?

如何拿到题目所给的暴力分?

如果该题要你维护一些信息，直接暴力修改暴力查询即可

如何拿到题目所给的暴力分?

如果该题要你维护一些信息，直接暴力修改暴力查询即可

如何优化?

如何拿到题目所给的暴力分?

如果该题要你维护一些信息，直接暴力修改暴力查询即可

如何优化?

有时候为了卡裸暴力，出题人会多次询问较大的区间，我们可以预先把这种区间的答案算出啦，然后O(1)回答

如何拿到题目所给的暴力分?

如果该题要你维护一些信息，直接暴力修改暴力查询即可

如何优化?

有时候为了卡裸暴力，出题人会多次询问较大的区间，我们可以预先把这种区间的答案算出啦，然后O(1)回答

可以参考雅礼集训day2蔬菜，该题要求每次询问一个矩形内的答案。

某位不愿意透露姓名的韩姓同学在一开始算出了整个矩形的答案，然后O（1）回答便A掉了此题。

某位不愿意透露姓名的韩姓同学在一开始算出了整个矩形的答案，然后O（1）回答便A掉了此题。

我们考虑如何改进这种方法。

某位不愿意透露姓名的韩姓同学在一开始算出了整个矩形的答案，然后O（1）回答便A掉了此题。

我们考虑如何改进这种方法。

其实这种方法本质上就是记忆化，所以我们可以将暴力算过的答案记下来，多次询问时便可以O（1）回答。

对于一类计数问题和最优化问题，拿到暴力分一般只需要暴力枚举方案即可。

对于一类计数问题和最优化问题，拿到暴力分一般只需要暴力枚举方案即可。

我们考虑优化dfs，减少递归次数。

对于一类计数问题和最优化问题，拿到暴力分一般只需要暴力枚举方案即可。

我们考虑优化dfs，减少递归次数。

对于计数问题，如果在递归到一定层数后，你可以O（1）计算后面的答案，或者判断后面无解就立刻退出。

对于一类计数问题和最优化问题，拿到暴力分一般只需要暴力枚举方案即可。

我们考虑优化dfs，减少递归次数。

对于计数问题，如果在递归到一定层数后，你可以O（1）计算后面的答案，或者判断后面无解就立刻退出。

当然，如果你写的是更高级一点的方法，如枚举其中的某几项，那前面枚举的项可能会对后面的项产生约束，可以大大降低你枚举的数量。

对于一类计数问题和最优化问题，拿到暴力分一般只需要暴力枚举方案即可。

我们考虑优化dfs，减少递归次数。

对于计数问题，如果在递归到一定层数后，你可以O（1）计算后面的答案，或者判断后面无解就立刻退出。

当然，如果你写的是更高级一点的方法，如枚举其中的某几项，那前面枚举的项可能会对后面的项产生约束，可以大大降低你枚举的数量。

例如做数论题时，经常枚举每个数的倍数算答案，复杂度ln(n)。而不是枚举每个数的约数，因为这样$n\sqrt{n}$的。

对于一类最优化问题，我们可以加入很多最优化剪枝和可行性剪枝。

对于一类最优化问题，我们可以加入很多最优化剪枝和可行性剪枝。

最简单的，如果当前答案已经不可能比全局答案更优了，就可以直接退出。

例题：小木棍

题目描述

乔治有一些同样长的小木棍，他把这些木棍随意砍成几段，直到每段的长都不超过50。现在，他想把小木棍拼接成原来的样子，但是却忘记了自己开始时有多少根木棍和它们的长度。给出每段小木棍的长度，编程帮他找出原始木棍的最小可能长度。

数据范围 $n <= 65$,每个小木棍长度 $<= 50$

这道题貌似没有什么别的做法，只能搜索剪枝。

这道题貌似没有什么别的做法，只能搜索剪枝。

剪枝1:首先我们枚举答案，因为题目要求长度最小，所以我们从小到大枚举。很显然，答案只会是木棍长度总和的约数，并且大于等于最长的小木棍的长度。

这道题貌似没有什么别的做法，只能搜索剪枝。

剪枝1:首先我们枚举答案，因为题目要求长度最小，所以我们从小到大枚举。很显然，答案只会是木棍长度总和的约数，并且大于等于最长的小木棍的长度。

剪枝2:在枚举使用哪根小木棍时，我们从大到小枚举，因为放进去的木棍越长，剩下的部分就越短，受到的限制越大。

这道题貌似没有什么别的做法，只能搜索剪枝。

剪枝1:首先我们枚举答案，因为题目要求长度最小，所以我们从小到大枚举。很显然，答案只会是木棍长度总和的约数，并且大于等于最长的小木棍的长度。

剪枝2:在枚举使用哪根小木棍时，我们从大到小枚举，因为放进去的木棍越长，剩下的部分就越短，受到的限制越大。

剪枝3:如果当前长度加上后面所有能用的都不够，就可以直接退出。

剪枝4:在一开始，我们可以先做一遍背包，求出哪些长度可以凑出来，如果当前需要的长度凑不出来就可以返回了。

剪枝4:在一开始，我们可以先做一遍背包，求出哪些长度可以凑出来，如果当前需要的长度凑不出来就可以返回了。

剪枝5:在凑出某一根木棍后，对于下一个放进去的木棍，我们不需要枚举，直接找当前最大的放进去。

剪枝4:在一开始，我们可以先做一遍背包，求出哪些长度可以凑出来，如果当前需要的长度凑不出来就可以返回了。

剪枝5:在凑出某一根木棍后，对于下一个放进去的木棍，我们不需要枚举，直接找当前最大的放进去。

剪枝6:如果搜索到一个长度为a的木棍不行，那么可以跳过所有和它长度相同的木棍。

无论复杂度是多少都不会TLE的终极禁忌大法

无论复杂度是多少都不会TLE的终极禁忌大法

$$if(clock()/CLOCKS\_PER\_SEC > 0.95)return;$$

无论复杂度是多少都不会TLE的终极禁忌大法

$if(clock()/CLOCKS\_PER\_SEC > 0.95)return;$

比如国庆集训day4t2,写出一个70分的暴力，在加上此剪枝便可以获得95分的好成绩。

中级

分块

有时候我们需要支持修改和查询区间信息，但是你发现你无法用正常的数据结构来做，或者是你不会，分块可能可以帮助你拿到许多分甚至A掉这题。

分块

有时候我们需要支持修改和查询区间信息，但是你发现你无法用正常的数据结构来做，或者是你不会，分块可能可以帮助你拿到许多分甚至A掉这题。

分块的大致思想就是将整个序列分成S块，每个块内暴力计算整个块的答案，查询时，边角暴力查询，整块的一起查询。一般，修改是O（S）的，查询是O（$\frac{N}{S}$）的，S取$\sqrt{N}$时最优。

例题1

题目描述

已知一个数列，你需要进行下面三种操作：

1.将某区间每一个数乘上x

2.将某区间每一个数加上x

3.求出某区间每一个数的和

这显然可以用线段树做，但我们考虑用分块来解决。

这显然可以用线段树做，但我们考虑用分块来解决。

跟在线段树上差不多，我们对每个块维护块的和，加法标记和乘法标记，直接查询即可。

例题2

给你一个长度为n的序列a，m次询问，每次询问区间众数。

$ai <= 1e9 \quad n, m <= 5e4$

例题2

给你一个长度为n的序列a，m次询问，每次询问区间众数。

$ai <= 1e9 \quad n, m <= 5e4$

此题无法用线段树做的原因就是我们无法快速合并两个区间的信息。

例题2

给你一个长度为n的序列a，m次询问，每次询问区间众数。

$ai <= 1e9 \quad n, m <= 5e4$

此题无法用线段树做的原因就是我们无法快速合并两个区间的信息。

所以我们在分块的时候可以处理出每两个块之间的答案。

例题2

给你一个长度为n的序列a，m次询问，每次询问区间众数。

$ai <= 1e9 \quad n, m <= 5e4$

此题无法用线段树做的原因就是我们无法快速合并两个区间的信息。

所以我们在分块的时候可以处理出每两个块之间的答案。

对于边角料，我们记个前缀和，表示前i个块，j出现了几次，然后我们暴力扫描边角料，并更新答案。

莫队，一种解决序列上多次询问的暴力算法。

莫队，一种解决序列上多次询问的暴力算法。

具体来说就是把询问按左端点算在的块编号排序，然后用一左一右两个指针在序列上滑动，来计算答案。

莫队，一种解决序列上多次询问的暴力算法。

具体来说就是把询问按左端点算在的块编号排序，然后用一左一右两个指针在序列上滑动，来计算答案。

拓展应用，树上莫队，二维莫队，回滚莫队。

其他一些小trick

其他一些小trick

对于某些计数题，可以暴力打出前几项找规律。

其他一些小trick

对于某些计数题，可以暴力打出前几项找规律。

如果不会做，可以打一些表，多的一档部分分。

其他一些小trick

对于某些计数题，可以暴力打出前几项找规律。

如果不会做，可以打一些表，多的一档部分分。

贪心题不会做，就多写几个不会比答案优的贪心一起取max。

其他一些小trick

对于某些计数题，可以暴力打出前几项找规律。

如果不会做，可以打一些表，多的一档部分分。

贪心题不会做，就多写几个不会比答案优的贪心一起取max。

大胆猜测结论

其他一些小trick

对于某些计数题，可以暴力打出前几项找规律。

如果不会做，可以打一些表，多的一档部分分。

贪心题不会做，就多写几个不会比答案优的贪心一起取max。

大胆猜测结论

用树链剖分求LCA比倍增要快

其他一些小trick

对于某些计数题，可以暴力打出前几项找规律。

如果不会做，可以打一些表，多的一档部分分。

贪心题不会做，就多写几个不会比答案优的贪心一起取max。

大胆猜测结论

用树链剖分求LCA比倍增要快

尽量访问连续内存，别问我这是为什么。

高级

对于数据范围在大概1000以内的最优化问题，我们可以采用爬山，模拟退火等方法获得大量分数。

对于数据范围在大概1000以内的最优化问题，我们可以采用爬山，模拟退火等方法获得大量分数。

爬山，多次随机撒点，每次贪心向能使答案变优的地方走。

对于数据范围在大概1000以内的最优化问题，我们可以采用爬山，模拟退火等方法获得大量分数。

爬山，多次随机撒点，每次贪心向能使答案变优的地方走。

模拟退火，爬山算法的局限性在于很可能陷入局部最优解。

对于数据范围在大概1000以内的最优化问题，我们可以采用爬山，模拟退火等方法获得大量分数。

爬山，多次随机撒点，每次贪心向能使答案变优的地方走。

模拟退火，爬山算法的局限性在于很可能陷入局部最优解。

所以我们引入温度参数，我们设计一个跟温度和答案有关的函数，每次有机率接受一个比当前答案要劣的解，因为这样更可能找到全局最优解。

对于数据范围在大概1000以内的最优化问题，我们可以采用爬山，模拟退火等方法获得大量分数。

爬山，多次随机撒点，每次贪心向能使答案变优的地方走。

模拟退火，爬山算法的局限性在于很可能陷入局部最优解。

所以我们引入温度参数，我们设计一个跟温度和答案有关的函数，每次有机率接受一个比当前答案要劣的解，因为这样更可能找到全局最优解。

温度参数会随着迭代次数的增多而下降。也就是说我们接受更劣解的概率会变小。

吊打XXX

题面描述：

有n个重物，每个重物系在一条足够长的绳子上。每条绳子自上而下穿过桌面上的洞，然后系在一起。假设绳子是完全弹性的（不会造成能量损失），桌子足够高（因而重物不会垂到地上），且忽略所有的摩擦。问绳结X最终平衡于何处。 注意：桌面上的洞都比绳结X小得多，所以即使某个重物特别重，绳结X也不可能穿过桌面上的洞掉下来，最多是卡在某个洞口处。

$n \leq 1000$

吊打XXX

每次将所有的力都正交分解，算出合力，再合成一个力。

吊打XXX

每次将所有的力都正交分解，算出合力，再合成一个力。

向力的方向走一段，直到某一次合力小于某个精度。

经典问题

给定一个n个点m条边的图，求这个图的最大团。 $n, m/le1000$

经典问题

由于最大团的条件非常苛刻，大部分点是不影响答案。

经典问题

由于最大团的条件非常苛刻，大部分点是不影响答案。

每次随机一个序列，然后按顺序加入答案集合，暴力check即可。

[BJOI2014]想法

题目描述

小强和阿米巴是好朋友。小强要出一套题目。

他的题目以涉及面广（偏）、考察深入（怪）、思维强度大（难）著称。他为了出题，一共攒了M个本质不同的想法，每个想法形成了一个题目。不过，他觉得拿这些题目去考察选手会把比赛搞的太过变态，所以，想请阿米巴来帮忙调整一下他的题目。

阿米巴指出，为了让一场考试的题目的考察点尽量全面，有一个通用的做法叫做“组合”。如果把两个题目A和B组合在一起，那么组合而成的题目涉及到的想法的集合就是A涉及到的想法的集合和B涉及到的想法的集合的并。并且，题目是可以反复组合的。

[BJOI2014]想法

例如，小强现在有三个想法1,2,3，分别对应了题目P1,P2,P3。
现在，小强把P1和P2组合得到P4。P4涉及的想法的集合是1,2。
之后，小强把P2和P3组合得到P5。P5涉及的想法的集合是2,3。
最后，小强把P4和P5组合得到P6。P6涉及的想法的集合是1,2,3。
现在，小强告诉你每个题目都是如何组合而来的。你要回答的就是，每个题目涉及的想法的集合有多大。
不过，这个问题是很难的。于是，你只需要能够以比较高的概率回答的比较准确即可。
$M \leq 100000, N \leq 1000000$

[BJOI2014]想法

【评分方法】 对于每个输出文件，如果其中你有95%以上的行的答案和正确答案的误差不超过25%，那么你就可以得到分数。所谓误差不超过25%，即，如果正确答案是X，那么你的答案在[0.8X,1.25X]这个闭区间内。

[BJOI2014]想法

对于每个想法$Random$一个随机器代表它，合并的时候保留前T小的数,并记录其出现次数。最后算答案时，如果出现种类少于T就直接加上去，否则我们估计他的大小为$T * RAND_MAX/RAND_T$，其中$RAND_T$是第T小的$RAND$ 为什么这样可以呢?

[BJOI2014]想法

对于每个想法$Random$一个随机器代表它，合并的时候保留前T小的数,并记录其出现次数。最后算答案时，如果出现种类少于T就直接加上去，否则我们估计他的大小为$T*RAND_MAX/RAND_T$，其中$RAND_T$是第T小的$RAND$ 为什么这样可以呢?

设包含的总想法数为L，在L中取到T的期望为T/L，而随机数是均匀分布的，所以T/L会等于$RAND_T/RAND_MAX$，所以$L=T*RAND_MAX/RAND_T$

[BJOI2014]想法

对于每个想法$Random$一个随机器代表它，合并的时候保留前T小的数,并记录其出现次数。最后算答案时，如果出现种类少于T就直接加上去，否则我们估计他的大小为$T*RAND_MAX/RAND_T$，其中$RAND_T$是第T小的$RAND$ 为什么这样可以呢?

设包含的总想法数为L，在L中取到T的期望为T/L，而随机数是均匀分布的，所以T/L会等于$RAND_T/RAND_MAX$，所以$L=T*RAND_MAX/RAND_T$

多随机几次取平均值就行了。

常数优化

常数优化

访问连续内存。

常数优化

访问连续内存。

调用数组其实有一个寻址的过程，但如果你连续调用连续一段下表，寻址的过程将会变的很快。

常数优化

访问连续内存。

调用数组其实有一个寻址的过程，但如果你连续调用连续一段下表，寻址的过程将会变的很快。

常见方法有:

常数优化

常数优化

调整循环顺序,让最里面的循环调用最后一层下表，如矩阵乘法，状压dp，倍增等。

常数优化

调整循环顺序,让最里面的循环调用最后一层下表，如矩阵乘法，状压dp，倍增等。

重新编号，例如把一条斜线标号成连续一段。

常数优化

调整循环顺序,让最里面的循环调用最后一层下表，如矩阵乘法，状压dp，倍增等。

重新编号，例如把一条斜线标号成连续一段。

减少递归调用，有些题目保证树上$fa[i] < i$，那么我们树形dp时可以从编号大的向编号小的转移，或者要多次树形dp时，可以在dfn上转移。

常数优化

调整循环顺序,让最里面的循环调用最后一层下表，如矩阵乘法，状压dp，倍增等。

重新编号，例如把一条斜线标号成连续一段。

减少递归调用，有些题目保证树上$fa[i] < i$，那么我们树形dp时可以从编号大的向编号小的转移，或者要多次树形dp时，可以在dfn上转移。

能用树状数组就不要用线段树。

常数优化

调整循环顺序,让最里面的循环调用最后一层下表，如矩阵乘法，状压dp，倍增等。

重新编号，例如把一条斜线标号成连续一段。

减少递归调用，有些题目保证树上$fa[i] < i$，那么我们树形dp时可以从编号大的向编号小的转移，或者要多次树形dp时，可以在dfn上转移。

能用树状数组就不要用线段树。

对于模数在int范围内的运算，尽量不用$longlong$，必要时甚至可以用$short, char$来加速。

可能有用的优化

可能有用的优化

使用$getchar_unlocked()$来代替$getchar()$，在$Linux$下会快一些(~~可能吧,debug讲的~~)。

可能有用的优化

使用$getchar_unlocked()$来代替$getchar()$，在$Linux$下会快一些(~~可能吧,debug讲的~~)。

循环展开可能会有意想不到的收获,推荐展开4层。

可能有用的优化

使用$getchar_unlocked()$来代替$getchar()$，在$Linux$下会快一些(~~可能吧,debug讲的~~)。

循环展开可能会有意想不到的收获,推荐展开4层。

指针调用数组。

Thanks