

Peter 笔记

Peterlits Zo

2020 年 5 月 31 日

目录

1	----- 分割线 -----	5
2	cf's 4A problem	5
3	Python 的编码问题	5
4	matplotlib 的字体输出	6
5	matplotlib 的 cmap	6
6	使用 TikZ 创建条件图	7
7	L ^A T _E X 中的对齐问题	7
8	Python 中的 setup.py	7
9	python 的 pyyaml 问题	8
10	cf's 71A problem	8
11	cf's 118A problem	8
12	cf's 85A problem	9
13	关于 C++ 中的 using 和 typedef	10
14	Python 中的 __new__ 方法	10
15	Python 中的 hash	11

目录	2
16 cf's 1288A problem	11
17 cf's 1288B problem	12
17.1 后记	13
18 cf's 1288C problem	13
18.1 关于排列组合	13
19 XeTeX 的命令行乱码	14
20 cf's 1285B problem	14
20.1 segmentation fault	14
20.2 max	15
20.3 accumulate	15
20.4 最后，这道题的答案	15
21 cf's 1285C problem	16
21.1 关于题面	16
21.2 其他答案	16
22 关于 undefined reference to 'WinMain'	16
23 cf's 1294B problem	17
23.1 little bug	17
23.2 关于其他人的答案	17
24 cf's 1294C problem	18
25 cf's 1293A problem	18
25.1 其他代码	19
26 cf's 1292 problem	19
26.1 他人答案	19
27 cf's 1294D problem	19
27.1 他人答案	20
28 puts, printf and fwrite	20
29 cf's 1295B problem	20
29.1 最小同余正整数	21
29.2 寻找元素	21
29.3 整数同号	21
30 cf's 1296A problem	22

目录	3
31 cf's 1303A problem	22
31.1 我的答案	22
31.2 其他人的答案 - 1	22
31.3 其他人的答案 - 2	23
32 cf's 1303B problem	23
32.1 关于long long和scanf	23
32.2 其他人的答案	24
33 cf's 1303C problem	24
33.1 其他人的答案	24
34 cf's 1304C problem	25
35 cf's 1304 problem	26
36 cf's 1296C problem	26
36.1 其他人的答案	27
37 cf's 1307B problem	27
38 cf's 1307C problem	28
39 MIT 6.006 part1(unit1)	28
40 cf's 1296D problem	29
41 一道数学题	30
41.1 中值不成立的原因	30
42 cf's 1299A problem	31
43 cf's 1321B problem	32
44 python 的多项式拟合	32
45 在 wsl 中编写含 CJK 字符的文章	33
46 L^AT_EX 的旋转表格	33
47 令 cmd 接受参数	34
48 在 wsl 中启动 jupyter-notebook	34
49 latex 关于\end{...} 的报错问题	35
50 cf's 1321C problem	35

目录	4
51 cf's 1316B problem	35
52 shell 脚本中的单引号和双引号	36
53 cf's 1312C problem	36
53.1 注意上限	36
53.2 关于本应该输出整数的浮点算法	37
54 链表的使用	37
54.1 动态的分法	37
55 cf's 1324D problem	38
56 关于 map<bitset> 的使用	38

SECTION 1

----- 分割线 -----

以下是原笔记内容，原来是用 python 做的一个小程序来管理的。

以后应该会直接用 \LaTeX 来做笔记吧。和之前自动生成的不一样，之后的文件，会保持最新的文件在最上面。

SECTION 2

cf's 4A problem

2020-01-12 09:19:39.244631

divide a even ingeter into two parts, each of them is even

这说明了这个整数能被 2 整除（因为 $\text{even} + \text{even} = \text{even}$ ），且不能为 2.

SECTION 3

Python 的编码问题

2020-01-12 12:17:27.624951

尝试用 `print('...', file=out_put_file)` 来进行文本输出，发现输出的文本不是用 UTF-8 来编码的，而是好像用的国标（国家的标准？）

在 `str` 中字符是用 `unicode` 来编码的，是没有被 `encode` 的二进制数据，输出到文件，输出的内容编码是由系统的配置决定的。

猜想：open 时会指定 `encoding` 可以解决这个问题。

open 时指定 `encoding` 会有带有编码信息的属性的文本对象。因为有状态的影响，所以该对象的 `write method` 会用编码下的二进制实现去写文件。

而 `print` 可以指定 `file`，但是它（应该）不会读取文件内部的编码状态，（这个时候我想起来了，有时用 `u8` 编码的时候好像会在文件头留下一个特别的标记？）而是直接用系统指定的编码格式去输出二进制数据去写数据。

猜想：`print` 是把文件输出到 `sys.stdout` 上（应该，至少意思是这样）。输出的环境是由系统的配置/环境决定的，为了环境能够正常显示文本数据，就应该会用配置下的文件编码格式来编码 `Unicode` 的数据流，这也是为什么它不会去读取文本文件的编码状态的原因吧？（甚至还没有参

数可以调节输出的编码状态，这样)

SECTION 4

matplotlib 的字体输出

2020-01-12 12:19:30.764825

这个问题我之前也记在白皮书上了的，但是在写一点也 ok。

matplotlib以什么字体输出是根据`cont.family`,`font.serif`,`font.sans-serif`等决定的。(text .usetex = True是指排版用 T_EX，字体用配置指定的，还是全部由 T_EX 自己决定的?)

`font.family`指定输出的字体的字系，有诸如`serif`可以选择。

一般`serif`，中文字体感觉不错的有经典的`SimSun`(宋体)，而`sans-serif`，用于编程的`Consolas`也不错。

关于修改配置，可以在文件中修改，或者用`plt.rcParams['xxxx.yyyy'] = data`来修改的。(当然可能有其他的修改输出格式的办法)

在`axes.texta`中，用`family`参数可以临时指定字系。应该也可能有其他的方法可以。

SECTION 5

matplotlib 的 cmap

2020-01-12 12:20:10.551049

之前网上的示例用到了 ‘RdYlGn’ 的 `cmap`，通过`plt.get_cmap`来获取的。之前以为`RdYlGn`的中间是白的，以为它好像是：*Red* → *Orange* → *White* → *Lightgreen* → *Green* 的样子。结果看了才知道并不是这样子的，它中间反而是黄的，是`Yl`的样子，有一说一，确实，这也太白了一点。

也有颜色表，比如`tab20c`，一格一格的。

不是很确定，返回的 `cmap` 对象可能调用一个元素为 `int` 或者 `float` 的可迭代对象，然后返回对应的颜色列表。

SECTION 6

使用 TikZ 创建条件图

2020-01-12 12:20:53.074716

我最近非常想要设计一个语法统一优美的作图语言，我简直被python宠坏了。

唉.....

先提一句: `\tikz\fill[orange](1ex, 1ex)circle (1ex);` 可以画个圆。(这个TikZ太麻烦了。

而`\tikz \draw[->] begin -- end;`可以用来画箭头。

今天在知乎上面逛了一圈，有人说可以用python来搞个 UML 图，试试就试试。

p.s. 我现在用的是graphviz的dot语言。还可以吧，但是总是有点麻烦。语法格式也很不统一。

SECTION 7

L^AT_EX 中的对齐问题

2020-01-12 12:21:33.898432

L^AT_EX 中有两种对齐方式：(1) 环境对齐，(2) 命令对齐。其中环境的话是上下都空了个间隔，用命令会好一点。但怎么说呢，命令的话会作用到一个`\par`。

注：VsCode 中用`\$number`来表示捕获的分组。

SECTION 8

Python 中的 setup.py

2020-01-12 12:22:21.429750

通过 setup.py 文件，可以使用命令`python setup.py install`来安装包。(注，要在 setup.py 目录下运行才会 ok，也就是说，cwd是 setup.py 的父目录)

也可以使用`setuptools.setup(...)`的那啥来指定外部名，之后就可以在外面直接用了。

SECTION 9

python 的 pyyaml 问题

2020-01-12 12:22:55.901702

直接用 pyyaml 会有点问题。现在我是在用 ruamel.yaml 的第三方包。

效果不错，但是还有要改进的地方。我想去参与开发这个项目。有一说一，这个项目还不错的说。

SECTION 10

cf's 71A problem

2020-01-12 09:50:00.711916

在 C++ 中因为对象没有特殊方法（当然，构造方法和解构方法除外，还有运算符重载，如果这也算的话）

所以说并不是所有对象都可以转化为字符串的。从另一个方面来讲，如果过于底层的话，的确不需要用到这种方法（比如 Python 的 `__str__` 方法）。

（话说，我是在说服自己吗??）

对于基本的数据，`std::string` 提供了特殊的方法，在 `std` 的命名空间里，提供了方法 `std::to_string(number_type)`，

另外，如果输出要换行的话，也不能忘掉。

p.s. 在上一题中，我也要时刻注意输入数据的范围。

SECTION 11

cf's 118A problem

2020-01-12 15:35:41.865311

遇到了几个有意思的问题。

首先是如何把字符串转换为小写的字符串。在 python 中只需要使用 `str.lower()` 就可以得到一个拷贝了。（谢天谢地，我现在特别怀念 python）但是很明显这在 C++ 中时行不通的。

第一种转换方法是使用 `transform` 函数再配上一个 `lambda` 函数。`transform` 是由标准库 `algo-`

rithm 提供的，如关于它的介绍网站¹说的那样，它的几个声明之一是：

```
template< class InputIt, class OutputIt, class UnaryOperation >
OutputIt transform( InputIt first1, InputIt last1, OutputIt d_first,
    UnaryOperation unary_op );
```

它会把 unary_op 作用到这些 $[first1, last1)$ 上去，而输出到 $[first2, +\infty)$ 上去。
而 lambda 函数，最简单的形式是这两种：

```
[ captures ] ( params ) { body }
[ captures ] { body }
```

回到主题来，这个函数就是为了封装住来自 ctype 的函数 `std::tolower` 的函数原型是 `int tolower(int ch);`

以上是第一种方法。

第二种方法是使用 boost 库，不表。

之后遇到了有多个使用或和等于的逻辑判断符，换个思路，其实用 set 可能也是一个不错的思路。有一点很有意思，如果找不到，一般会返回该容器的 `.end()` 的值。

SECTION 12

cf's 85A problem

2020-01-13 01:32:53.936011

和上次那道关于字符串的题很像。都涉及到了把字符串转换为相应的小写形式。
关于转换的函数，应该是下列的样子：

```
std::string lower(std::string str){
    std::transform(str.begin(), str.end(), str.begin(),
        [](unsigned char c){return std::tolower(c)});
    return str;
}
```

我自己在使用的時候，lambda 函数没有加上 `return` 语句，下次一定。(lambda 函数是一个黑盒，必须要有输入有输出)。

¹在 <https://en.cppreference.com/w/cpp/algorithm/transform> 中

SECTION 13

关于 C++ 中的 using 和 typedef

2020-01-13 17:25:19.221487

总是因为泛型的原因要声明很长的变量类型，但是其实有为变量类型设置别名的方法。第一点是使用typedef来重命名变量。格式如下：

```
typedef org new;
```

另外的方法是使用using来进行命名变量的工作。这是 C++11 起才开始支持的，语法会更好一些，更统一化一些。另外它还支持模板操作。普通的用法是这样：

```
using new = org;
```

或者说这样：

```
template <typename T> using my_type = whatever<T>;  
  
my_type<int> my_var;
```

这样相较而言，typedef 就好像是宏定义的一样（当然并不是）。

SECTION 14

Python 中的 __new__ 方法

2020-01-14 09:28:06.576783

在 Python 中的__init__方法一般只是用来设置属性用的。换言之，__init__只是在使用__new__后获得对象后给对象加属性而使用的特殊方法。

所以说，真正可以获得对象的方法，还是要用到__new__的特殊方法。

而一般的类设计是不需要定义__new__特殊方法的，原因是对于它们来说继承属于object类的__new__方法就 OK 了（事实上是没有 object 这个类型的声明的，但是关于这个概念，每个设计类的人都要理解，因为它用 Python 的解释器实现的，是一切类的基石，就像内建类型一样，不过话说，object 也的确是内建类型）。

所以可以认为__new__是特殊方法中的特殊方法。是调用类之后的之后第一个被调用的类方法。而它生成的对象更是其他方法的基础。

因为这个，__new__不同与其他方法一样，反之，它被传入的第一个参数是cls，是类对象，而不像其他方法一样传入的是实例对象，也就是self。在最后的最后，__new__会返回一个类对象所对应的实例对象。

从类到对象，一般而言只需要调用`object.__new__(cls)`方法就可以了。如果想对自己的对象加入更多的细节，都可以在自己的类下的`myclass.__new__`定义余下的，甚至还可以实现元类。

当然如果有选择的话，在自己的对象下实现`__init__`来定义，这永远是最优选择，就如 Python 之禅所说的一样。当然从另一个方面来看的话，我们会发现`__init__`并没有我们想象的那么必不可少。很多时候甚至可以找到其他的办法来实现`__init__`方法所能办到的。但是其他的方法一般来说完全没必要，简单的`__init__`已经简单得够招人喜欢了。

在`__new__`中，为了使用父类所已经完成了的工作，也可以使用强有力的`super()`，它的参数还有几个可以传值呢，从而为继承提供了更好的基础。

SECTION 15

Python 中的 hash

2020-01-15 02:24:11.383948

想要完成的功能：为不同的类型的对象进行一个独一无二的 hash 标记。

为了完成这个功能，我看了看 Python 的标准库，`hashlib`库。`hashlib`库为 hash 算法提供了良好的支持，比如说要计算 my-object 的 sha-1 的哈希值，可以这样：

```
m = hashlib.sha1()          # 新建 sha1 对象
m.update(str(my_object))
m.digest()                  # sha-1 值
m.hexdigest()               # 表示为 16 进制字符串的 sha-1 值
```

(?):update 可以直接接受字符串吗？它不用被编码吗？

SECTION 16

cf's 1288A problem

2020-01-15 07:11:57.073754

看了看其他人的回答，发现了有人求最小值的时候没有用循环，还用到了莫名奇妙的变量，我想，莫非这是用导数算出来的吗？

于是我找到了导数等于 0 的点，然后带入原函数，发现了眼熟的变量..... 我的天呐，我竟然看到别人用数学解题，这算不算优化过度？我蛮喜欢数学的，但是我也喜欢简单直接的可读性，我就佩服佩服一下了。我还是用循环来做题吧。

还有人写了一行 Perl 代码，而相比之下我写了 31 行才搞出来。先生真若神人也。

在我自己的代码里也有不少的问题，希望在 C++ 里变量重用，导致一些判断的时候出了不想出现的结果，毕竟原值已经被更改了。

这是一个问题，我下次一定要仔细看看，多做题，争取养成 0bug 的习惯。

(又看了若干道，用数学方法来做的数不胜数，但是也看到了一些有趣的工具)

在 `cmath` 中提供的 `std::ceil` 和 `std::floor`，它们可以把 `float` 类型的值转换为相近的类整型浮点数。其中 `std::ceil` 负责把数进行“上升”，而 `std::floor` 负责把数进行“下降”。

除此之外还有 `std::round` 和 `std::trunc`。都是对浮点数进行离散的函数。

我自己是自己写了一个函数来实现 `std::ceil` 的功能，性能不知道会差多少，但应该没差多少。

SECTION 17

cf's 1288B problem

2020-01-15 09:01:57.170873

数学真香，我爱数学，数学爱我。

这道题的题目定义了一个函数，叫做 `conc`，实例如下：

```
>>> conc(314, 15926535)
31415926535
>>> conc(5741, 59635)
574159635
```

原文是，“`conc(a, b)` is the concatenation of `a` and `b`.” 也就是说，`conc` 把 `a` 和 `b` 拼接起来得到一个新的数，有关系式

$$\text{conc}(a, b) = a \times 10^{\text{len of } b \text{ in base } 10} + b$$

目前看来为了得到 `b` 在 base 10 下的长度只能用算法试出来。

如果有关系式满足

$$\text{conc}(a, b) = a \cdot b + a + b$$

化简后则恒有：

$$10^{\text{len of } b \text{ in base } 10} = b + 1$$

也就是说如果要在 $(1 \dots A)$ 和 $(1 \dots B)$ 中找到所有符合条件的 `a, b`，只需要求出 $10^{\text{len of } b \text{ in base } 10} = b + 1$ 的 `b`，然后乘以 `A`（因为所有的 `a` 都是满足条件的），答案就出来了。

当然除了数学问题，还有其他的很多问题，比如说：

```
while(temp--){
    //change the value of value;
```

上面这个代码可能不会运行 `temp` 遍的，因为 `temp` 在头部和循环题中用的都是同一个内存空间。

17.1. 后记

(看来别人的回答后.....)

我对不起我的数学老师！我单知道在 Python 里面是简单直接用 `len(str(num))` 来获取长度，获取在十进制下的长度，我不知道其他的函数还有更数学化的 `log10`！我真傻，真的！

(手动狗头)

SECTION 18

cf's 1288C problem

2020-01-16 06:55:23.613204

这道题的题意经分析之后可以知道，主要是为了求得单调递增数列的可能排列数。

它给出了一个数列的长度 len 和每个元素的可能取值 $1 \leq a_i \leq n$ 。我设可能排列数为 $f(len, range)$ ，其中 $range$ 是 $n - 1 + 1$ 的值， a_i 只能取整数。比如说， $f(2, 2)$ 是 1, 1, 1, 2, 2, 2 三的排列的排列数，故值为 3。

注意到， $f(len, range)$ 在一般情况下，可以分为两种方面考虑，第一种是数组的第一个元素取到了 $range$ 对应的第一个元素，所以剩下数组剩下的元素长度减一，元素取值范围不变（因为不是严格单调），总的排列数为 $f(len - 1, range)$ 。

另一种是数组的第一个元素没有取到 $range$ 对应的第一个长度，这个时候总的排列数为 $f(len, range - 1)$ 。

故 $f(len, range)$ 满足下式：

$$f(l, r) = \begin{cases} 1, & \text{if } r = 1, \\ r, & \text{if } l = 1, \\ f(l - 1, r) + f(l, r - 1), & \text{if } l \neq 1 \text{ and } r \neq 1. \end{cases}$$

所以建立一个 $len \times range$ 的二维数组，自底向上把数组填满到 $f(len, range)$ ，就可以求出答案了。

这个之外我还犯了一个小问题， $10e9+7$ 不是 $10^9 + 7$ ，因为在代码中那个 `e` 就已经有 10 的意思了，所以正确写法是 $1e9+7$ ，昨天找这个找了好久，真是太笨了。

18.1. 关于排列组合

看了其他人的做法，发现有不少人用的方法是通过排列组合，也就是通过 $f(len, range) = C_{len+range-1}^{range}$ 来解决问题的。我能理解，但是如果我一开始没有想到，那这样才可以把 $f(len, range) = f(len - 1, range) + f(len, range - 1)$ 解成我想要的 $C_{len+range-1}^{range}$ 呢？

SECTION 19

XeTeX 的命令行乱码

2020-01-17 06:30:12.541703

发现在命令行里 XeTeX 中的信息输出老是乱码，乱码总是会怀疑到编码的问题上来。

仔细在网上找了找，用 `cmd` 的命令 `chcp 65001` 改变了它的输出编码，顿时输出的东西就正常了（`chcp 65001` 是把编码页改到 UTF-8 上面去）。

现在有一个问题，是 XeLaTeX 编码格式本身就是输出的 UTF-8 格式，还是调用它的 `subprocess` 的输出格式是 UTF-8？

而且，我认为程序和 shell 之间用的是编码后的二进制数据流进行交流的，是这么一回事吗？

SECTION 20

cf's 1285B problem

2020-01-17 17:06:11.000355

有点伤心，这道题我没有做出来。

这道题是为了求取最大的连续子串之和，令其为 $f(list, start, end)$ ，有三种可能情况。第一种，数组串本身就是最优子串，其值为 $sum(list.begin \rightarrow list.end)$ 。第二种，最优子串不含有数组串最后一个元素，所以其值为 $f(list, start, end - 1)$ 。第三种和第二种类似，为 $f(list, start + 1, end)$ 。

可以注意到其中第二种和第三种有重复的地方，比如 $f(list, start + 1, end - 1)$ 就同时是第二和第三种情况的子串。

为了降低时间复杂度，可以使用动态规划。我采用了自底向上的设计模式，所以每个子问题的解都依赖与之前的子问题之解。每个子问题都用了一块内存来存储其的最大的连续子串之和。（我甚至还记录了它们对应的 $sum(list.begin \rightarrow list.end)$ ！）这样，计算每个子问题的时间复杂度为 $O(1)$ ，一共的复杂度为 $O(n^2)$ 。

结果过不了？我晕了。

接下来说一些这个题揭露出来的问题。

20.1. segmentation fault

在编译后运行时直接跳出了这个错误，并提示发生了段错误。

它的意思是访问了不应该访问的地址，操作系统及时阻止了（干得好），并抛出了错误原因。

20.2. max

关于求取最值，在之前有认识过求取容器中最值的标准库函数 `std::max_element`，今天要用到其他的函数方法。

在非容器中，使用可以使用 `std::max`，它的一个原型如下：

```
template< class T >
const T& max( const T& a, const T& b );
```

而另一个原型则如下：

```
template< class T >
T max( std::initializer_list<T> ilist );
```

这个函数可以表示两个数的最大值，也可以表示一个 `initializer_list` 的最大值。

比如说为了取得三者之间的最大值，可以使用：

```
std::max(std::max(x, y), z);
```

或者使用更简单的语法结构：

```
std::max({x, y, z});
```

20.3. accumulate

求和。

从第一个迭代器开始，一直求到最后一个迭代器。包含第一个迭代器的值，但不包含最后一个。

所以说要表达类似 Python 中的 `[:]`，可以使用 `v.begin()` 和 `v.end()`。如果要表达 `[a:]`，则可以使用 `v.begin()+a`，而 `end` 不变。但是如果表达 `[a:b]` 的话，则应该表达为：`v.begin()+a`，而结尾则变为 `v.begin()+b`，效果是一样的，都取不到最后一个值。

20.4. 最后，这道题的答案

我的想法是把 Yasser 的和算出来，然后把 Adel 的最优子串算出来。两者相互比较，最后判断输出答案。

但是他们的答案却不是这样。

他们的想法是：

对于给定数组 $\{a_1, a_2, \dots, a_i, \dots, a_n\}$ ，全部的和明显是 $S_A = \text{sum}(\{a_1, a_2, \dots, a_n\})$ 。

如果开始的部分和 $\text{sum}(\{a_1, a_2, \dots, a_i\}) \leq 0$ ，则明显有 $\text{sum}(\{a_{i+1}, \dots, a_n\}) \geq S_A$ 。

同理，结尾部分存在该子列，或者开头和结尾都存在这种子列，都很明显满足这个条件。

逆否命题也成立。故原条件可以推广到更容易求解的状态。

所以说只需要在前后处寻找是否有符合条件的子列。时间复杂度从 $O(n^2)$ 变成了 $O(n)$ ，只是求不出来最优子列的值了，只能判断有无最优子列。

SECTION 21

cf's 1285C problem

2020-01-23 09:41:49.392911

现在还在running on test 36, 都不知道发生了什么。

21.1. 关于题面

首先题面说了有 $LCM(a, b) = X$, 其中 “ $LCM(a, b)$ is the smallest positive integer that is divisible by both a and b”。很明显, 如果存在数 x , 有 $x|a$, $x|b$ 的时候, 也就有 $LCM(a, b) = LCM(\frac{a}{x}, b)$, 因为 $LCM(a, b) = \frac{ab}{gcd(a, b)}$, 而又有 $LCM(\frac{a}{x}, b) = \frac{ab}{gcd(a, xb)}$, 两式等价。

所以对于 a, b , 很容易可以找到互质的两个数 c, d , 使得 $LCM(a, b) = LCM(c, d) = cd$, 这时 c, d 最小, 所以 $max(c, d)$ 相对最小。

现在给定 $N = cd$, 不妨假定 $c \leq d$, 则有 $c \in [1, \lfloor \sqrt{N} \rfloor]$, 此时不难看出 $N \pmod c = 0$, 也就是 $c|N$, 且 $gcd(c, d) = 1$, 不然得到的数为 $\frac{cd}{x}$, 而不为 cd 。

迭代一次, 寻找满足条件的 c 的最大值, 答案也就出来了。

21.2. 其他答案

有一点点改进的空间, 那就是逆向寻找, 找到一个值就可以输出了。

SECTION 22

关于 undefined reference to 'WinMain'

2020-01-24 05:16:31.300438

因为如题目这样的 **Compilation Error**, 导致了我重新安装了一边 g++, 因为一直在纳闷明明没有使用 WinAppiliction 的我, 是怎样搞出的 “WinMain”, 结果看了一下 (在半个小时之后了), 是我的 main 函数写错了, 写成了 maim。

SECTION 23

cf's 1294B problem

2020-01-25 06:07:54.481951

这是一个简化的机器人走路的题，只能向上或者向右走。并询问你是否能够完成任务和完成任务的方法。

分析后可知，当机器人处于一个点时，它只能到达以它为原点的第一象限区域。也就是说，每一个点都必须要在前一个点的右上方。易知，对于一个点来说，除了其右上方的点外，它相对与其他点来说都是在右上方向的。

所以对其他的点来说，令它们以 (x_i, y_i) 排好序后，就分为两种情况。第一种，在同一个 x 上，排在后面的都是在它的上方，第二种，在不同的 x 上，其点的 y 坐标很明显会大于这个 x 坐标上最上面的一个点的 y 坐标。（如果不是，就无法完成任务）

23.1. little bug

其一。

对于 `*a.b`，我不是很清楚它们之间的优先级.....它会先调用点运算符，之后在调用`*`号运算符，所以要使用 `(*a).b` 的形式。

其二。

`int`和`char*`是不能相乘的(我知道这是屁话,但还是希望它能搞个`string`出来。然后`int`和`string`也是不能相乘的，这个倒是超出了我的预算。要使用重复的字符串，应该使用：`string(int n, char c)`来生成。

其三。

通过上面的那一回事，我发现了盲点！`type t(a, b)`和`type t = type(a, b)`应该是等价的才对。

其四。

关于`for auto`，它有这种形式：`for(auto& a: a_s)`，其中的`a`，其实是`(*iterator)`，而不是迭代器。

23.2. 关于其他人的答案

其他的都差不多，但是好像`pair`是可以比较大小的，所以标准排序算法`sort(type_t a.begin(), type_t a.end())`是可以支持的。这就让给`pair`排序成为了可能。

SECTION 24

cf's 1294C problem

2020-01-25 13:49:00.375339

这个题的题面是为了让一个数 N 分解成三个不相等的数 a, b, c 。即有 $N = a \times b \times c$ ，而且 a, b, c 三个数互不相等，且所有数大于等于 2。

不妨设 $a < b < c$ ，所以 $N = a \times b \times c > a^3$ ，即有 $a \in [2, \lfloor \sqrt[3]{N} \rfloor]$ 。

```
int result[2], index = 0;
//(index = 0) -> (i^3 < N); (index = 1) -> (i^2 < N);
for(int i = 2; pow(i, 3 - index) < N && index < 2; ++i)
    if(N % i == 0)
        result[index++] = i,
        N /= i;
if(index != 2)
    cout << "NO" << endl;
else
    cout << "YES" << endl << result[0] << " "
        << result[1] << " " << N << endl;
```

当在for循环中,可以保证如果result[0]和result[1]能正确产出,num也能成为所谓的result[3]。

所以说判别标准就是所谓的index是否为 2，否则就说明了至少有一个result 是没有被正确产出的。

SECTION 25

cf's 1293A problem

2020-01-28 09:05:51.832202

本题给出了连续离散集合 $A = \{1, \dots, n\}$ ，同时也给出了非连续的离散集合 $B = \{b_1, \dots, b_k\}$ ，且有 $b_i \in A, i \in [1, k]$ 。

开始给定了初始数字 $s \in A$ ，然后求出 i ，对于 $I = s + i$ or $s - i$ ，如果对于 i 有 $\exists I, I \in A$ and $I \notin B$ ，则求出对应的最小的 i 。

很明显，当 $s \notin B$ 时， $i = 0$ 。

我是通过set来作为 B 的容器（以减少查询的时间），然后从 $i = 0$ 开始查询，看看对于 $I \in A$ 是否满足 $I \notin B$ ，当有满足的则返回 i ，此时的 i 即为最小的 i 。

另外我发现了一点，在 C++ 中，只有 0 才代表了false，其他的数都代表了true，即使是负数也不例外。

25.1. 其他代码

对于`set`，我猜测`find`和`count`的复杂度应该差不多。所以在不支持 C++20 的编译器来说，`count`和`contains`在某些方面表现了惊人的可代替性。

SECTION 26

cf's 1292 problem

2020-01-29 09:02:59.956693

这道题构建了一个 $2 \times n$ 的地图。在 q 次输入中， i -th 的输入为 (r_i, c_i) ，则在地图上添加或移除相应位置的障碍，且求出是否有路径可以从 $(1, 1)$ 格一直走到 $(2, n)$ 格的位置。

我马上就想到了图的广度搜索，但很明显不对。

然后发现模拟行动路线，走到 (r, c) 位置上时，如果有障碍物在 $(r + 1, c)$ 或者 $(r + 1, c + 1)$ 的时候就明显无法走通。对于每一次输入，都要完整的走一遍，最坏的时间复杂度是 $O(n)$ 。果然`time limit`了。

之后对每个障碍物都看看是否周围有不该存在的障碍物。时间复杂度就变成了 $O(i)$ 了，随着输入次数的增加而增加。还是没通过。

26.1. 他人答案

在看到了其他答案之后，我发现可以使用一个值`barr`来表示通畅度。

在我做够题之后再想想这道题吧。

SECTION 27

cf's 1294D problem

2020-01-30 04:29:07.199743

这道题有着严格的时间限制。我小心翼翼的把它优化成了 $O(s)$ 的时间复杂度，但是依然卡在时间上了。而且最优情况下的时间复杂度是 $O(1)$ 的说.....

27.1. 他人答案

我看到有人用 Python 做出来了，我才发现我就是个笨比。

我分析一下我的想法，就是每一次循环的时候，都从头开始算，虽然有辅助记忆的变量，但还是解出不能。

而看看其他同学的答案，发现他们是在每一次之后接着上一次循环继续做。这样所用的时间就会少一些。

另外我发现使用printf和scanf的代码会比不使用的代码，运行时间会少十倍。（我真的震惊到爆^[1]）

这么说的话，虽然不喜欢混用 C 和 C++ 的风格，但是为了能通过，还是使用 C 风格的会运行的快一点。（希望之后不会逼我使用大整数而不是模板）

SECTION 28

puts, printf and fwrite

2020-01-31 02:28:52.041320

（感觉我已经走火入魔了）

首先输出一个字符的时候，其实最快的是putchar，但是要输出字符串的时候，不输出格式化字符串的时候，函数的速度有：fwrite>puts/fputs>printf。

其中fwrite的函数声明是：

```
size_t fwrite(const void *ptr, size_t size, size_t count, FILE *stream);
```

而根据 stack overflow 的一篇回答²，可以看出它省略了调用strlen的函数开销。而相较于printf，它更省略了其他的开销。

SECTION 29

cf's 1295B problem

2020-02-01 02:14:27.397813

又是一道没有做出来的题目.....

²<https://stackoverflow.com/questions/2454474/what-is-the-difference-between-printf-and-puts-in-c/2457714#2457714>

29.1. 最小同余正整数

为了得到 n 关于 m 的余数，自然是下意识使用 $n(\bmod m)$ ，但是在 C++ 中使用 `n % m`，却可能不是想要的正整数结果，比如：

```
cout << -1 % 100 << endl;
// -1
```

原因是为了将求余推广到整数上，有以下的等价关系：

```
//n > 0, m > 0;
n % m;
n % (-m) == n % m;
(-n) % m == -(n % m); //abs(n % m) < abs(m);
```

很明显当 $n < 0$ 时，求出来的就不是正整数了，而为了求出最小同余正整数，根据求余的性质，只需要在求余之后加上 $|m|$ 就可以了。唉，如果它和 Python 学学就好了，在 Python 中 `-1 % 23` 就是 22。而在 C++ 中不得不用：

```
int atom(const int& n, const int& m){
    int a = n % m;
    if(n >= 0) return a;
    else return a + abs(m);
}
```

来作为结尾。

29.2. 寻找元素

在 `std::set` 中寻找元素，可以简单的使用如 `count`，或者 `find(set.begin(), set.end(), item) != set.end()`，如果在 C++20 中，还有 `method contains` 来寻找。

而在 `std::vector`，也只能使用 `find(vec.begin(), vec.end(), item) != vec.end()` 来寻找元素。

当然也可以定义一个模板函数来简化它。

29.3. 整数同号

为了得到两个数是否同号的布尔值，可以使用 `std::signbit`，对于整数有：

```
signbit(0); // false;
signbit(1); // false;
signbit(-1); // true;
```

SECTION 30

cf's 1296A problem

2020-02-08 11:07:20.858709

为了给数组赋值，可以使用：

```
int len_arr;
cin >> len_arr;
vector<int> arr(len_arr);
while(len_arr--){
    cin >> arr[len_arr];
}
```

或者使用for auto结构进行遍历：

```
for(auto &i: arr)
    cin >> i;
```

我在做这个题的时候，一开始使用了`arr[len_arr-1]`代替了`arr[len_arr]`，之后就理所当然崩溃了，原因是减一的操作是在判断之后，所以说，在循环体中`len_arr`的值最小是 0。

另外为了求取sum值，应该要使用for auto语法或者说是使用`accumulate`，不过for auto使用的时候会需要一个值，而另一个则可以直接返回值。

SECTION 31

cf's 1303A problem

2020-02-14 16:09:05.315418

给定一个由 0 和 1 构成的数值，为了让数的 1 聚集在一起，求最少要删除的 0 的个数。

我的思路是，把首末两端的 0 给去掉，然后count1 的个数，即得到答案。

31.1. 我的答案

我设置了一个标志，用来判断是否在首末为 1 的子串中，如果标志为真，如果遇到 0 则把temp_cnt自增，不然就令`cnt += temp_cnt`，这样也能避免末尾的 0 被归入计算中。

31.2. 其他人的答案 - 1

string有一个方法是find，它会接受一个字符或者字符串，和初始位置（默认为 0），返回被找到字符的位置。

利用它，可以在寻找两个 1 之间的 0 的个数。比如给出初始位置为 n ，而返回值为 m ，则如果 $n = m$ ，则说明两个 1 之间没有空隙，有 $m - n$ 个零，并让 $n + 1$ 。

而如果 $n < m$ 也同理。

31.3. 其他人的答案 - 2

`string`除了`find`，也还有`rfind`，所以把首末的 1 的位置找到并递归就明显好做得多。

SECTION 32

cf's 1303B problem

2020-02-14 16:35:21.606044

首先因为`scanf`和`long long`之间，不太熟悉而所以处处碰壁。

32.1. 关于`long long`和`scanf`

测试代码如下：

```
#include<bits/stdc++.h>
using namespace std;
using ll = long long;

int main(){
    ll a, c;
    int b;
    scanf("%d%d%lld", &a, &b, &c);
    cout << a << ' ' << b << ' ' << c << endl;
    printf("%lld %d %d\n", a, b, c);
    return 0;
}
```

测试输入输出如下：

```
$ ./a.out
4294967296 4294967296 4294967296
0 0 4294967296
0 0 0
```

可以看到哇，对于`a`来说，只有使用`%lld`，才会使得它把正确的数据储存到相关的内存位置上，而不然的话，它则会使用错误的数据，可以看到高位的 1 被舍弃掉了。

同理，`printf`则根本没有读高位的 1。

32.2. 其他人的答案

在这道题中我要求到 $\lceil \frac{n}{m} \rceil$ ，而众所周知 C++ 中整数相除是得不到小数的，所以我用了：

```
(n % m) ? n / m + 1 : n / m
```

但换一种思考方式，只有 $m|n$ 的时候， $n \bmod m$ 才会等于 0， $\lfloor \frac{n}{m} \rfloor$ ，在这种时候可以直接等于 $\lceil \frac{n}{m} \rceil$ ，但是在其他时候都有：

$$\lfloor \frac{n}{m} \rfloor + 1 = \lceil \frac{n}{m} \rceil$$

所以说只要令 n 加上 $m - 1$ 成为 n' 时，当原 n 存在有： $n \bmod m = 0$ 时，变化后的 n' 有 $\lfloor \frac{n'}{m} \rfloor$ 的值不变。但是对于原 n 有 $n \bmod m \in [1, m - 1]$ 时，都有变化后恒成立： $\lfloor \frac{n'}{m} \rfloor$ 有原 n 的 $\lfloor \frac{n}{m} \rfloor + 1$ 的值。

故上代码的等价代码为：

```
(n + m - 1) / m
```

SECTION 33

cf's 1303C problem

2020-02-14 19:14:25.856520

这道题，给定了一个密码，密码的每两个相邻的字母在键盘上也是相邻的，求取符合条件的一维键盘布局。

很明显是要求输入一个密码，输出一个布局格式。而对于格式而言，因为是一维的，所以我是想着可以根据这个建一个图，其中一般情况下，每个点要不然有 2 个边，要不然没有边（就是没有构成密码的字母），而有一个边的点就是密码区块的开始和结束，所有的密码都是在密码区块中的。

以上是我的思路。当然具体要复杂一些，包括判断有无环，还有密码长度为一的特殊情况。

33.1. 其他人的答案

有人是这么做的：使用 `bitset` 记录所有密码的字符集，如果密码的下一位没有在字符集中，则把字符链接到密码区块的前面中。如果一直是这种线性的结构的话，维持 $p = 0$ 的位置， p 一直指着最新的字符。

这样的话，到时候就能直接输出密码区块，之后再把不在字符集中的所有字符输出。

但是密码的下一位出现在了字符集中，而且 $p = 0$ （这代表了它之前是线性的），而且最重要的是，这个字符不在正在增长的密码区块 `result` 的第 $p+1$ 位中，也就是和本位 `result[p]` 的上一位不一样，说明了下一位在密码区块中已经有左右两个字母，还要和本位在一起，这在一维上是矛盾的。

但是`p`不可能一直指向最新的元素，如果它指向最老的元素，即`p = a.size() - 1`，此时有一个不在字符集中的元素，这可以把该元素链接在后面以成为最老的元素（因为这个元素和之前那个元素很近，又不在密码区块中，那只能在密码区块的另一边中了啦）。但是的话，而且`result[p-1]=ch`，就令`p--`，让它指向它应该在的位置。

以下是最主要的循环：

```
for(auto& ch: input){
    // if ch is in alphabet:
    if(alphabet[ch-'a'])
        // if it is not at the tail, so we can move p
        if(p < result.size() - 1 && result[p+1] == ch)
            p++;
        // if it is not at the head, so we can move p
        else if(p > 0 && result[p-1] == ch)
            p--;
        // oh we can't move it!
        else{
            flag = true;
            break;
        }
    // if ch is not in alphabet,
    // then should link it to result,
    // its head or tail, whatever.
    else
        // if it is on the head - it is easy
        if(p == 0)
            result = ch + result;
        // or it is on the tail
        else if(p == result.size() - 1){
            result += ch;
            p++;
        // it is on the inside of result
        } else {
            flag = true;
            break;
        }
    alphabet.set(ch - 'a');
}
```

SECTION 34

cf's 1304C problem

2020-02-16 10:11:01.261449

这道题需要判断有无相对应的回文串，所以我必须要得到一个字符串的`reversed`版本。

在我的程序中使用的是自己实现的`reversed`函数，但是在标准库中已经有了`std::reverse`函数，传入两个迭代器指向首和末，实现原址颠倒。里面用到的函数有`std::iter_swap`

此外我之前企图在迭代的过程中更改原容器的值。但是更改容器的操作好像（应该）会引发一些错误。

此外，我试图把输入和运行相隔开，但是这样的话，可能甚至会让情况复杂化!!! 所以如何取舍呢，有点难☹。

SECTION 35

cf's 1304 problem

2020-02-16 10:58:40.308151

是时候学点新知识了，铛铛！**struct**！之前忙着做题，不敢使用这个怕做错了就麻烦了，结果我还是完全记得这个语法结构的说。

使用结构肯定比使用数组要好，毕竟数组是使用来储存意义相同的数据，而结构是用来储存意义不一样的结构。用来比较的话，一个相当于 Python 中的列表，另一个就像元组和类。

此外，鉴于我经常在里面犯一些低级错误，重载输出操作符也是很有必要的，比如说：

```
ostream& operator<<(ostream& os, const vector<p>& it){
    os << '[';
    for(auto &i: it){
        os << "{" << i.t << ", " << i.h << ", " << i.l << '}', ' ';
    }
    os << ']' << endl;
}
```

其他的和紫名大佬差不多，除了我把循环和处理和输入输出都分开了。

我在想是不是没有必要再竞赛中把输入输出和处理隔开。

SECTION 36

cf's 1296C problem

2020-02-16 22:15:33.313424

我又被卡住啦！鼓掌！（啦啦啦啦）

这道题看起来很像是 dp 的题嘛，但是没有想到不行☹，也是哈，毕竟是 $O(n^2)$ 嘛。

这道题意思如下：有一个字符串，要寻找它的最短的符合规则的连续子字符串，而所谓规则，就是该字符串中的 'U' 和 'D'，'R' 和 'L' 的数量两两相等。

我是这么想的：定义一个结构体 $y = f(str)$ 来存储 $str[i..j]$ 的四个值，于是结构体满足： $y_{i..j} = f(str[i..j]) = f(str[i..j-1] + str[j]) = y_{i..j-1} + f(str[j])$

虽然我是菜鸡，但是动态规划什么的，自底向上简简单单的啦。说着我啪嗒啪嗒敲好了代码，还为结构体重载了运算符。

结果time limit，也是毕竟是 $O(n^2)$ 的算法嘛，暴力也是 $O(n^2)$ 的复杂度对吧。

但是真的有比这个好的算法吗.....

36.1. 其他人的答案

还是有更好的方法，这道题的目的就是为了找到 i 和 j ，但是并没有必要找到 $1 \leq i \leq j \leq len$ 中所有满足的 $a[i..j]$ 的数据结构，因为这样子，时间复杂度为 $O(len^2)$ 。

但是从另外一种方向考虑的话，这种题目的数据结构是有它的特点的：对于 $a[1..i]$ 和 $a[1..j]$ 来说的话，如果增加的 'U' 和 'D'，增加的 'R' 和 'L' 两两相等的话，那么一定 $a[i..j]$ 满足条件。

如果把所有的 $a[1..i]$ 都储存下来，当一个数据结构相等的时候，就可以肯定，我们找到了一个符合条件的值。

综上，时间复杂度可以降到 $O(n \lg n)$ ，甚至是 $O(n)$ （如果用哈希表的话）。

SECTION 37

cf's 1307B problem

2020-02-18 11:42:37.636214

危 我 危！我差点就要掉到绿色等级去了啦!!! 只做了一道题，第二题，第三题完全不会，说实话还是有点点灰心的吧。

回到这道题来，我认为这道题完全就有误导我的嫌疑！

给定原点 $O(0,0)$ 和目标点 $A(x_A, 0)$ ，又给定了可选的距离 $B = \{b_i\}$ （可以在二维平面上移动）。然后要使用这些可选的距离从原点移动到目标点。

要求取最少次数。

开始我的想法是：如果用 m 表示最大距离，那么如果能用 m 就最好用。要是 $m|x_A$ 成立，那么很明显 $\frac{x_A}{m}$ 是所求值。但是不成立呢？我认为可以先以 $2m$ 为单位地开始移动，到最后很明显剩下的距离有 $e \in [1, 2m - 1]$ ，如果 $e \in B$ ，那么很明显最后一节一步搞完，不然两部搞完（用三角形两边之和大于第三边，那么两步可以移动的距离就是一个范围： $(0, 2m]$ ）。

但是一开始的实例就过不去，因为当 $x_A = 12$ 时， $B = 3, 4, 5$ 时，我认为的算法给出的答案是： $5 \rightarrow 5 \rightarrow 2$ (two step)，一共是 4 步。而底下的注释告诉我应该是 3 步： $4 \rightarrow 4 \rightarrow 4$ 。就是这个迷惑了我，让我以为其他的数字，也会参与到构建路径中，让我以为有一个最优解，以至于不得使用动态规划。

动态规划可是有 $O(n^2)$ 的复杂度，这道题就很明显的堵住了我。

之后看了其他人的答案后：我开始为他们的代码的短小精悍感到吃惊，结果看来反而和我一开始的想法有同工之妙。

他们还是用最大距离 m 来解题，而最后剩下的距离和之前的 m 步长的一步加起来，肯定在 $[m+1, 2m]$ ，故最后最少必定可以用到两步。易证明到，在这个分割点之前是最优解，分割点之后是最优解，而这个分割点本身也是最优分割点（虽然不止一个最优分割点，但其他分割点的答案不比它的答案还有优，也就是说，这么分割必得最优解）。

SECTION 38

cf's 1307C problem

2020-02-18 07:16:08.124380

老规矩先说题干。给定一个字符串，找到符合规矩后寻找到的出现次数最多的子字符串。规矩是原字符串中寻找成等差数列的有限长度下标集合（其中最短长度是 1），然后将其按照下标提取出来。

这道题的陷阱就是这个等差数列了，这个也正是最迷惑人的东西了，因为题解中最主要的部分和等差一点关系也没有。

以下是题解原文：

We observe that if the hidden string that occurs the most times has length longer than 2, then there must exist one that occurs just as many times of length exactly 2. This is true because we can always just take the first 2 letters; there can't be any collisions. Therefore, we only need to check strings of lengths 1 and 2. Checking strings of length 1 is easy. To check strings of length 2, we can iterate across S from left to right and update the number of times we have seen each string of length 1 and 2 using DP.

如上，证明分两步：一，长度 $len_b > 2$ 的字符串 b 出现次数小于等于长度 $len_s = 2$ 的字符串 s ：如果有字符串 b 存在的话，其对应的前缀 s 必定也满足下标成等差数列。故 b 出现次数必定小于等于 s 出现的次数。二，长度为一的字符串可能出现次数小于长度为二的字符串的出现次数：不同下标集合构成的长度为二的字符串，可能前缀都是同一个下标对应的字符，所以长度为 1 的字符串可能对应多个长度为 2 的字符串，但是如果存在的话，长度为 2 的字符串却有且只有一个对应的 $len_b > 2$ 长度字符串。

如上证明了我们最主要的地方就是分析长度为 1 和 2 的字符串。

SECTION 39

MIT 6.006 part1(unit1)

2020-02-19 14:58:32.463955

In this course, I am interesting about the finding peak in 1D-graph or 2D.

So what is the peak in 1D-graph? for a number in a 1D-array, if its left L (if L exists) is less than it and its right R also do it (if R exists), then the number is a peak.

there is a $O(\lg n)$ function that can work it out: firstly, look at $n/2$ position, if the element is a peak then break the loop; if it is not, and the left is bigger than itself, we can know that in the sub-array $A[0..n/2 - 1]$ must have a peak element inside. (why?) else it can look for the sub-array $A[n/2 + 1..n - 1]$.

It is easy to find out the complexity is $O(\lg n)$.

For the question why I assert the sub-array must have a peak at least inside, the answer is that the sub-array must have a max-element, and if the max-element is at the edge of the array, then it's easy to know that it is a peak.

In a 2D-graph, the peak is bigger than or equal with its round cell numbers. the best function to find one of those peaks swiftly is: firstly, look at the middle column and find out the max-element of the column, and if the element is a peak then return the answer, but if it does not and the round cell number C at column $n/2 - 1$ is bigger than it, we can say that on the area from column 0 to column $n/2 - 1$ there must be a peak at least.

Why? - if the peak P is on the column $n/2 - 1$, the peak is must bigger than C and then is bigger than all numbers on column $n/2$, because if it is small than C , the max-element is not it, and we can now look at the max-element of the area.

SECTION 40

cf's 1296D problem

2020-02-20 15:30:16.633458

这是一道简单的 D 题，我不得不说不亏是我（骄傲）。

有几个关于 C++ 的小知识和之前复习到的小技巧：

第一个是 Variable-length array (VLA)，还为维基百科做了一点点贡献，VLA 是在运行时分配内存空间给数组的一个语言性质，一般是在 C99 之后支持。

另外是复习之前的一个小技巧：如果想表达 $\lceil \frac{m}{n} \rceil$ ，则可以使用： $(m + n - 1) / n$ 。

使用标准库函数 `std::sort` 时，作用于数组 `A` 时，作用方法：`sort(A, A+len-1)`。

另外发现宏函数时支持多参数的。

SECTION 41

一道数学题

2020-02-20 19:37:50.946162

题目摘抄如下：

设实数 a, b 满足 $0 < a < b$ ，证明：

$$\frac{\arctan b - \arctan a}{b - a} > \frac{1}{\sqrt{1+a^2} \cdot \sqrt{1+b^2}}$$

一开始的时候很容易看到可以把原式化为：

$$\frac{\arctan b - \arctan a}{b - a} > \sqrt{(\arctan a)' \cdot (\arctan b)'}$$

从而会想到使用中值定理，但是该式根本推导不到有用的中值定理上，而正解是使用反函数的性质。（说明了看见反函数按理应该使用其性质的一般套路）

不妨令 $x = \arctan a$, $y = \arctan b$ ，则根据反函数的性质有：（根据原题可知： $y > x > 0$ ）
 $a = \tan x$, $b = \tan y$ ，所以很明显原始等价于：

$$\frac{x - y}{\tan x - \tan y} > \frac{1}{\left| \frac{1}{\cos x} \cdot \frac{1}{\cos y} \right|}$$

进行又一次简单的变换有：

$$(\cos x \cos y) \cdot \frac{x - y}{\sin(x - y)} > |\cos x \cos y|$$

很明显有：

$$\frac{x - y}{\sin(x - y)} > 1 (y > x > 0)$$

故原式成立。

41.1. 中值不成立的原因

根据中值定理该式 $\frac{\arctan a - \arctan b}{a - b}$ 很明显有： $\exists \xi, (\arctan \xi)' = \frac{\arctan a - \arctan b}{a - b}$ ，所以 $((\arctan \xi)')^2 > (\arctan a)'(\arctan b)'$ 。

为了利用中值定理理解出该题，则只能通过对 $\forall \xi \in (a, b)$ ，都有该式成立来证明。但很可惜对于 $\xi \in (a, b)$ ， $\exists \xi \in (a, b)$ 可以令其不成立。

比如因为 $0 < a < b$ ，有当 $\xi \rightarrow b$ 时，有：（因为 $\arctan x$ 的导函数在正半轴上单调递减）

$$\lim_{\xi \rightarrow b} ((\arctan \xi)')^2 = ((\arctan b)')^2 < (\arctan a)'(\arctan b)'$$

所以使用中值定理是一种错误的选择，该方法绝对搞不出来正确答案出来。

SECTION 42

cf's 1299A problem

2020-02-20 23:25:24.150958

这是一道我不会做的 A 题。

原题定义了一个函数 $f(x, y) = x|y - y$ ，其中的符号 $|$ 代表了进行位运算。又定义了一个数组 $[a, b, c, d, \dots]$ 的值应该为：

$$[a, b, c, d, \dots] = f(\dots f(f(f(a, b), c), d), \dots)$$

现在给定了一个数组，请求得到进行调换顺序之后值最大的数组。

位运算是一个我不太熟悉的东西，但是仔细分析还是很容易发现：对于 x 和 y 而言的第 i -th 位 x_i 和 y_i 而言，进行 f 变换之后，新的值 z 的第 i -th 位 z_i 仅仅与 x_i 和 y_i 相关。因为当 y_i 等于 1 时， $(x|y)_i$ 必定为 1，所以 z_i 必定为 0。（因为当被减数上第 i -th 位为 0 而减数上该位为 1 的话就需要借位，势必和其他位有关）同理可以得出当且仅当 x_i 并且 $\neg y_i$ 时， z_i 为真，故 $z_i = x_i \wedge \neg y_i$ 。

由上因为所有位均满足该运算规律，故有 $z = x \& (\sim y)$ 。

所以对数组而言，自然也有：

$$[a, b, c, d, \dots] = a \& (\sim b) \& (\sim c) \& (\sim d) \& \dots$$

因为 $\&$ 位运算满足交换律，所以原式的值只和第一个数有关。且其所有被运算的数第 i -th 位为真，其才为真。那么什么时候需要调换以让该式的结果更大呢？

从最高位开始比较，如果有且只有一个数在该位为一，那么让它成为第 0 位。因为其他位上都会取反变成 1，而这个数本身的位上的数也保留下来了，从而取到最大。（太棒了）

可是我对位运算不太熟悉，下面是别人的位运算语句：

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    vector<int> a(n);
    for(auto& x: a)
        cin >> x;
    for(int bit=30; bit-->0){
        if(count_if(begin(a), end(a),
            [&](auto x){return x >> bit & 1;})==1){
            swap(a[0], *find_if(begin(a), end(a),
                [&](auto x){return x >> bit & 1;}));
            break;
        }
    }
    for(auto x: a)
        std::cout<<x<<' ';
    std::cout<<'\n';
}
```

首先有巧妙的位运算小窍门： $\text{num} \gg \text{bit} \& 1$ ，它可以确认 num 的第 bit 位是否为一。

其次有有意思的`count_if`和`find_if`。

再然后是 `lameda` 函数。之前我写过的，这里就暂且不表了。

SECTION 43

cf's 1321B problem

2020-03-02 13:43:23.543056

在昨天的 cf 竞赛中，我又挂在了 B 题，原题是这样的：有人想要去旅行，每个城市都有相应的数值，并且第 i 个城市的数值为 d_i ，当且仅当第 j 座城市的数值满足 $b_j - b_i = j - i$ 时，她才可以从城市 i 走到城市 b 中。

我一开始这么想的：点，路，有价值，这不就是有权图的最长路径问题吗？一种有向无环图的最长路程是具有最优子结构的，无论自底向上还是至顶向下，都是可以满足动态规划的。

我于是刷刷的写完了，虽然途中遇上了几个 `core dumped` 说是下标溢出了，但是还是找到错误并且搞好了。

但是罕见的遇上了 `memory limit exceeded`，其实我现在还是感觉很奇怪的，因为怎么想就只是需要用 $O(n^2)$ 的复杂而已，还主要是因为储存图才这样子。

但是昨天于立恒跟我说了他的思路的时候，我感到柳暗花明又一村，发现当且仅当 $b_j - j = b_i - i$ 的时候才会有两个城市可以连接在一起。

所以说最好的方法就是建立一个 `map`，当 $b_n - n = C$ 时，让 `mapC` 增加 b_n 。而 `map` 中最大的值也就是这道题的答案。

SECTION 44

python 的多项式拟合

2020-03-03 20:57:50.277423

之前的话我想要获得一个动态的页面，可以根据不同的窗口大小来动态匹配文本大小，比如说在一些窗口宽度大的地方，一般的像素密度比会小一些，为了显示合适的字，需要把字调小（比如说，16px），有时候在小屏幕上面，则需要把字号调大，比如说调成32px。

我不禁有些疑惑，像素到底是什么呢？比如说虽然移动端字的px值调大了，但是实际上的字的大小却变小了，虽然说是变小了，但是设备像素，本身就高，实际的物理尺寸却没有随着密度变得过于小（实际上移动端的密度对于电脑来说大得吓人）。

实际上，在真实的设备像素和众多的 html 长度之间，隔着一层 CSS 像素层，是一种相对像

素。因为在移动段的 ppi 过于小，而且因为对应的页面指定的px不变。所以说在移动端，我们需要使用更多的像素来表示一个对应的 CSS 像素。

其中设备像素叫做 DP，而 CSS 像素.....还真叫 css 像素。

描述物理像素和 css 像素之间的关系的式子就是 DPR，设备像素比。

而还有 PPI 来描述每单位的像素密度，说实话 PPI 的作用不大，因为不同的设备上并不是说，各各单位一致就可以获得一个完美的关系，一般手机屏幕小，PPI 高，元素的绝对宽高应该是相应会减少。

所以说我们可以根据宽度的 css 像素来确定确定根的字长。

按理来说，在经典视角下我们看到的网页都会差不多一样大，因为 css 像素是在经典使用视角的长度来进行确定的。

但是一般来说我们会适当的想减少信息大小来使得信息更加清楚。所以针对不同的屏幕 css 像素宽度，代表了最少可以显示的信息流，信息流相对较小，那么我们更应该减少信息流来获得很好的观看体验。来达到大小差不多的样子（应该？）

为了只使用一个表达式来表达不同情况下的多项式拟合，使用 python 的 numpy 无疑让人心情愉悦。通过使用numpy.polyfit和numpy.poly1d，很轻松就能得到一个适当的曲线。

SECTION 45

在 wsl 中编写含 CJK 字符的文章

2020-03-04 18:07:35.291983

我用的 wsl 是 Dedian 发行版，尝试使用 pandoc 和 xelatex 来转换 md 文档时发生了不少的事情，比如说不支持中文，这是因为没有设置\CJKmainfont，而且必须要设置正确，不然到时候调用的时候会报错说没有这个字符文件。

字符文件的名字和它的文件名不一致，所以说可以会含有空格，使用fc-list命令，可以获得它的正式名称。在这里可以安装一些字符包（一般来说都是开源的），当然，字符包的名称，字符文件的名称和字符文件的文件名是三个不同的概念。

另外有一件事情有点邪门，明明来说 wsl 是可以让 exe 应用程序访问的，但是 pandoc 好像不行。这就奇怪了。

SECTION 46

L^AT_EX 的旋转表格

2020-03-05 13:51:19.334722

因为表格太长了，所以说有点时候需要把相关的表格横着放。

google 了很容易发现，有一个叫做`rotating`的 package 提供了 `sidewaystable`环境，提供了类似于自带的`table`功能，不过不同的只是它的表格是颠倒过来的。

我这一次使用了`[p]`选项让它放在新的一页。还使用了 `\caption`来生成图片，使用`\label`来为引用服务。

SECTION 47

令 cmd 接受参数

2020-03-05 18:29:41.111661

在 wsl 中可以通过 `cmd.exe` 来进入命令行，但是也可以传入参数来运行 ~

使用类似于`cmd.exe /C "command line"`来驱动程序。传入的命令完全是在 `cmd.exe` 中运行，所以说也可以读取到 `cmd.exe` 中的 `PATH`。通过这个完全可以用来让本来运行在 windows 的程序套个 sheel 壳子从而在 wsl 中运行 ~

开心中 ~

SECTION 48

在 wsl 中启动 jupyter-notebook

2020-03-06 18:11:44.615188

为了在 wsl 中使用 jupyter，那么一定需要一个浏览器。而 wsl 一开始就只有终端，并没有提供图形界面（我很喜欢）。为了使用它，可以使用`jupyter-notebook --no-browser`，它提供了一个端口和一长串 token。使用这两个就可以通过网络协议来使用！太棒了！

但是有的时候可能之前有东西占住了端口，8888是不行的了，也可以使用一个选项`--port=other-port`来自己定义一个端口。

SECTION 49

latex 关于\end{...}的报错问题

2020-03-07 12:14:05.564314

做数学作业的时候 latex 说有一个错误发生在\end{...}上，我百思不得其解，后来发现是一个环境只写了\begin{...}，但是没有写\end{...}，这就是原因☹。

所以说报错的地方可能和真实错误的地方有一定距离。

SECTION 50

cf's 1321C problem

2020-03-07 23:55:00.900381

我好像又把一道题目想复杂了☹。

题意如下：给定一个字符串，如果相邻的字符恰好比它本身小，那么将它移去。

答案是：从'z'开始遍历到'a'，如果遇到字符串中是遍历的这个数，而且恰好满足条件，那么将它移去。（☹！是简单的模拟！）

当然也不是什么都没有学到。我学到了string的一个 method，erase可以用来删除字符。

当然每一次找到了之后就从头找起.....（妈呀我感觉好气！我真的是一个大笨蛋！）

SECTION 51

cf's 1316B problem

2020-03-08 14:21:58.394338

老规矩先说题意。这道题给定了一个字符串（长度为 l ），和一个指定的动作：reverse子字符串。对数 k 而言，需要将它从 $[0, k)$ 一格一格的走到 $[l - k, l)$ ，每一次都将范围内的子字符串进行操作。

说说学到的几个操作：

定义别名！如何为一个 STL 定义别名呢？使用 using：

```
template<typename T>
using vec=vector<T>;
```

需要注意的是vec还是需要使用<type>，我好像有点理解了template的工作原理。

其次有一点需要注意，`string`是不能接受字符的，只能接受字符串。
话说，这道题对复杂度的要求还真是宽大啊。

SECTION 52

shell 脚本中的单引号和双引号

2020-03-09 14:47:36.004608

我想要自己搭建一个独特的 `promat`，所以为了自己的路径提示，我定义了一个函数，并且使用了双引号来定义我的 `PROMAT` 变量。

但是这样子有一个地方出乎我的意料，当我 `cd` 到另一个目录的时候，`promat` 应该进行更改，但是事实上却没有按照我的意图。

在 `shell` 脚本中双引号是会首先把本身的变量和运行命令一一运行和替换，而单引号不会。但是 `zsh` 本身在接收一个字符串之后会每次都又运行一遍，所以说我们应该传入一个没有被替换的文本串 ~

SECTION 53

cf's 1312C problem

2020-03-10 16:02:05.549992

给定一个目标数组，然后用指定的 `base` 通过 $base^{p_0} + base^{p_1} + \dots$ 来表示，如果数组里面每一个数组中的每个数构成的 p_0, p_1, \dots 互不相同，而且任意两个数中的 p_i 互不相同就可以说这个数组是一个好数组。

通过这个题我知道要注意两件事情：

53.1. 注意上限

原题中说的是数组中的数的取值范围是 $[0, 10^{16}]$ ，但是我还是傻乎乎的使用了 `int`（不然这道题我就过了啦）。

53.2. 关于本应该输出整数的浮点算法

在使用对数的时候，按理来说有点时候应该输出一个整数的，但是却输出了浮点数，而且之后无论本来应该是整数还是说是小数都需要过一遍`floor`函数，所以有些整数比如说4.9999998就会很不好意思的变成一个不应该变成的数。下面有三种方法：

1. 加上配重，比如说`1e-4`或者`1e-5`之类的。但是这种办法总感觉蛮容易被别人 hack 的。
2. 检验，得到对应的整数的时候进行检验，看看得到的符不符合预期。
3. 使用整数算法，比如使用除法得到对应的对数的解法。

SECTION 54

链表的使用

2020-03-11 08:44:46.127475

刚开始想要建立一个链表，但是其实上对 C 语言的管理不清不楚，以为和 python 一样，只要有引用计数，就会一直存在，但是不然，C 语言的话，只要是变量离开了它的定义域的话，就会被销毁，不仅仅是回收栈上面的内存，连对象的析构方法也会被一并调用。

我一开始用一个指针指向了一个局部变量，结果一出局部定义域，这个变量的空间就被标记为自由的内存栈空间，但是值却没有初始化，只是被标记了而已。

之后的话我又重新声明了一个局部变量（在另外一个定义域之内），好家伙，结果它用了前面一个的地址，链表就形成环了啦！我百思不得其解。

54.1. 动态的分法

在 C++ 中一般是通过`new`和`delete`来管理内存的新建和释放。在 C 语言中则是用`malloc`和`free`。

语法格式差不多是：

```
T* var = new T <initialization>;
delete var;
```

以上哈，还有对于空指针`nullptr`的话，`delete`是安全的操作。

SECTION 55

cf's 1324D problem

2020-03-13 16:58:11.221877

终于有一次让我做到了 D 题(虽然还是没有做出来),这道题是给定两组数据,分别是 $\{a_1, \dots, a_i, \dots, a_n\}$ 和 $\{b_1, \dots, b_i, \dots, b_n\}$, 如果下标 i 和 j 满足:

$$a_i + a_j > b_i + b_j \Rightarrow (a_i - b_i) + (a_j - b_j) > 0$$

求出可能的下标组合的个数。

我用笨办法来做, 先令 $c_i = a_i - b_i$, 那么对于每一个 i 而言, 我从正数找到最后一个可以让 j 的值满足 $c_i + c_j > 0$ 。那么累加起来, 可能的组合数就求出来了 (大体如此)。

答案是怎么做的呢? 好吧, 既然你要找到一个令 $c_j \geq 1 - c_i$ 的 j 值, 那么说二分搜索它不香吗? (指 `lower_bound`) 好吧一下子把 $O(n^2)$ 变成了 $(n \log n)$ 啦。

SECTION 56

关于 `map<bitset>` 的使用

2020-03-14 14:45:30.148432

众所周知, `map` 作为一个模板一般接受四个值, 而我一般只给定前两个, 而后面的值让模板自动补全即可。

今天我打算用 `bitset` 作为键, 但是编译器好像看不起 `bitset`, 潇潇洒洒给我了一千字的长文, 比我自己写的代码还多, 从头看到尾, 字里行间都是 `operator<`, 我寻思咋地啦, 我没有用到小于号啊, 你别报其他头文件的错啊, 你报这个的, 但别人不听, 还跟我抬起干来了。

在 `stack overflow` 上面使用了一些小花招, 搞到了答案, 一看, 原来作为红黑树是要通过小于号来进行比较的, 才能产生对应的树。(嗨! 我也不是不知道, 但还真没想到那方面去) 所以说, 我转念一想, 这个的意思是说, `bitset` 不支持大小比较咯? (嗨, 这不废话吗! 这玩意? 宁还真把它当数啦)

为了比较它这个类型, 我们必须给 `map` 多赋值一个类型, 这个类型的对象, 必须可以调用来比较两个 `key` 之间的大小关系。

差不多吧, 我边写边抄, 差不多代码如下:

```
template<size_t N>
struct bitset_less{
    bool operator() (const bitset<N> & a, const bitset<N> & b){
        if(N == 0) return true;
        size_t i = N - 1;
        do {
            if(a[i] ^ b[i])
```

```
        if(a[i]) return false;
        else return true;
    --i;
} while ( i > 0 )
}
```

将 `bitset_less` 传入代码即可。

在抄代码中首先使用 i 从 $N - 1$ 一直循环到 0，但是我忘记了 `size_t` 作为一个无符号正整数，是没有负数的，所以我设定的大于零这个条件是很成立的。（我在 [stack overflow](#) 上面问的第一道题）