

1 cf's 4A problem

2020-01-12 09:19:39.244631

divide a even integer 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.

2 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）

3 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 $\text{T}_\text{E}\text{X}$, 字体用配置指定的, 还是全部由 $\text{T}_\text{E}\text{X}$ 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。

4 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* \rightarrow *Orange* \rightarrow *White* \rightarrow *Lightgreen* \rightarrow *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。

5 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: `xtikzxfill[orange](1ex, 1ex)circle (1ex);` 可以画个圆。(这个 TikZ 太麻烦了。

而 `xtikz xdraw[->] 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。

6 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 `xpar` 。

注: VsCode 中用 `$nnumber` 来表示捕获的分组。

7 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。

8 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。

9 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。

10 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 `algorithm` 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 >
```

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

```
OutputIt transform( InputIt first1, InputIt last1, OutputIt d_first,
    UnaryOperation unary_op );
```

它会把 unary_op 作用到这些 $[frist1, last1)$ 上去，而输出到 $[frist2, +\infty)$ 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。

11 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)。

12 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 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）。

13 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。

14 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？

15 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。

16 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。

16.1 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 \log_{10} ! 我真傻, 真的!

(手动狗头)

17 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 $f(\text{len}, \text{range})$, 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欸, 真是太笨了。

17.1 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?

18 XeTeX 的命令乱码

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？

19 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。

19.1 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。

19.2 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});
```

19.3 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。

19.4 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。

20 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。

20.1 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。

20.2 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。

21 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`。

22 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）

22.1 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。

22.2 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。

23 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[2]`。

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。

24 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。

24.1 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。

25 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。

25.1 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。