- Testcase-Tools
  - 前期准备:
  - 1. 测试用例生成器
  - 2. 比较器
  - 3. SPJ(Special Judge)检查器

Testcase-Tools

English Readme

由Cyaron提供支持的测试用例工具集

前期准备:

1. 运行 `pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple`
2. 创建一个文件夹来保存问题所需的所有文件，例如，我创建了 `generator_example\`

你需要在此目录中放置两个文件：`Config.py`和 `std.cpp`

你需要这样放置它们：

```
generator_example\
  |- Config.py
  |- std.cpp
```

并且分别复制 `Config_template.py`和 `std_template.cpp`的内容到 `Config.py` 与 `std.cpp`中

或者你可以使用命令 `python FastInit.py {路径名称}`来快速初始化一个问题目录，例如: `python FastInit.py fastinit_example`

3. 自定义Config.py 在 `Config.py`中，您需要自定义方法 `Gen.generator`，它应该做以下两件事之一：
   1. 返回测试用例输入的字符串内容，例如：

```python
from cyaron import *
class Gen:
    @staticmethod
    def generator(data_group: int, io: IO) -> str:
        return '114514' # 将 '114514' 输出到 {data_group}.in
```

2. 将测试用例数据写入 `io`（推荐使用），例如：

```python
from cyaron import *
from libs.glib import ToolSet

class Gen:
    @staticmethod
    def generator(data_group: int, io: IO) -> None:
        a = randint(ToolSet.INT_MIN, ToolSet.INT_MAX)
        b = randint(ToolSet.INT_MIN, ToolSet.INT_MAX)
        io.input_writeln(a, b)
        # 将 a、b 输出到 {data_group}.in, 在一行内且以一个空格隔开
```

同时 `Config.py`中也有一些可以调节的参数, 具体如下所示:

```
基本功能:
std_name = 'std.cpp'   标准c++程序的代码名称
data_set = 20          要生成的数据组数

实验性功能(不推荐更改):
no_gen = False         不生成输入文件
genOut = False         在Gen.generator中生成输出文件而不是用{std_name}.cpp
```

具体的IO操作以及Cyaron工具函数、类, 请参照Cyaron Wiki

4. 自定义std.cpp 在 `std.cpp`中，您只需要编写一个可以解决问题的c++标准程序

您不需要重定向stdin流或stdout流

问题目录内部可以没有 `std.cpp`，你可以在 `Config.py`中将 `genOut`设置为`True`，并且在 `Gen.generator`中写上输出，但是这仍然是一个实验性功能

1. 测试用例生成器

用法：`python general.py {folder_name}`

用例：`python general.py generator_example`

测试用例将自动生成并压缩，文件夹将如下所示：

```
generator_example\
  |- Config.py
  |- std.cpp
  |- std.exe
  |- 1.in
  |- 1.out
  |- 2.in
  |- 2.out
  | ....
  |- generator_example.zip
```

2. 比较器

使用比较器，您可以用同一个数据比较两个cpp代码的输出

用法：`python compare.py {Data Generator} {program1} {program2} {(Optional) Number of runs}`

用例1：`python compare.py generator_example compare_example/1.cpp compare_example/2.cpp`

此用例为对拍两份程序直到发生错误或答案不一致

用例2：`python compare.py generator_example compare_example/1.cpp compare_example/2.cpp 100`

此用例为对两份程序直到发生错误或答案不一致或超出运行次数限制(100次)

注意: 此处的程序路径为相对路径

3. SPJ(Special Judge)检查器

使用spj检查器，您可以生产输入数据并批量spj您的程序

用法：`python spj_checker.py {Data Generator} {spj_program} {tested_program} {(Optional) Number of runs}`

用例1：`python compare.py generator_example spj_example/spj.cpp spj_example/to_test_1.cpp`

此用例为spj程序直到发生错误或答案不正确

用例2：`python compare.py generator_example spj_example/spj.cpp spj_example/to_test_1.cpp 100`

此用例为spj程序直到发生错误或答案不正确或超出运行次数限制(100次)

关于如何写spj: 戳我

注意: 此处的程序路径为相对路径