1 Введение

В данной задаче требуется написать программу для управления космической ракетой. Программа будет носить временное название «Умный Пилот». Цель полёта — взлететь с поверхности Земли и выйти на эллиптическую орбиту с перигейным расстоянием не меньше заданного. Перигей — точка орбиты, ближайшая к центру Земли, а перигейное расстояние — расстояние от перигея до центра Земли.

Система запуска ракеты происходит следующим образом. Всё время полёта от старта и до выхода на орбиту Земли разбивается на $T = 1000$ игровых секунд. В начале каждой секунды программа-интерактор собирает текущие физические характеристики ракеты и передаёт их на вход «Умному Пилоту». Задача «Умного пилота» — решить, каким образом продолжать движение ракеты в течение следующей игровой секунды. Для этого «Умный Пилот» передаёт программе-интерактору управляющие параметры, о которых будет рассказано далее.

Число баллов, полученных за решение, тем выше, чем больше топлива останется в ракете к моменту выхода на требуемую орбиту.

2 Строение ракеты

В начальный момент времени ракета состоит из n пронумерованных от 1 до n ступеней, и *активной* ступенью является ступень номер n. Ракета летит, сжигая топливо в активной ступени, пока полностью его не израсходует. Когда топливо в активной ступени кончается, ступень отсоединяется от ракеты, и активной ступенью становится ступень с номером на 1 меньше. Если топливо закончилось в ступени номер 1, то она не отсоединяется, и движение продолжается без возможности использовать двигатели.

Ступень номер i обладает массой m_i (без учёта массы топлива), массой находящегося в ней топлива f_i, длиной l_i, максимальным потреблением топлива c_i и удельным импульсом p_i. В каждый момент времени может работать двигатель лишь активной ступени. Длиной ракеты L считается суммарная длина всех её ступеней, которые ещё не отсоединились. Ракета также обладает площадью поперечного сечения S, постоянной по всей длине ракеты.

3 Управление

Каждую секунду «Умный Пилот» получает на вход текущее состояние ракеты. От «Умного Пилота» требуется выдать пару чисел α и ϵ.

Параметр α — вещественное число от 0 до 1 (безразмерное). Грубо говоря, этот параметр задаёт, на какую долю от максимума должен работать двигатель. Точнее, если активная ступень i имеет максимальное потребление топлива c_i, то двигатель будет работать с реальным потреблением топлива $\alpha \cdot c_i$. Если в течение следующей секунды некоторые ступени отсоединятся, то следующие за ними ступени продолжат работать с тем же параметром α.

Параметр ϵ — это вещественное число от -0.1 до 0.1 (в радианах на секунду в квадрате). Оно задаёт угловое ускорение ракеты в течение следующей секунды. На вращение ракеты топливо не тратится.

4 Физика

Мир этой задачи двумерный. В нём заведена декартова система координат xy. Все физические величины задаются в системе СИ: длина в метрах, угол в радианах, время в секундах, масса в килограммах.

Ракета считается материальной точкой. Это значит, что её положение в пространстве задаётся единственной точкой, и в этой же точке прикладываются все силы, действующие на ракету. Длина же ракеты и площадь её сечения будут учитываться лишь при рассчёте силы сопротивления воздуха. У ракеты есть направление — угол её наклона к оси x, отсчитанный против часовой стрелки. Направление ракеты задаёт направление силы тяги двигателей.

В каждый момент времени положение ракеты в пространстве характеризуется 4-мя параметрами: позицией в пространстве p, скоростью v, направлением φ, скоростью изменения направления ω. Параметры φ и ω также называются текущим углом и угловой скоростью. Первые два параметра изменяются под действием сил. По второму закону Ньютона ускорение $a = \frac{F_{\text{тяж}}+F_{\text{сопр}}+F_{\text{двиг}}}{m}$, где m — текущая масса ракеты. Три силы $F_{\text{тяж}}$, $F_{\text{сопр}}$ и $F_{\text{двиг}}$, действующие на ракету, — сила тяжести, сила сопротивления воздуха и сила двигателей ракеты соответственно. Параметры же φ и ω изменяются под действием углового скорения ϵ, задаваемого «Умным Пилотом».

Земля представляет собой круг с центром в точке $(0, 0)$ и радиусом $R = 6.378 \times 10^6$. Земля имеет массу $M = 5.973 \times 10^{24}$. Ракета притягивается к центру Земли с силой $|F_{\text{тяж}}| = G\frac{Mm}{r^2}$. Здесь $G = 6.674 \times 10^{-11}$ — гравитационная постоянная, m — масса ракеты, а r — расстояние от неё до центра Земли. В начальный момент $|F_{\text{тяж}}|$ считается равной нулю, так как она полностью компенсируется силой сопротивления поверхности Земли.

Земля окружена воздушным слоем. Воздушный слой присутствует на высоте меньше $K = 10^5$ над поверхностью Земли. На высоте K проходит линия Кармана. Выше этой линии начинается безвоздушное пространство. Ниже — на ракету действует сила сопротивления воздуха. Для её рассчёта представим ракету цилиндром длины L с кругом площади S в основании. Направление ракеты при этом сонаправлено с осью цилиндра. Заметаемая ракетой площадь S' равна площади проекции этого цилиндра на плоскость, перпендикулярную скорости ракеты. Вычислить силу сопротивления воздуха можно по формуле $|F_{\text{сопр}}| = C_1 S' |v|^2 (1 - h/K)^{C_2}$. Направление силы сопротивления воздуха противоположно скорости ракеты. Известны константы $C_1 = 0.416$ и $C_2 = 5.256$.

Последняя сила — сила двигателей ракеты $|F_{\text{двиг}}| = \alpha c_i p_i$, направленная по направлению ракеты. Здесь i — номер активной в данный момент ступени. Если же все ступени, кроме ступени 1 отсоединились, и в ступени 1 топливо закончилось, то сила двигателей равна нулю.

Ракета начинает свой полёт на поверхности Земли в точке $(R, 0)$, с нулевой скоростью, с нулевым направлением (сонаправленно оси x) и нулевой угловой скоростью.

Перигейное расстояние определяется в произвольный момент времени следующим образом. Если в этот момент полностью отключить все двигатели ракеты, а также отключить сопротивление воздуха и разрешить пролетать сквозь Землю, то ракета продолжит лететь по некоторой траектории. Если эта траектория является эллипсом, тогда минимальное расстояние от этого эллипса до центра Земли считается перигейным расстоянием. Перигейное расстояние в какие-то моменты времени может быть меньше радиуса Земли, однако врезаться в Землю вашей ракете категорически запрещается. Запуск ракеты завершается успешно в тот момент, когда перигейное расстояние оказывается больше заданного значения.

5 Протокол взаимодействия

5.1 Начало

Сначала программа-интерактор сообщает «Умному Пилоту» начальные данные задачи, имеющие следующий формат:

```
desired_perigee [требуемое перигейное расстояние]
duration_in_seconds [длительность полёта в секундах]
rocket_diameter [диаметр ракеты]
stages_number [число ступеней n]
```

В этом описании:

- `[требуемое перигейное расстояние]` — вещественное число, большее или равное $(R + K)$, где R — радиус Земли, а K — линия Кармана. Ваша ракета должна достичь перигейного расстояния не меньше этого числа — тогда полёт завершится успешно.
- `[длительность полёта в секундах]` — всегда равно 1000. По истечении этого времени полёт прекращается неуспешно.
- `[диаметр ракеты]` — вещественное положительное число, которое влияет на сопротивление воздуха (подробности в разделе «Физика»).
- `[число ступеней n]` — целое положительное число, определяющее начальное количество ступеней в ракете.

Затем программа-интерактор подаёт решению на вход описания ступеней ракеты по порядку от 1-й до n-й по одной ступени на строке. Описание i-й ступени выглядит так:

```
empty_mass [m_i] fuel [f_i] length [l_i] consumption [c_i] impulse [p_i]
```

Все значения в этой строке — вещественные положительные числа. Здесь:

- m_i — масса i-ой ступени без топлива.
- f_i — масса топлива, изначально залитого в i-ую ступень.
- l_i — длина i-ой ступени.
- c_i — максимально возможное потребление топлива двигателя i-ой ступени.
- p_i — удельный импульс двигателя i-ой ступени.

Дальше совместная работа «Умного Пилота» и программы-интерактора происходит по итерациям. Каждая итерация соответствует одной секунде полёта.

5.2 Итерация

На каждой итерации на вход «Умному Пилоту» сначала подаётся текущий статус:

```
status [текущий статус]
```

Здесь `[текущий статус]` — это строка, принимающая одно из значений:

- `crashed` — ракета врезалась в Землю, запуск закончился неуспешно.
- `finish` — либо ракета достигла заданного перигейного расстояния и запуск закончился успешно, либо истекло отведённое на полёт время и запуск закончился неуспешно.
- `moving` — полёт ракеты продолжается, требуется ввести управление.

Получив статус, отличный от `moving`, ваша программа должна немедленно завершить работу. В противном случае ваше решение может получить нежелательный или неожиданный вердикт.

Если статус отличен от `crashed`, то программа-интерактор после статуса выдаёт текущее состояние ракеты в формате:

```
time [время в секундах]
perigee [перигейное расстояние]
stages_left [число оставшихся ступеней]
last_stage_fuel [масса топлива в последней ступени]
position [координата x] [координата y]
velocity [проекция скорости на ось x] [проекция скорости на ось y]
direction [текущий угол]
angular_velocity [угловая скорость]
```

Данные поля имеют следующие значения:

- `[время в секундах]` — сколько секунд полёта уже прошло (целое число).
- `[перигейное расстояние]` — текущее перигейное расстояние, или -1, если текущая орбита не является эллипсом. Подробно определение перигейного расстояния описано в разделе «Физика».
- `[число оставшихся ступеней]` — сколько ступеней осталось в ракете на текущий момент (целое число), остальные уже отброшены.
- `[масса топлива в последней ступени]` — сколько килограмм топлива сейчас имеется в активной ступени.
- `[координаты x/y]` — текущее положение ракеты.
- `[проекции скорости на оси x/y]` — текущий вектор скорости ракеты, заданный в виде x и y компонент.
- `[текущий угол]` — направление ракеты, то есть угол наклона относительно оси x.
- `[угловая скорость]` — скорость вращения ракеты, то есть производная текущего угла по времени.

Если не написано обратное, то все значения в этом блоке вещественные.

При получении статуса `moving` ваше решение должно вывести в выходной поток два вещественных числа φ и ω, разделённые пробелом. Значение этих чисел и ограничения на них описаны выше в разделе «Управление». После этого программа-интерактор промоделирует ещё одну секунду полёта и начнётся новая итерация.

6 Система оценки

Каждый тест задачи — входной файл с описанием ракеты. По каждому тесту каждой задачи оценивание проходит независимо.

Ограничение на время работы решения на одном тесте — **2 секунды**.

Ограничение на размер выделяемой решением памяти на одном тесте — **512 МБ**.

6.1 Предварительная оценка

Сначала все участники получают за тест *предварительную оценку*. Предварительная оценка равна -1 в следующих случаях:

1. Если ракета врезалась в Землю или не вышла на нужную орбиту к концу отведённого времени. В этом случае решение получает вердикт Wrong Answer.

2. Если решение превысило ограничение на суммарное время работы, превысило ограничение на объём потребляемой памяти, или завершилось с ненулевым кодом возврата, например в результате критической ошибки. Во всех этих случаях решение получает такое же вердикт, который оно обычно получает на соревнованиях по правилам ACM.

3. Если решение неверно взаимодействует с программой-интерактором, то оно получает вердикт Timeout. Это может произойти, например, если решение не делает `flush` выходного потока после вывода команды, или если решение читает данные в неверном формате, или если решение читает данные после получения статуса `finish` или `crashed`.

4. Если участник не отправил никакое решение по задаче с этим тестом.

Решение, успешно вышедшее на эллиптическую орбиту с требуемым перигейным расстоянием, получает на тесте вердикт ПРОВЕРЕНО. Предварительная оценка за тест равна отношению массы топлива, оставшегося в ракете к концу полёта, к начальной массе топлива в ракете. Таким образом предварительная оценка при успешном завершении — это вещественное число от 0 до 1.

6.2 Баллы

После того, как всем участникам выставлены предварительные оценки за тест, вычисляются *баллы*, которые они за него получают. Участник, не отправивший по задаче решения, дошедшего до процесса тестирования, получает 0 баллов. Участник, чьё решение участвовало в тестировании, получает за тест $\frac{1+t}{k}$ баллов, где t — количество участников, у которых предварительная оценка строго меньше, чем у него, а k — общее количество участников.

6.3 Задачи

Вы можете отправлять ваше решение во время тура в четыре разные задачи.

Задачи «Ракета 1», «Ракета 2» и «Ракета 3» содержат ровно по одному «открытому» тесту. Три этих теста вам **известны**: они есть в архиве материалов. По каждой из этих задач в зачёт идёт **лучший** результат среди всех ваших посылок, причём это верно как при построении финального рейтинга, так и текущего рейтинга по время тура.

Задача «Неизвестная Ракета» содержит 100 тестов, сгенерированных псевдослучайным образом. Генератор, которым генерируются эти тесты, включён в архив материалов. В течение тура в системе тестирования будут лежать тесты, которые сгенерированы этим генератором с seed = 0. Вы можете сгенерировать эти тесты самостоятельно, запустив генератор без указания seed.

Обратите внимание, что ваши результаты в течение тура по задаче «Неизвестная Ракета» **предварительные** ! После конца тура жюри сгенерирует по этой задаче другой набор тестов, запустив этот же генератор, но с другим (секретным) значением seed. В финальное тестирование будет взято только **последнее** решение, отправленное вами по этой задаче. Все эти решения будут протестированы на новом наборе тестов и баллы за задачу будут пересчитаны.

6.4 Рейтинг

Итоговый результат участника за тур вычисляется как:

$$700 \cdot (S_1 + S_2 + S_3) + 9 \cdot S_0$$

Здесь S_1, S_2 и S_3 — это баллы, полученные за единственный тест по задачам «Ракета 1», «Ракета 2» и «Ракета 3» соответственно, а S_0 — это сумма баллов по всем ста тестам в задаче «Неизвестная Ракета».

Получается, что:

1. итоговый результат лежит в диапазоне $[0, 3000]$,
2. итоговый результат на 70% определяется открытыми тестами по первым трём задачам.

Участники сортируются по итоговому результату в порядке убывания. Чем больше итоговый результат, тем выше место в рейтинге и тем лучше для участника.

6.5 Финальное тестирование

После завершения тура тесты по задаче «Неизвестная Ракета» будут перегенерированы и будет запущено финальное тестирование по этой задаче, в результате которого определятся баллы участников по задаче.

По каждой из задач «Ракета 1», «Ракета 2» и «Ракета 3» для каждого участника будет выбрана максимальная предварительная оценка, достигнутая им в течение тура. По этим предварительным оценкам будут вычислены баллы участников. Перетестирования по этим задачам **не** будет.

Итоговый результат и финальный рейтинг будут определены по баллам в точности так, как описано выше.

7 Материалы

Для разработки и тестирования «Умного Пилота» предоставляется архив материалов. Его можно скачать в системе тестирования nsuts на вкладке «Новости». Рекомендуется распаковать содержимое архива в `D:\tools` и работать в этой же директории.
Содержимое архива:

- `interactor.exe` — исполняемый файл программы-интерактора.
- `src\*` — исходный код программы-интерактора и визуализатора.
- `run.py` — скрипт для запуска решения вместе с интерактором.
- `vis.py` — скрипт для визуализации журнала последнего запуска.
- `gen.py` — скрипт для генерации тестов по задаче «Неизвестная Ракета».
- `statement.pdf` — условие задачи, которое вы сейчас читаете.
- `rocket1.txt` — открытый тест по задаче «Ракета 1».
- `rocket2.txt` — открытый тест по задаче «Ракета 2».
- `rocket3.txt` — открытый тест по задаче «Ракета 3».

Кроме того, в архиве есть примеры решений:

- `sol1cpp.cpp` — пример решения задачи «Ракета 1» на языке C++.
- `sol2cpp.cpp` — пример решения задачи «Ракета 2» на языке C++.
- `sol3cpp.cpp` — пример решения задачи «Ракета 3» на языке C++.
- `sol1java.java` — пример решения задачи «Ракета 1» на языке Java.
- `sol2java.java` — пример решения задачи «Ракета 2» на языке Java.
- `sol3java.java` — пример решения задачи «Ракета 3» на языке Java.

Ниже приведено несколько команд для быстрого старта.

Скомпилировать пример решения на C++ по задаче «Ракета 1» и запустить его, а потом показать журнал запуска в визуализаторе:

```
cl sol1cpp.cpp
run rocket1.txt sol1cpp
vis
```

Скомпилировать пример решения на Java по задаче «Ракета 2» и запустить его, а потом показать журнал запуска в визуализаторе:

```
javac sol2java.java
run rocket2.txt java sol2java
vis
```

7.1 Система тестирования

Также рекомендуется сразу зайти в систему тестирования nsuts и отправить по всем 4 задачам тура пример решения: `sol1cpp.cpp` или `sol1java.java`. После этого зайдите на вкладку «Результаты».

По открытым задачам должно быть видно вашу предварительную оценку за единственный тест, а по задаче «Неизвестная Ракета» — строку вердиктов, полученных на тестах. Кроме того, по каждому тесту каждой задачи вы можете скачать архив с журналом полёта, или запустить визуализатор прямо в браузере, не скачивая архив.

Чтобы запустить визуализатор прямо в браузере, кликните по предварительной оценке для открытой задачи или по столбику в диаграмме для задачи «Неизвестная Ракета».

7.2 Запуск решения

Решение можно запустить с помощью скрипта `run.py`, используя командную строку:

```
run [тест] [решение]
```

Здесь:

- `[тест]` — путь к тесту, то есть к текстовому файлу со входными данными. Например, можно указать `rocket1.txt`, чтобы запустить решение на тесте из задача «Ракета 1».
- `[решение]` — исполняемый файл решения, которое надо запустить. Например, для примера решения на C++ можно указать `sol1cpp.exe` (нужно предварительно скомпилировать решение). Для решения на Java нужно написать два слова: `java [имякласса]`. Если не указывать аргумент `[решение]`, то программа-интерактор будет работать со стандартными потоками ввода-вывода, и можно будет "общаться" с ней самостоятельно через консоль.

В случае успешного запуска в директории создаётся текстовый файл `log.txt` — журнал запуска. Его можно просмотреть в визуализаторе, используя скрипт `vis.py`:

```
vis [журнал]
```

Здесь `[журнал]` — путь к текстовому файлу, откуда будет читаться журнал запуска. Этот параметр можно опустить, тогда журнал будет читаться из `log.txt`.

Визуализатор открывается в браузере Firefox. В визуализаторе можно наблюдать движение ракеты в течение всего времени полёта, следить за её положением, направлением, скоростью и другими параметрами. Можно изменять скорость работы, ставить на паузу, перематывать время.

7.3 Генератор gen.py

Скрипт `gen.py` позволяет сгенерировать тесты по задаче «Неизвестная Ракета»:

```
gen [seed] [директория] [кол-во тестов]
```

Скрипт принимает три опциональных параметра:

- `[seed]` — магическое число, полностью определяющее результаты генерации. По умолчанию равен нулю: в этом случае генерируются в точности те тесты, которые используются в задаче «Неизвестная Ракета» в течение тура. Для финального тестирования жюри перегенерирует тесты по этой задаче, используя другое магическое число.
- `[директория]` — в какую директорию писать тесты. По умолчанию тесты записываются в директорию `many`.
- `[кол-во тестов]` — сколько тестов генерировать. По умолчанию 100 штук.

В генерируемых тестах некоторые величины постоянные, а некоторые генерируются случайным образом с равномерным распределение по какому-то диапазону. Все величины генерируются целыми.

- `desired_perigee` = 6480000
- `duration_in_seconds` = 1000
- `stages_number` = 4

Случайные величины:

- `rocket_diameter` $\in [10, 50]$
- `empty_mass` $\in [1000, 10000]$
- `fuel` $\in [1000, 40000]$
- `length` $\in [5, 100]$
- `consumption` $\in [250, 400]$
- `impulse` $\in [15000, 40000]$