Mizzza54

Яндекс, Осенняя стажировка 2021

1 авг 2021, 16:47:21
старт: 1 авг 2021, 09:53:16
финиш: 1 авг 2021, 15:53:16
длительность: 06:00:00

F. В поисках сломанного коммита

Язык	Ограничение времени	Ограничение памяти	Ввод	Вывод
Все языки	3 секунды	512Mb	стандартный ввод или input.txt	стандартный вывод или output.txt
Python 3.9.1	6 секунд	512Mb		
Python 2.7 (PyPy 4.0.0)	6 секунд	512Mb		
Python 3.7 (PyPy 7.3.3)	6 секунд	512Mb		
Python 2.7	6 секунд	512Mb		

В одной небезызвестной компании, менеджер старшего разряда Андрей с командой создают новый продукт « Он.Компановщик ». Как и всегда, разработка большого проекта предполагает множество коммитов в репозиторий с проектом, количество которых временами доходит до 100 в день. Так как производить тестирование проекта после каждого коммита не представляется возможным, то в самый неудобный для этого момент (по всем законам жанра ужасов) проект перестал проходить тесты. Молодой, но опытный менеджер Андрей поставил перед собой задачу найти тот коммит, который всё ломает.
Андрей знает, что коммиты в репозитории делятся на два типа: **большие** «B» (big) и **маленькие** «S» (small). Также Андрей знает, что время пересборки проекта зависит только от того, был ли большой коммит между ревизиями сборки. Например, если последовательность коммитов выглядит как «BSBS», а у Андрея есть в текущий момент собранный проект после последнего (4-го) коммита, то он может собрать проект после 3-го коммита за a минут, так как четвёртый коммит маленький, а проект после второго коммита за b минут, так как 4-й коммит маленький, но 3-й большой.

Более формально, если в текущий момент проект собран после i-го коммита, а Андрей хочет собрать проект после j-го коммита, то пересборка займёт b минут, если между коммитами i и j есть большой коммит, и a минут в противном случае.

К сожалению, в сфере сборки Андрей ещё слишком неопытен, поэтому он может хранить только один бинарный файл (бинарь) проекта единовременно, и сейчас перед ним собранный бинарь после последнего коммита, который гарантированно не проходит тесты. Можно считать, что бинарь перед первым коммитом был протестирован и все тесты проходил. Заметьте, что в отличие от сборки, время тестирования собранного бинаря равно нулю.

Помогите Андрею рассчитать, за какое минимальное гарантированное время он может найти сломанный коммит.

Гарантируется, что среди коммитов есть ровно один ломающий, а также, что ни один коммит не чинит код после поломки. Иными словами, для некоторого k код проходит тесты после первых k коммитов и не проходит их в дальнейшем.

Формат ввода

В первой строке вводятся три числа n, a, b ($1 \le n \le 3000$, $1 \le a \le b \le 1000$) — количество коммитов в проекте, время сборки только из маленьких коммитов и время сборки, если есть большой коммит соответственно.
Вторая строка содержит n символов «B» и «S», соотвествующие размерам коммитов.

Формат вывода

В единственной строке выведите минимальное гарантированное время, за которое можно найти сломанный коммит.

Пример 1

Ввод	Вывод
5 1 100 SBSSS	101

Пример 2

Ввод	Вывод
8 2 4 BBSSBBBS	12

Пример 3

Ввод	Вывод
13 324 845 SSSBSSBBSSSBB	2859

Примечания

В первом примере из условия собран бинарь после последнего коммита, будем обозначать текущий бинарь символом «x».

1. 0 минут состояние «SBSSSx».
2. Сборка релиза после второго коммита будет занимать 1 минуту, так как разница между ревизиями состоит только из трёх маленьких коммитов («SSS»).
3. 1 минута состояние «SBxSSS».
4. В зависимости от того, проходит ли тесты данная ревизия, следующий шаг должен быть либо «SxBSSS», либо «SBSxSS»). В первом случае пересборка будет занимать 100 минут, а после её тестирования мы однозначно поймём, какой именно коммит содержит ошибку.

   Во втором случае пересборка занимает 1 минуту, но однозначно определить коммит получится не всегда, так как будет необходима сборка ревизии «SBSSxS», если ревизия «SBSxSS» не содержит ошибок. В итоге максимальное время проверки будет $\max(1+100, 1+1, 1+1+1) = 101$.

Во втором примере из условия изначальное состояние «BBSSBBBSx».

1. Сначала собираем за 4 минуты ревизию «BBSSxBBBS».
2. Если в ревизии содержится ошибка, то собираем «BBxSSBBBS» за 2 минуты, а дальше либо «BxBSSBBBS» за 4, либо «BBSxSBBBS» за 2 и находим нужный коммит.
3. Если в ревизии не содержится ошибка, то собираем «BBSSBBxBS» за 4 минут, дальше либо «BBSSBxBBS» за 4, либо «BBSSBBBxS» за 4 и находим нужный коммит.
4. В итоге максимальное время проверки будет $\max(4+2+4, 4+2+2, 4+4+4, 4+4+4) = 12$.

Язык Oracle Java 8

Набрать здесь | Отправить файл

```java
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.NavigableSet;
import java.util.StringTokenizer;
import java.util.TreeSet;

/**
 * @author Michael Gerasimov
 */
public class BigAndSmallCommits {
    FastReader scanner = new FastReader(System.in);
    int n, a, b;
    char[] history;
    NavigableSet<Integer> bigIndexes = new TreeSet<>();

    private void run() {
        n = scanner.nextInt();
        a = scanner.nextInt();
        b = scanner.nextInt();
        history = scanner.next().toCharArray();
        for (int i = 0; i < history.length; i++) {
            if (history[i] == 'B') {
                bigIndexes.add(i);
            }
        }
        System.out.println(solve(0, n - 1, 0, Trend.LEFT));
    }

    private int solve(int l, int r, int sum, Trend trend) {
        if (l == r || r < 0 || l > r) {
//            switch (trend) {
//                case LEFT:
//                    sum -= history[l + 1] == 'S' ? a : b;
//                    break;
//                case RIGHT:
//                    sum -= history[l - 1] == 'S' ? a : b;
//                    break;
```

Отправить

Предыдущая