

Testing und Debugging: Übung 12

Abgabetermin 14.01.2014 13:59

Name

Matrikelnummer

1 Testfall Vereinfachung

In der Vorlesung haben Sie bereits gesehen, dass *Delta Debugging* im schlimmsten Fall eine Laufzeit hat die quadratisch in der Eingabelänge ist.

1.1 Die Laufzeit im besten Fall (10 Punkte)

Sei l die Länge der Eingabe.

IST DIE LAUFZEIT IM BESTEN FALL:

- Linear in der Eingabelänge: $\text{asciimath}:[O(l)]$.
- Logarithmisch in der Eingabelänge: $\text{asciimath}:[O(\log(l))]$.
- Linear-Logarithmisch in der Eingabelänge: $\text{asciimath}:[O(l*\log(l))]$.

Begründen Sie Ihre Antwort (z.B. zu welcher Strategie degeneriert Delta Debugging in diesem Fall)!

•
•
•
•
•
•
•

1.2 Unter welchen Voraussetzungen tritt der Fall mit der bestmöglichen Laufzeit ein? (10 Punkte)

- Alle Tests geben *FAIL* zurück.
- Es gibt in jeder der vorkommenden Zerlegung immer eine Hälfte die zu einem *FAIL* führt.
- Alle Tests geben *PASS* zurück.
- Keine, der hier angegebenen Voraussetzungen führen zur bestmöglichen Laufzeit.

•
•

1.3 Welche der Möglichkeiten (a-c) der vorangegangenen Aufgabe kann niemals auftreten? (10 Punkte)

Begründen Sie ihre Antwort:

•
•
•
•
•

1.4 Beobachten des Laufzeitverhaltens. (80 Punkte)

In der Programmieraufgabe sollen Sie acht Testfälle, die mittels eines Randomtesting in Übung 06 erzeugt wurden, vereinfachen. Geben Sie für jeden Testfall folgende Informationen an:

1. Anzahl der API Befehle im gegebenen Testfall.
2. Mögliche beste Laufzeit.
3. Mögliche schlechteste Laufzeit.
4. Anzahl der für die Minimierung notwendigen Aufrufe der *test* Funktion.
5. Ausgehend von den Informationen in 1. - 4.: Ist der aktuelle Lauf von *ddmin* eher bei den best Möglichen oder bei den schlechtes Möglichen Läufen einzuordnen.

test_Queue1:

.
.
.
.
.

test_Queue2:

.
.
.
.
.

test_Queue3:

.
.
.
.
.

test_Queue4:

.
.
.
.
.

test_Queue5:

.
.
.
.
.

test_Queue6:

.
.
.
.
.

test_Queue7:

•
•
•
•
•

test_Queue8:

•
•
•
•
•

2 Programmieraufgabe: Anwenden von *ddmin* (100 Punkte)

In der Vorlesung 12 wurde der *ddmin* Algorithmus besprochen und anhand einer Python implementation (vgl. Folien zur Vorlesung 12: *vorlesung12/python/Bugreport_24735_(4,_Delta_Debugging).py*) praktisch angewendet.

Der Algorithmus ist sehr generisch aufgebaut und benötigt zur Benutzung für ein spezielles Problem lediglich eine Funktion *Names test(s)* die für einen gegebenen String oder Liste entscheidet, ob dieser Eingabewert (*s*) den Test besteht (*return "PASS"*) oder nicht besteht (*return "FAIL"*).

Dateien für die Programmieraufgabe



Grading/initpy
Grading/Grading.py
uebung12.py *

Hinweisdatei für Python: Das Verzeichnis ist ein Modul
Quellcode der Testklasse, die in *run_test.py* verwendet wird
In dieser Datei müssen Sie die Funktionen *test()* und *minimize_all()* gemäß Aufgabenstellung implementieren
Zu vereinfachende Testfälle.

test_Queue1 -
test_Queue8
test_min_Queue1 -
test_min_Queue8 *

Vereinfachte Testfälle (erst nach dem Lauf von *python3 uebung12.py* vorhanden, sofern die Funktionen richtig implementiert wurden).

Die mit [*] markierten Dateien müssen **bearbeitet** und **abgegeben** werden.

2.1 Ihre Aufgabe

Implementieren Sie die Funktion *test(s)* so, dass sie den gegebenen API Strom *S* ausführt und entscheidet ob der Lauf einen Fehler in der jeweiligen Queue ausgelöst hat oder nicht.

Implementieren Sie die Funktion *minimize_all()*. Die Funktion soll für jede Queue:

- den gegebene Testcase einlesen (falls vorhanden!)
- ihn vereinfachen
- prüfen, ob der vereinfachte Testfall immer noch den Fehler auslöst
- nach erfolgreichem Test den vereinfachten Testfall abspeichern.



Achtung

Benutzen Sie für die zu ladenden und speichernden Dateien die Dateinamen die wie sie in den jeweiligen Variablen in *minimize_all()* stehen.

**Tipp**

Für Aufgabe 1.4. kann `mimize_all()` **hilfreiche** Bildschirmausgaben enthalten. Beschränken Sie sich bei den Ausgaben je Testfall auf max. 10 Zeilen.

2.2 Bewertung

Es werden die acht minimierten Testfälle geprüft und acht neue Testfälle minimiert.

2.3 Abgabe der Programmieraufgabe

1. Falls noch nicht vorhanden, dann erstellen Sie sich auch <https://github.com> einen Account. (Pro Gruppe ist nur ein Account und eine Abgabe nötig!)
2. Schicken Sie bei Ihrer ersten Abgabe den **Namen** des github Accounts (ohne Passwort) an sebastian.stigler@htw-aalen.de mit dem Betreff: *Testing und Debugging Github Account*
3. Gehen Sie in Ihrer virtuellen Maschine in das Aufgabenverzeichnis (wo sich Ihre bearbeitete Aufgabe und die Datei `submit.cfg` befinden).
4. Tippen Sie in der Konsole den Befehl `submit`. Dieser wird beim ersten Ausführen nach den Zugangsdaten Ihres github Accounts fragen. Anschließend werden die bearbeiteten Aufgaben verschlüsselt auf <https://gist.github.com> abgelegt.
5. Diese Datei wird nach dem Abgabetermin automatisch zur Korrektur heruntergeladen.

Viel Erfolg
