

## 1 Dingen

- Momenteel gradle dus werkt over alle CI systemen heen aangezien ze allemaal gradle supporten
- Leg architectuur uit: naar server, orde bepalen met python predictors

## 2 Algoritmes

### 2.1 All in order

allemaal op volgorde van toevoeging aan database

### 2.2 All random

allemaal in random volgorde

### 2.3 Affected in order

al degene waarvan gecoverde code is aangepast op volgorde van toevoeging

### 2.4 Affected random

al degene waarvan gecoverde code is aangepast in random volgorde

### 2.5 Greedy cover: zo snel mogelijk zo groot mogelijke coverset opbouwen

### 2.6 Greedy time: zo snel mogelijk zoveel mogelijk tests uitvoeren

### 2.7 HGS

- bepaal voor elke lijn door welke test(s) die gecoverd wordt in een dictionary
- groepeer lijnen die door exact dezelfde set tests worden gecoverd om de grote van die dictionary te verkleinen
- zolang er lijnen zijn die niet gecoverd zijn
  - vind de lijn die door zo weinig mogelijk tests wordt gecoverd
  - voer daarvan de test uit die de meeste resterende lijnen covert

### 3 Rocket

- bepaal voor elke test hoelang het in de vorige run duurde om die uit te voeren
- bereken de som D hiervan = duur van uitvoering voor volledige run
- maak een matrix met als kolommen de tests en als rijen de laatste 10 uitvoeringen, en plaats 1 in een cel als de test uit die kolom tijdens de run uit die rij geslaagd is, -1 als die gefaald is
- bepaal de cumulatieve prioriteit per test door elke cel maal een gewicht te doen en kolomsgewijs op te tellen:
  - 0.7 voor bovenste rij
  - 0.2 voor 2e rij
  - 0.1 voor andere rijen
- tel bij die waarde de relatieve tijdsduur op ( = duur in vorige run / D)
- voer tests op volgorde van juist berekende waarde zodanig dat tests die recent gefaald hebben en niet lang duren eerder worden uitgevoerd

#### 3.1 Alpha

eigen algoritme combineert alle ideeën van bovenstaande

- maak een set C van alle lijnen code die gecoverd worden door minstens één test
- bereken gemiddelde duurtijd van een test
- als minstens 1 keer geslaagd -> gemiddelde van enkel de geslaagde keren
- als enkel gefaald -> gemiddelde van alles (dus enkel de gefaalde keren)
- voer alle tests uit die in de vorige run zowel gefaald zijn als waarvan de gecoverde code aangepast is op volgorde van duurtijd (snelste eerst)
  - verwijder lijnen die deze tests coveren uit C
- voer alle tests uit die in de vorige run gefaald zijn, op volgorde van duurtijd (snelste eerst)
  - verwijder lijnen die deze tests coveren uit C
- voer alle tests uit waarvan gecoverde code aangepast is op volgorde van grootste intersectie met C (-> dus die nog meeste bijdragen)
- voer resterende nog niet uitgevoerde tests uit op volgorde van grootste intersectie met C (-> dus die nog meeste bijdragen)
  - als er test bijzit waarvan intersectie leeg is, voer niet uit

### 4 Demo

- eerst met allinorder, dan met alpha ## Dodona plugin Verander best iets in de identification file

## **4.1 Timeago**

bevat een bug

## **4.2 Planning**

- Testen op andere projecten, scripts schrijven om die data te genereren in eerste sem al wat gedaan
- Ruby of Python support toevoegen zodat wel degelijk voor meerdere talen gebruikt kan worden + meer data
- Vanaf begin april normaalgezien enkel nog schrijven

## **4.3 Vragen**

- Mail van Martine: titel + indiendatum ingeven via Plato; hoe bindend is de indiendeadline? Stel niet klaar, of stel juist wel klaar.