Concluzii inițiale asupra proiectului

Pynguin reprezintă un instrument automat de generare a testelor pentru limbajul de programare Python, contribuind la îmbunătățirea calității codului prin crearea automată a cazurilor de test. **Pytest** este un framework de testare extrem de popular în ecosistemul Python, cunoscut pentru sintaxa sa clară și flexibilitatea de a extinde funcționalitățile testării. Împreună, aceste unelte oferă un mediu robust pentru validarea și asigurarea calității aplicațiilor Python.

Ca să înțelegem mai bine cum se descurcă **Pynguin** în practică, am decis să-l testăm pe mai multe soluții la probleme de algoritmică de pe LeetCode. În paralel, folosim și **PyTest** pentru a compara rezultatele. Ideea e să vedem cât de bine reușește Pynguin să genereze teste utile în situații reale, nu doar în exemple artificiale. Prin această abordare, încercăm să ne facem o părere cât mai clară despre cât de eficient și de fiabil este în generarea automată de teste.

Limitări identificate:

1. **Folosirea numerelor complexe ca input**
   Am observat că Pynguin are tendința de a genera cazuri de testare în care sunt folosite **numere complexe**, chiar dacă aceste valori nu au legătură cu ceea ce așteaptă funcția respectivă. În majoritatea cazurilor, acest lucru duce la rezultate irelevante sau erori.

2. **Ignorarea tipurilor specificate în antetul funcției**
   Chiar dacă am folosit adnotări de tip în definirea funcțiilor, generatorul uneori le ignoră și încearcă să apeleze funcțiile cu tipuri de date incompatibile. Asta poate crea confuzie și afectează validitatea testelor generate.

3. **Apelarea greșită a funcțiilor fără parametri**
   Deși o funcție nu are parametri, Pynguin poate totuși să genereze un apel în care îi pasează argumente, ceea ce evident duce la o eroare de execuție.

4. **Transmite valori individuale în loc de liste**
   În unele cazuri, dacă funcția așteaptă o listă, Pynguin trimite o valoare singulară (de exemplu, un int), ceea ce din nou produce erori sau comportamente neintenționate.

Aspecte pozitive:

1. **Suport pentru seed-uri reproducibile**
   Faptul că putem seta un seed pentru generarea testelor ajută mult atunci când vrem să reproducem exact aceleași cazuri de test, mai ales pentru debugging sau testare repetitivă.

2. **Acoperire configurabilă**
   E foarte util că putem decide ce tip de acoperire vrem să urmărim — fie că e vorba de acoperirea ramurilor, a instrucțiunilor sau altceva. Asta face instrumentul mai flexibil, în funcție de nevoile proiectului.

3. **Punct bun de plecare pentru testare automată**
   În ciuda micilor probleme menționate mai sus, Pynguin este un **start solid** atunci când vrei să introduci testarea automată într-un proiect. Ajută să acoperi rapid o parte din funcționalitate și oferă o bază peste care poți construi teste mai rafinate ulterior.

Ce urmează:

Pentru a îmbunătăți și mai mult procesul de testare, următorul pas pe care îl avem în vedere este **generarea de teste folosind modele de tip LLM (Large Language Models)**. Ideea este să combinăm avantajele generării automate cu o înțelegere mai profundă a contextului funcțional, pentru a produce teste mai inteligente și mai relevante.

Bibliografie:

- Documentație
- Paper
- Probleme testate