פרויקט DiffTool - סמסטר חורף 2022-2023

מגישות:

נעה רוזנטל 318175130 noaro@campus.technion.ac.il

אביטל סנדלר 208390468 s.avital@campus.technion.ac.il

תיאור הבעיה

פרויקט ה-DiffTool בא לאפשר למשתמש להשוות בין תכניות שמבצעות את אותה המשימה בדרכים שונות, זאת על גבי מערכת פלאדין (פירוט על המערכת בהמשך). הכלי נועד לעזור למפתחים לדבג תוכנית בהתבסס על תוכנית מקבילה, כך שנתון לנו שהיא פועלת כמצופה. באופן זה, מתאפשר למפתח לקבל תמונת מצב אודות התכנית שלו, להבין היכן השוני בין התוכנית שלו לתוכנית המקבילה, וכן האם שוני זה משפיע על התוצאה או שהתכניות שקולות.

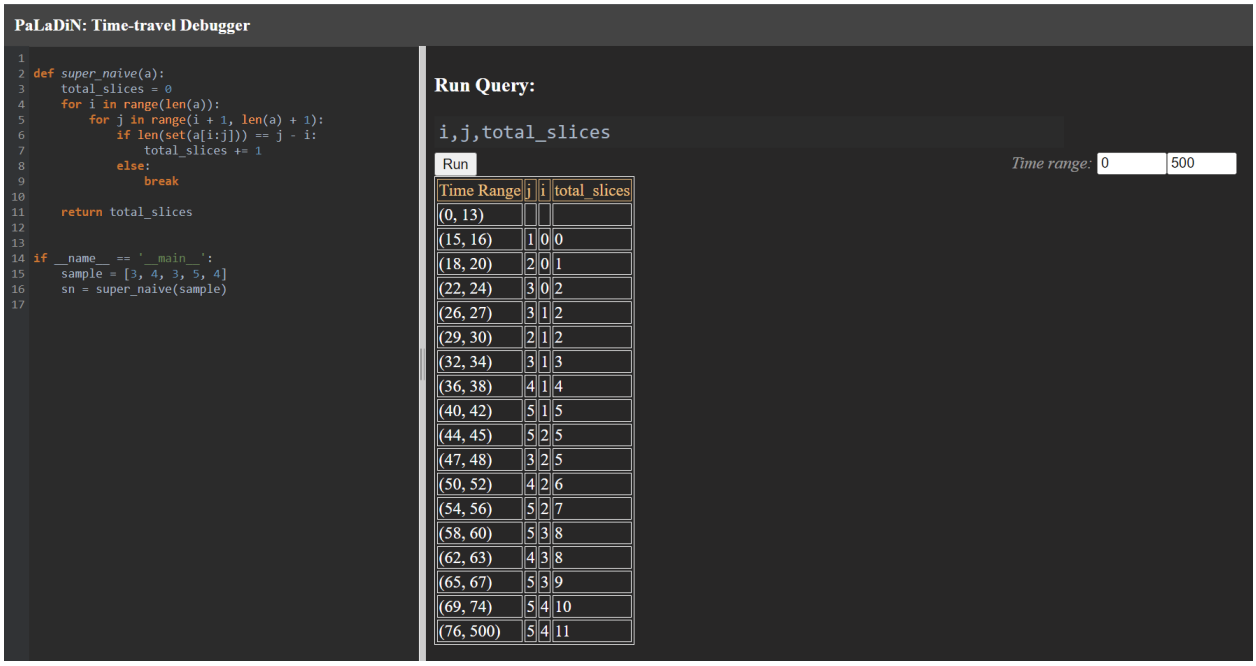
מערכת פלאדין

מערכת פלאדין היא מערכת של שרת ולקוח.
השרת מקבל כקלט קובץ של תוכנית הרצה בפייתון, ומבצע אנליזה לתוכנית. בעקבות האנליזה, נוצר אובייקט מסוג Archive שמכיל בתוכו את כל מהלך התוכנית. האובייקט Archive מורכב מאסופת רשומות, records, כאשר כל רשומה מייצגת פעולה שמתרחשת בקוד.
בסיום ריצת השרת, נוצר קובץ csv שמכיל את אובייקט ה-Archive (קובץ זה מתקבל כקלט של הכלי DiffTool).
צד הלקוח מאפשר למשתמש להכניס שאילתות אודות המשתנים השונים של התוכנית. השאילתות רצות על אובייקט ה-Archive שיצר השרת, ומייצרות טבלה לפי זמנים, כאשר כל רשומה בטבלה מייצגת את מצב המשתנים של השאילתה בנקודת זמן.

לדוגמה, שורת הרצה עבור התוכנית caterpillar_super_naive.py:

```
<path_to_paladin_engine>\paladin_engine\PaladinUI\paladin_cli\paladin_cli.py --run
--csv DiffTool\caterpillar_super_naive.csv --run-debug-server True --port 1234
<path_to_paladin_engine>\paladin_engine\PaladinEngine\tests\test_resources\examples\caterpillar\caterpillar_super_naive.py
```

דוגמה לשאילתה ולפלט שמתקבל:



פרויקט DiffTool

מטרת הכלי היא להשוות בין שתי תכניות שמבצעות את אותה המשימה בדרכים שונות, על ידי מציאת אזורי השקה ביניהן, הבאים לידי ביטוי בשוויון בין הפרמטרים של שתי התוכניות, בנקודות מקבילות בריצות שלהן.

מבנה

הכלי ממומש בשפת פייתון, וכתוב בקובץ paladin_diff_tool.py.
לצורך ביצוע ההשוואה, הכלי עושה שימוש בקבצי ה-csv של שתי התכניות, המיוצרים על ידי פלאדין.
הקובץ מייצא את הפונקציות הבאות, המיועדות לשימוש ב-jupyter notebook:

1) הפונקציה get_data: מקבלת csv של תוכנית ומחזירה dataframe.
2) הפונקציה merge_tables: מקבלת שני dataframes, שהוחזרו מקריאות ל-get_data, וכן פרמטרים להשוואה, ומחזירה טבלה מאוחדת לפי הפרמטרים שהועברו, וכן מחזירה את מספרי השורות המתאימות ומספרי השורות שאינן מתאימות בין הריצות (להמשך שימוש במחברת ה-jupyter).
   הפרמטרים להשוואה מתחלקים לשניים:
   - פרמטרים שנועדו להשוות בין גוף הריצה של כל אחת מהתוכניות - merge_condition,
   - פרמטרים שנועדו להשוות בין רשומות התוצאה של כל אחת מהתוכניות - result_condition.
   נבחין כי יש חשיבות להפרדה בין שני הסוגים, שכן תמיד נרצה להשוות בין רשומות התוצאה על פי הפרמטר המכיל את התוצאה, ללא חשיבות שאר הפרמטרים, לעומת זאת, בגוף הריצה נרצה לחפש נקודות השקה על סמך פרמטרים אחרים.

הכלי נועד לשימוש מתוך jupyter notebook, זאת על מנת לאפשר למשתמש לפעול בצורה דינאמית, ולשחק עם הפרמטרים עד שמתקבלת התוצאה המתאימה, בדגש על query_vars שמקבלת הפונקציה get_data ו-merge_condition, result_condition שמקבלת הפונקציה merge_tables.

מחלקות

- המחלקה TableInfo: מאפשרת את ריכוז המידע הנתון אודות תכנית מסוימת, כך שניתן יהיה להעבירו בצורה נוחה לפונקציות השונות. המידע כולל את הנתיב לקובץ csv שנוצר עבור התכנית על ידי פלאדין, וכן סיומת (suffix), שתתווסף לכותרות הטבלה של התכנית על מנת להבדיל את הפרמטרים שלה מהפרמטרים של התכנית האחרת (כדי למנוע התנגשות במקרה של בחירת שמות זהים).
- המחלקה MergeTableInstance: מחלקה פנימית של הכלי, המאפשרת את העברת המידע הנחוץ עבור תהליך היצירה של הטבלה המאוחדת בין הפונקציות השונות. לשם כך נוצר מופע של המחלקה עבור כל אחת מהתוכניות.

אופן שימוש

הפונקציות שמיוצאות על ידי הכלי נועדו לשימוש מתוך jupyter notebook.
לאחר פתיחת המחברת, יש לבצע:

```
import paladin_diff_tool
```

לאחר מכן, <u>עבור כל אחת משתי התוכניות</u>, יש לבצע את השלבים הבאים:

1) ליצור מופע של המחלקה TableInfo, עם המידע הרלוונטי עבור התוכנית.
   לדוגמה:

```
table_info_1 = paladin_diff_tool.TableInfo(
    csv_file_path="caterpillar_super_duper_naive.csv",
    suffix="_sdn")
```

2) לקרוא לפונקציה get_data, עם המידע הרלוונטי עבור התוכנית.
לדוגמה:

```
table1 = paladin_diff_tool.get_data(
    table_info=table_info_1,
    query_vars=["total_slices","i","j", "sdn"]
)
```

הפונקציה מחזירה את ה-dataframe, אותו ניתן להציג בתוך המחברת.

כעת, ניתן לבצע את איחוד הטבלאות, על ידי קריאה לפונקציה merge_tables.
לדוגמה:

```
merged, match_indices, diff_indices = paladin_diff_tool.merge_tables(
   table_info_1=table_info_1,
   table_info_2=table_info_2,
   table1=table1,
   table2=table2,
   merge_condition_1=["total_slices","i","j"],
   merge_condition_2=["total_slices","i","j"],
   result_condition_1=["sdn"],
   result_condition_2=["sn"]
)
```

הפונקציה מחזירה שלושה ערכים: הערך הראשון הוא הטבלה המאוחדת, אחריו רשימה של מספרי השורות המתאימות, ולבסוף רשימה של מספרי השורות שאינן מתאימות.
בשלב זה ניתן להדפיס את הטבלה המאוחדת כמו שהיא, או לחלופין ניתן לעשות שימוש במידע על מספרי השורות על מנת לצבוע את הטבלה באדום ובירוק בהתאם, ולהפוך אותה לקריאה יותר:

```
import pandas as pd
merged.style.set_properties(
    subset=pd.IndexSlice[merged.loc[match_indices].index,],
    **{'background-color': paladin_diff_tool.GREEN_COLOR}
).set_properties(
    subset=pd.IndexSlice[merged.loc[diff_indices].index,],
    **{'background-color': paladin_diff_tool.RED_COLOR}
).format(na_rep='', precision=2)
```

דוגמאות

דוגמה פשוטה: is_prime

התוכנית בודקת האם מספר נתון number הוא ראשוני.
בפתרון ה-naive אנו מבצעים לולאה שרצה מ-2 עד number, לעומתו בפתרון ה-square אנו מבצעים לולאה שרצה מ-2 עד לשורש של number.
בדוגמה זו, אם כן, שתי התוכניות זהות עד כדי כך שאחת מהן ממשיכה להריץ איטרציות נוספות לאחר שהאחרת מסיימת.
דוגמת הרצה עבור number=13:

	Time Range_n	i@4_n	number@11_n	result@12_n	Time Range_s	i@8_s	number@15_s	result@16_s
0	(0, 1)				(0, 1)			
1	(3, 6)		13.00		(3, 8)		13.00	
2	(8, 8)	3.00	13.00		(10, 11)	3.00	13.00	
3	(10, 10)	5.00	13.00					
4	(12, 12)	7.00	13.00					
5	(14, 14)	9.00	13.00					
6	(16, 16)	11.00	13.00					
7	(18, 18)	12.00	13.00					
8	(20, 500)	12.00	13.00	True	(13, 500)	3.00	13.00	True

דוגמה נוספת: caterpillar

התוכנית מקבלת כקלט מערך של מספרים, ומחזירה את כמות תתי המערכים המכילים איברים שונים ללא חזרה.
בפתרון ה-super duper naive אנו מבצעים שתי לולאות ללא תנאי עצירה פנימיים, לעומתו בפתרון
ה-super_naive אנו מבצעים את אותן שתי הלולאות, אך כאשר מגיעים לתת מערך המכיל כפילות, מפסיקים
לרוץ על המשך המערך ומבצעים break על מנת להמשיך לאיטרציה הבאה של הלולאה החיצונית.
בדוגמה זו, אם כן, שתי התוכניות זהות עד כדי כך שאחת מהן מדלגת על חלק מהאיטרציות.
דוגמת הרצה עבור המערך

sample=[3,4,3,5,4]:

	Time Range_sdn	total_slices_sdn	i_sdn	j_sdn	sdn_sdn	Time Range_sn	total_slices_sn	i_sn	j_sn	sn_sn
0	(0, 8)					(0, 8)				
1	(10, 10)	0.00				(10, 10)	0.00			
2	(12, 13)	0.00	0.00			(12, 13)	0.00	0.00		
3	(15, 16)	0.00	0.00	1.00		(15, 16)	0.00	0.00	1.00	
4	(18, 20)	1.00	0.00	2.00		(18, 20)	1.00	0.00	2.00	
5	(22, 23)	2.00	0.00	3.00		(22, 24)	2.00	0.00	3.00	
6	(25, 25)	2.00	0.00	4.00		(26, 27)	2.00	1.00	3.00	
7	(27, 27)	2.00	0.00	5.00						
8	(29, 30)	2.00	1.00	5.00						
9	(32, 33)	2.00	1.00	2.00		(29, 30)	2.00	1.00	2.00	
10	(35, 37)	3.00	1.00	3.00		(32, 34)	3.00	1.00	3.00	
11	(39, 41)	4.00	1.00	4.00		(36, 38)	4.00	1.00	4.00	
12	(43, 44)	5.00	1.00	5.00		(40, 42)	5.00	1.00	5.00	
13	(46, 47)	5.00	2.00	5.00		(44, 45)	5.00	2.00	5.00	
14	(49, 50)	5.00	2.00	3.00		(47, 48)	5.00	2.00	3.00	
15	(52, 54)	6.00	2.00	4.00		(50, 52)	6.00	2.00	4.00	
16	(56, 58)	7.00	2.00	5.00		(54, 56)	7.00	2.00	5.00	
17	(60, 62)	8.00	3.00	5.00		(58, 60)	8.00	3.00	5.00	
18	(64, 65)	8.00	3.00	4.00		(62, 63)	8.00	3.00	4.00	
19	(67, 69)	9.00	3.00	5.00		(65, 67)	9.00	3.00	5.00	
20	(71, 76)	10.00	4.00	5.00		(69, 74)	10.00	4.00	5.00	
21	(78, 78)	11.00	4.00	5.00		(76, 76)	11.00	4.00	5.00	
22	(80, 500)	11.00	4.00	5.00	11.00	(78, 500)	11.00	4.00	5.00	11.00