Αναφορά Διόρθωσης Μετρικών Halstead

Σκοπός

Σκοπός της παρούσας αναφοράς είναι η τεκμηριωμένη διόρθωση των απαντήσεων που παρήγαγε το ChatGPT σχετικά με τον υπολογισμό των μετρικών Halstead για δύο υλοποιήσεις της ρουτίνας του Kaprekar (σε Python και ANSI C). Η ανάλυση βασίζεται σε σαφώς ορισμένες παραδοχές μέτρησης και συγκρίνει τα **λανθασμένα αποτελέσματα** με τα **ορθά**, αιτιολογώντας αναλυτικά τις αποκλίσεις.

Αναφορά στο excel των υπολογισμών: Project_3.xlsx

Ερώτημα 1:

Μπορείτε να βρείτε πού έχει κάνει λάθος το ChatGPT και να δώσετε τους σωστούς υπολογισμούς και αποτελέσματα για τις τιμές αυτές; Καλύτερα δώστε ένα αναλυτικό πίνακα (όπως στο παράδειγμα στις διαφάνειες) που να περιλαμβάνει τα πάντα.

Παραδοχές Μέτρησης

- Όλα τα strings (συμπεριλαμβανομένων f-strings) μετρώνται ως **ένα εντέλο**, ακόμη και αν περιέχουν μεταβλητές. Μια άλλη παραδοχή θα μπορούσε να είναι η μέτρηση ενός έξτρα εντέλου στη θέση της χρήσης της μεταβλητής, κάτι το οποίο δεν επιλέγεται στη συγκεκριμένη περίπτωση.
- το .join() και το .isdigit() θεωρούνται ξεχωριστοί τελεστές. Μια παραδοχή θα μπορούσε να είναι να μετρήσουμε ξεχωριστά το "." ως operator και το join/isdigit ως έντελο μιας και το "." ενεργοποιεί τις build in συναρτήσεις
- Οι εντολές if, while, def, καθώς και τα if–else, μετρώνται **μαζί με το ":"**, καθώς αυτό αποτελεί αναπόσπαστο στοιχείο της ελάχιστης συντακτικής τους μορφής στην Python. το ":" είναι υποχρεωτικό για να οριστεί η εντολή if και η απουσία αυτού οδηγεί σε σφάλμα. Το ίδιο και για το while loop. Υπάρχει μόνο μια εξαίρεση χρήσης του if else χωρίς ":" και αυτή είναι στη περίπτωση των ternary operators τύπου "x = true if 1==1 else false". Για τους σκοπούς της άσκησης υποθέτουμε ότι η ελάχιστη υλοποίηση της if else για τη χρήση που γίνεται στο παράδειγμα απαιτεί και την ":" οπότε αυτή δεν μετριέται ώς ξεχωριστός τελεστής.
- Το += θεωρείται **αυτόνομος τελεστής** και δεν αναλύεται σε + και =
- Το == είναι διαφορετικός τελεστής από το =
- Το print() μετράται μαζί με τις παρενθέσεις του (συνάρτηση Python 3)
- Το sorted() θεωρείται τελεστής, όταν χρησιμοποιείται με περισσότερα από ένα ορίσματα, το "," μετράται ως **επιπλέον τελεστής**
- Τα reverse και True μετρώνται ως έντελα, καθώς επηρεάζουν τη συμπεριφορά της sorted()
- Τα σχόλια εξαιρούνται από τις μετρήσεις

Κύρια Λάθη του ChatGPT (Python)

- Διαχώρισε λανθασμένα το += σε + και =
- Μέτρησε το ":" ως ανεξάρτητο τελεστή αντί να το συνυπολογίσει με if, while, def
- παρέλειψε το τελεστή kaprekar_routine()
- Παρέλειψε το , ως τελεστή μεταξύ εντέλων.
- το if else είναι διαφορετικός τελεστής από το if μονο του
- Παρέλειψε τελεστές όπως def:, return, input(), sorted(), .join(), .isdigit().
- το print πάει μαζί με τις παρενθέσεις όχι μόνο του
- το == είναι διαφορετικός τελεστής από το σκέτο =
- Δεν υπολόγισε πολλά έντελα: strings, f-strings, True, 1, -1

Ορθά Αποτελέσματα Μετρικών (Python)

operators	**#**	**operands**	**#**
def :	1	number	4
=	5	count	1
while :	2	0	2
!=	1	6174	1
int ( )	2	num_str	3
.join()	2	f"{number:04d}"	1
sorted()	2	largest	2
,	1	smallest	2
kaprekar_routine()	2	""	2
-	1	reverse	1
+=	2	True	2
print( )	5	f"Step {count}: {largest} - {smallest} = {number}"	1
if :	2	"This number reaches 0 and won't reach 6174."	1
==	2	-1	1
return	1	f"Kaprekar's constant reached in {count} steps!"	1
if : else:	2	"Enter a 4-digit number (at least two different digits): "	1
input( )	1	user_input	5
len( )	2	4	1

and	1	1	1
.isdigit()	1	"All digits are the same. Try again."	1
set( )	1	"Invalid input. Enter exactly 4 digits."	1
break	1		

Φυσικά υπάρχουν και οι αντίστοιχες διαφορές στις μετρικές. Τα αποτελέσματα είναι τα εξής:

μετρική	**chat**	**σωστή**
n_1	15	22
n_2	9	21
N_1	38	40
N_2	25	35
n	24	43
N	63	75
N_{est}	-	~190.35
V	~288.9	~406.97
L_{est} (λ)	~0.048	~0.05
D	~20.83	~18.34
E	~6018	~7461.11
T	-	~414.51
B	-	~0.12

Ερώτημα 2:

Μπορείτε να βρείτε πού έχει κάνει λάθος το ChatGPT και να δώσετε τους σωστούς υπολογισμούς για τις τιμές αυτές στη 2η έκδοση του προγράμματος; Καλύτερα δώστε δύο αναλυτικούς πίνακες (όπως στο παράδειγμα στις διαφάνειες) ένα για τη συνάρτηση isPalidrome() και ένα για το main() και μετά υπολογίστε τα αποτελέσματα συνολικά. Η λύση σας πρέπει να περιλαμβάνει τα πάντα εκτός από τις γραμμές #include

Βασικές Παραδοχές

- Το digits[i] μετράται ως τελεστής digits[] και έντελο i (αντίστοιχα 0,1,2,3,4,i και j) , αντιμετωπίζοντας δηλαδή το array αυτό ως μια συνάρτηση. Γι' αυτό και δεν υπολογίζονται οι έξτρα αγκύλες, αλλά πηγαίνουν μαζί με τη συνάρτηση. Η αρχικοποίηση της digits[] θεωρείται μια ακόμα εμφάνισή της ως τελεστής.
- Τα {} δεν θεωρούνται μέρος της ελάχιστης υλοποίησης συναρτήσεων και μετρώνται ξεχωριστά.
- Η kaprekar_routine αποτελεί μια ρουτίνα, η οποία εξ' ορισμού (μιας και αναφερόμαστε στη γλώσσα C) απαιτεί τα άγκιστρα "{}" καθώς και τις παρενθέσεις "()". Γι΄αυτό το λόγο δεν μετριούνται ως ξεχωριστούς τελεστές.
- το kaprekar_routine() στην αρχικοποίηση και στην χρήση του είναι δύο διαφορετικοί τελεστές
- Οι τελεστές if(), while(), for() και printf() μετρώνται **μαζί με τις παρενθέσεις τους**.
- Η for(;;) περιλαμβάνει υποχρεωτικά και τα δύο ";"
- Το scanf() περιλαμβάνει το υποχρεωτικό κόμμα των ορισμάτων
- Το ++ θεωρείται διαφορετικός τελεστής από το +
- Όλα τα strings των printf θεωρούνται **έντελα**
- **Από την εκφώνηση**, εξαιρούνται από τη μέτρηση τα #include
- Τα σχόλια εξαιρούνται από τις μετρήσεις

Κύρια Λάθη του ChatGPT (ANSI C)

- εχει παραλείψει από τους τελεστές το void, kaprekar_routine () {}, int, return, main () {}, &, ||, kaprekar_routine()
- Αντιμετώπισε if, printf, while χωρίς τις παρενθέσεις τους
- Δεν αναγνώρισε το if else ως ξεχωριστό τελεστή
- Το else δεν είναι μόνος του τελεστής
- Ανέλυσε λανθασμένα το scanf() και το for()
- το ++ είναι διαφορετικό από το μονό + και δηλώνει +1
- Παρέλειψε έντελα όπως 1111, 9999, 2 και όλα τα strings

Ορθά Αποτελέσματα Μετρικών (ANSI C)

kaprekar_routine( )			
operators	**#**	**operands**	**#**
void	**1**	number	**10**
kaprekar_routine( ) { }	**1**	count	**4**
int	**7**	0	**6**
=	**13**	1111	**1**
;	**19**	"The number %04d is not valid for Kaprekar's routine.\n"	**1**
if ( )	**2**	6174	**1**

{}	**5**	i	**7**
%	**4**	j	**7**
==	**1**	1	**4**
printf( )	**3**	2	**3**
,	**6**	3	4
return	**1**	4	**2**
while ( )	**1**	1000	**3**
!=	**1**	100	**3**
/	**3**	10	**6**
( )	**2**	temp	**2**
for (; ;)	**2**	smallest	**3**
<	**2**	largest	**3**
++	**3**	"Step %d: %d - %d = %d\n"	**1**
+	**7**	"Reached Kaprekar's constant 6174 in %d steps.\n"	**1**
>	**1**		
*	**6**		
-	**1**		
digits[ ]	**19**		
main( )			
int	**2**	number	**5**
main( ) {}	**1**	"Enter a four-digit number (at least two different digits): "	**1**
;	**6**	"%d"	**1**
printf( )	**2**	1000	**1**
scanf(,)	**1**	9999	**1**
&	**1**	"Please enter a valid four-digit number.\n"	**1**
if( ) else	**1**	0	**1**

{}	**2**		
<	**1**		
>	**1**		
||	**1**		
kaprekar_routine ( )	**1**		
return	**1**		

Φυσικά υπάρχουν και οι αντίστοιχες διαφορές στις μετρικές. Τα αποτελέσματα είναι τα εξής:

μετρική	**chat**	**σωστή**
n_1	19	30
n_2	16	24
N_1	55	132
N_2	42	83
n	35	54
N	97	215
N_{est}	-	~257.25
V	~497.6	~1237.30
L_{est} (λ)	~0.040	~0.02
D	~24.94	~51.87
E	~12398	~64184.98
T	-	~3565.83
B	-	~0.51

Ερώτημα 3:

Μπορείτε να βρείτε πού έχει κάνει λάθος το ChatGPT σε αυτές τις συγκρίσεις; Δώστε τη δική σας αιτιολόγηση.

Με βάση τις δικές μου μετρήσεις προκύπτουν τα εξής:

metrics	Python	ANSI C	Larger	Description
n_1	22	30	ANSI C	Distinct operators
N_1	40	132	ANSI C	Total operators
n_2	21	24	ANSI C	Distinct operands
N_2	35	83	ANSI C	Total operands
n	43	54	ANSI C	Program Vocabulary size
N	75	215	ANSI C	Program Length
N_{est}	190.346161	257.245818	ANSI C	Program Length Estimator
V	406.969857	1237.30081	ANSI C	Program Volume
L_{est}	0.05454545	0.01927711	Python	Program Level
D	18.3333333	51.875	ANSI C	Program Difficulty / error proneness
E	7461.11404	64184.9797	ANSI C	Program Effort
T	414.506335	3565.8322	ANSI C	Programming Time
B	0.1231707	0.51712307	ANSI C	Program Bugs where $E_0 = 3100$

Ορθά Συμπεράσματα

- Η υλοποίηση σε **ANSI C** εμφανίζει σαφώς **μεγαλύτερο λεξιλόγιο (n, N)** λόγω της πιο αναλυτικής σύνταξης και της ρητής διαχείρισης τύπων και ελέγχων
- Ο όγκος (V), η δυσκολία (D) και η προσπάθεια (E) είναι σημαντικά αυξημένα στην C, γεγονός που υποδηλώνει:
    - μεγαλύτερη πολυπλοκότητα
    - αυξημένη πιθανότητα σφαλμάτων
    - μεγαλύτερο χρόνο ανάπτυξης
- Η Python παρουσιάζει υψηλότερο **Program Level (L)**, επιβεβαιώνοντας ότι είναι γλώσσα υψηλότερου επιπέδου.

Λάθος Συμπέρασμα του ChatGPT

Το ChatGPT υποτίμησε συστηματικά την πολυπλοκότητα της ANSI C υλοποίησης, λόγω ελλιπούς αναγνώρισης τελεστών και εντέλων, λανθασμένων συντακτικών παραδοχών, ασυνέπειας στον ορισμό της ελάχιστης υλοποίησης γλωσσικών δομών. Παρόλα αυτά τα συγκριτικά αποτελέσματα που παρουσίασε είναι σωστά, υποδηλώνοντας μεγαλύτερο λεξιλόγιο, όγκο και δυσκολία της C ενώ υψηλότερο επίπεδο της γλώσσας Python.

Τελικό Συμπέρασμα

Η παρούσα ανάλυση καταδεικνύει ότι οι μετρικές Halstead είναι **εξαιρετικά ευαίσθητες στις παραδοχές μέτρησης**. Η σωστή, συνεπής και γλωσσικά τεκμηριωμένη ερμηνεία των τελεστών και εντέλων είναι κρίσιμη για την εξαγωγή έγκυρων συμπερασμάτων. Οι διορθωμένες μετρήσεις επιβεβαιώνουν τη σημαντικά αυξημένη πολυπλοκότητα της ANSI C σε σχέση με την Python για το ίδιο πρόβλημα.