

Hashing Algorithm Comparison: Custom Hash vs SHA-256

Name: VIJAY S P (24MSRDF037)

Objective

Design a custom hashing function using character manipulation, modular arithmetic, and compression, and compare it with the standard `hashlib.sha256()` for various inputs.

Step-by-Step Breakdown of Custom Hash Algorithm

Step 1: ASCII Conversion with Index Multiplication

Each character in the input string is converted to its ASCII value and multiplied by its 1-based index:

```
ascii_val = ord(char) * (i + 1)
```

Step 2: Modular Mixing

Each result is reduced with modulo 97 (a prime) for randomness and better spread:

```
mixed_val = ascii_val % 97
```

Step 3: Block-wise Compression

The resulting list of values is processed into 32 "buckets" (for 32-character output). Each bucket sums values from its offset and is reduced using mod 256 (for byte range):

```
final_val = sum(offset_vals) % 256
```

Step 4: Hex Encoding

All values are converted to 2-character hexadecimal values and concatenated to get a 32-character hash.

Python Code

```
import hashlib
```

```
def custom_hash(input_str):
```

```
    HASH_LENGTH = 32
```

```
    MODULO = 97
```

```
    ascii_vals = [(ord(char) * (i + 1)) % MODULO for i, char in enumerate(input_str)]
```

```
    hash_nums = []
```

```
    for i in range(HASH_LENGTH):
```

```

total = 0
for j in range(i, len(ascii_vals), HASH_LENGTH):
    total += ascii_vals[j]
hash_nums.append(total % 256)
final_hash = ''.join(f'{val:02x}' for val in hash_nums)[:HASH_LENGTH]
return final_hash

```

```

def sha256_hash(input_str):
    return hashlib.sha256(input_str.encode()).hexdigest()

```

```

1 import hashlib
2
3 def custom_hash(input_str):
4     HASH_LENGTH = 32
5     MODULO = 97
6     ascii_vals = [(ord(char) * (i + 1)) % MODULO for i, char in enumerate(input_str)]
7     hash_nums = []
8     for i in range(HASH_LENGTH):
9         total = 0
10        for j in range(i, len(ascii_vals), HASH_LENGTH):
11            total += ascii_vals[j]
12        hash_nums.append(total % 256)
13    final_hash = ''.join(f'{val:02x}' for val in hash_nums)[:HASH_LENGTH]
14    return final_hash
15
16 def sha256_hash(input_str):
17     return hashlib.sha256(input_str.encode()).hexdigest()
18
19 test_cases = [
20     "",
21     "hello",
22     "Hello",
23     "sample",
24     "Sample123",
25     "This is a longer input string for testing",
26 ]
27
28 print(f"{'Input':<40}{'Custom Hash':<40}{'SHA-256 (first 32 chars)'}")
29 print("-" * 110)
30 for text in test_cases:
31     print(f"{'text':<40}{'custom_hash(text)':<40}{'sha256_hash(text)':<32}")
32

```

Custom Hash vs SHA-256 — Output Table

Input	Custom Hash (32 chars)	SHA-256 (first 32 chars)
	00000000000000000000000000000000	e3b0c44298fc1c149afb4c8996fb924
hello	e2131f1011140f170712050c0b060601	2cf24dba5fb0a30e26e83b2ac5b9e29e
Hello	8d131b242d261f270e1a191e1d080c0f	185f8db32271fe25f561a6fc938b2e26
sample	fa2a180e0712081810100e130f0a0502	1c6b1c3adf8e7c1c52ae45b241a47a9c
Sample123	f70b2e2120231c241d1c1a1716050403	55d90a6cd3178b800130b82f5a07d1f3

