daa_hashing

April 30, 2023

1 Análise

1.1 Introdução

Tabelas hash são estruturas de dados eficientes, amplamente utilizadas por proporcionarem acesso rápido, inserção e exclusão de elementos. Este relatório investiga a estrutura de dados da tabela hash, focando nas estratégias de resolução de colisões - endereçamento aberto e encadeamento separado - e no papel crucial das funções hash no desempenho geral das tabelas. Abordaremos também exemplos de algoritmos de hashing eficientes e subótimos.

O endereçamento aberto resolve colisões com buscas sequenciais na tabela, enquanto o encadeamento separado utiliza listas encadeadas em cada índice para armazenar elementos colidentes. Funções hash eficazes, como MurmurHash e CityHash, distribuem uniformemente as chaves no array, garantindo operações eficientes. Funções hash ruins, como simples operações de módulo, resultam em distribuição desigual e menor eficiência.

1.2 Bibliotecas desenvolvidas

Para a presente analise, foi desenvolvido pelo grupo três bibliotecas escritas em **C+** com **bindings** para a linguagem python:

- **picods**: Biblioteca para visualização de dados em gráficos e tabelas.
- **pydaa**: Biblioteca para Projeto e analise de algoritmos, que contém as implementações do problema da mochila binária.
- **pyaon**: Biblioteca auxiliar para a leitura de arquivos de entrada do problema.

Sendo assim, para instalar essas libs:

[ ]:
```
!pip install picods pydaa pyaon
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: picods in /usr/lib/python3.10/site-packages
(0.1.2)
Requirement already satisfied: pydaa in /home/pablo/.local/lib/python3.10/site-
packages (1.6.5)
Requirement already satisfied: pyaon in /usr/lib/python3.10/site-packages
(1.1.2)
Requirement already satisfied: pandas<2.0.0,>=1.5.3 in /usr/lib/python3.10/site-
packages (from picods) (1.5.3)
Requirement already satisfied: matplotlib<4.0.0,>=3.7.0 in
/usr/lib/python3.10/site-packages (from picods) (3.7.0)
```

```
Requirement already satisfied: packaging>=20.0 in /usr/lib/python3.10/site-
packages (from matplotlib<4.0.0,>=3.7.0->picods) (23.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/lib/python3.10/site-
packages (from matplotlib<4.0.0,>=3.7.0->picods) (1.0.7)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/lib/python3.10/site-
packages (from matplotlib<4.0.0,>=3.7.0->picods) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/lib/python3.10/site-packages
(from matplotlib<4.0.0,>=3.7.0->picods) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3.10/site-
packages (from matplotlib<4.0.0,>=3.7.0->picods) (9.4.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/lib/python3.10/site-
packages (from matplotlib<4.0.0,>=3.7.0->picods) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in /usr/lib/python3.10/site-
packages (from matplotlib<4.0.0,>=3.7.0->picods) (4.34.4)
Requirement already satisfied: numpy>=1.20 in
/home/pablo/.local/lib/python3.10/site-packages (from
matplotlib<4.0.0,>=3.7.0->picods) (1.24.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/lib/python3.10/site-
packages (from matplotlib<4.0.0,>=3.7.0->picods) (1.4.4)
Requirement already satisfied: pytz>=2020.1 in /usr/lib/python3.10/site-packages
(from pandas<2.0.0,>=1.5.3->picods) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.10/site-packages
(from python-dateutil>=2.7->matplotlib<4.0.0,>=3.7.0->picods) (1.16.0)
```

E importar as funções utilizadas:

[ ]:
```python
import os
from enum import Enum
from typing import Any, Callable, Dict, List

# File parsing
from pyaon.hashing import (load)

# Algorithms
from pydaa.hashing import (open_addressing_hash_table,
                           separate_chaining_hash_table)

# Hash functions
from hash_functions import (identity_hash, modulo_hash, multiplication_hash,
                            left_shift_hash, right_shift_hash, add_hash,
                            xor_hash, minus_hash, multiplicative_method_hash,
                            knuth_multiplicative_method_hash,
                            murmur_hash3_x86_32_hash, farm_hash_hash,
                            city_hash_hash)

# Visualizations
from picods import (picoplot, picotable)
```

```
import matplotlib.pyplot as plt

import json
```

1.3 Arquivos de entrada

O diretório `./data` contém todos os arquivos utilizados na analise. Esse diretório contém outros dois subdiretórios, `./data/build` e `./data/search`, que respectivamente são utilizados com entradas para construção(inserção de valores) e busca nas tabelas hash.

Cada arquivo tem o nome no formato *N.txt*, onde *N* denota a escala do problema, ou seja, a quantidade de itens. Por exemplo, um arquivo `50.txt` possui como conteúdo:

```
3   42  5   48  42 ...  13  3   20  12  37
```

Traduzindo esse arquivo como entrada de uma instância do problema, temos a inserção ou busca de 50 itens na tabela hash.

1.4 Tratamento dos dados

Dessa forma, lemos os dados necessários para a construção e busca nas tabelas Hash:

[ ]:
```
data_build = list(
            sorted(
                map(
                    lambda x: {
                        "size": int("".join(filter(str.isdigit, x))),
                        "data": load(f"./data/build/{x}"),
                    },
                    os.listdir("data/build"),
                ),
                key=lambda x: x["size"],
            )
        )
data_search = list(
            sorted(
                map(
                    lambda x: {
                        "size": int("".join(filter(str.isdigit, x))),
                        "data": load(f"./data/search/{x}"),
                    },
                    os.listdir("data/search"),
                ),
                key=lambda x: x["size"],
            )
        )
```

1.5 Funções Hash

Sendo assim, para o desenvolvimento desse trabalho, foram utilizadas as funções hash definidas a seguir.

[ ]:
```python
## Bad hashing functions
from collections.abc import Callable
from math import sqrt, floor
import cityhash
import farmhash


def identity_hash(_table_size: int) -> Callable[[int], int]:
    """
    Hash an integer by returning it.
    """
    return lambda key: key


def modulo_hash(_table_size: int, modulo: int = 10) -> Callable[[int], int]:
    """
    Return the modulo function of a number.
    """
    return lambda number: number % modulo


def multiplication_hash(table_size: int,
                        constant: int = 42) -> Callable[[int], int]:
    """
    Return a hash function that uses the multiplication method with a constant.
    """
    return lambda key: (key * constant) % table_size


def left_shift_hash(table_size: int, shift: int = 3) -> Callable[[int], int]:
    """
    Return a hash function that uses the left shift method with a shift.
    """
    return lambda key: (key << shift) % table_size


def right_shift_hash(table_size: int, shift: int = 3) -> Callable[[int], int]:
    """
    Return a hash function that uses the right shift method with a shift.
    """
    return lambda key: (key >> shift) % table_size
```

```
def add_hash(table_size: int, add: int = 42) -> Callable[[int], int]:
    """
    Return a hash function that uses the right shift method with a shift.
    """
    return lambda key: (key + add) % table_size


def xor_hash(table_size: int, xor: int = 42) -> Callable[[int], int]:
    """
    Return a hash function that uses the right shift method with a shift.
    """
    return lambda key: (key ^ xor) % table_size


def minus_hash(table_size: int, minus: int = 42) -> Callable[[int], int]:
    """
    Return a hash function that uses the right shift method with a shift.
    """
    return lambda key: (key - minus) % table_size


## Good hashing functions

def multiplicative_method_hash(table_size: int,
                               A: float = 0.42) -> Callable[[int], int]:
    return lambda key: floor(table_size * ((key * A) % 1))


def knuth_multiplicative_method_hash(table_size: int) -> Callable[[int], int]:
    """
    Return a hash function that uses the Knuth multiplicative method.
    """
    return lambda key: floor((key * (sqrt(5) - 1) / 2) % 1 * table_size)


def murmur_hash3_x86_32_hash(table_size: int, seed=0) -> Callable[[int], int]:
    """
    Return a hash function that uses the MurmurHash3 algorithm.
    https://en.wikipedia.org/wiki/MurmurHash
    """

    def fmix(h):
        h ^= h >> 16
        h = (h * 0x85ebca6b) & 0xFFFFFFFF
        h ^= h >> 13
        h = (h * 0xc2b2ae35) & 0xFFFFFFFF
        h ^= h >> 16
```

```
        return h

    def murmur_hash3_x86_32(key: int) -> int:
        key_bytes = key.to_bytes((key.bit_length() + 7) // 8, 'big')
        data = bytearray(key_bytes)
        nblocks = len(data) // 4

        h1 = seed

        c1 = 0xcc9e2d51
        c2 = 0x1b873593

        for i in range(nblocks):
            k1 = (data[4 * i + 3] << 24) | (data[4 * i + 2] << 16) | (
                data[4 * i + 1] << 8) | data[4 * i]

            k1 = (k1 * c1) & 0xFFFFFFFF
            k1 = ((k1 << 15) | (k1 >> (32 - 15))) & 0xFFFFFFFF
            k1 = (k1 * c2) & 0xFFFFFFFF

            h1 ^= k1
            h1 = ((h1 << 13) | (h1 >> (32 - 13))) & 0xFFFFFFFF
            h1 = (h1 * 5 + 0xe6546b64) & 0xFFFFFFFF

        tail = data[nblocks * 4:]
        k1 = 0
        if len(tail) >= 3:
            k1 ^= tail[2] << 16
        if len(tail) >= 2:
            k1 ^= tail[1] << 8
        if len(tail) >= 1:
            k1 ^= tail[0]
            k1 = (k1 * c1) & 0xFFFFFFFF
            k1 = ((k1 << 15) | (k1 >> (32 - 15))) & 0xFFFFFFFF
            k1 = (k1 * c2) & 0xFFFFFFFF
            h1 ^= k1

        h1 ^= len(data)

        h1 = fmix(h1)

        return h1 % table_size

    return murmur_hash3_x86_32


def farm_hash_hash(table_size: int, seed=0) -> Callable[[int], int]:
```

```
    """
    Return a hash function that uses the FarmHash algorithm.
    https://github.com/google/farmhash
    """

    def farm_hash(key: int) -> int:
        key_bytes = key.to_bytes((key.bit_length() + 7) // 8, 'big')
        hash_value = farmhash.FarmHash32WithSeed(key_bytes, seed)
        return hash_value % table_size

    return farm_hash


def city_hash_hash(table_size: int, seed=0) -> Callable[[int], int]:
    """
    Return a hash function that uses the CityHash algorithm.
    https://github.com/google/cityhash
    """

    def city_hash(key: int) -> int:
        key_bytes = key.to_bytes((key.bit_length() + 7) // 8, 'big')
        hash_value = cityhash.CityHash64WithSeed(key_bytes, seed)
        return hash_value % table_size

    return city_hash
```

1.6 Construção dos resultados

Seguindo, definimos os parâmetros, formatos e funções para a obtenção dos resultados.

[ ]:
```
# Hash functions factory
# For each hash function, we want to generate a hash function for each table␣
 ↪size
# e.g. hash_functions["multiplicative_method"](table_size)(key) will return the␣
 ↪hash value for the key
hash_functions = {
    "identity": identity_hash,
    "modulo": modulo_hash,
    "multiplication": multiplication_hash,
    "left_shift": left_shift_hash,
    "multiplicative_method": multiplicative_method_hash,
    "knuth_multiplicative_method": knuth_multiplicative_method_hash,
    "murmur_hash3_x86_32": murmur_hash3_x86_32_hash,
    "farm_hash": farm_hash_hash,
    "city_hash": city_hash_hash,
}
```

```
class Operation(Enum):
    INSERT = "insert"
    SEARCH = "search"


def process_data(
        data: List[Dict[str, Any]], hash_tables: Dict[str, Callable[..., Any]],
        hash_functions: Dict[str, Callable[[int], Callable[[int], int]]],
        operation: Operation) -> Dict[str, Dict[str, List[Dict[str, Any]]]]:
    results: Dict[str, Dict[str, List[Dict[str, Any]]]] = {}

    for table_type, table_constructor in hash_tables.items():
        results[table_type] = {}

        for hash_func_name, hash_func_constructor in hash_functions.items():
            results[table_type][hash_func_name] = []

            for data_entry in data:
                table_size = data_entry["size"]
                hash_func = hash_func_constructor(table_size)
                table = table_constructor(table_size, hash_func)

                total_time = 0
                comparisons = 0

                for key in data_entry["data"]:
                    if operation == Operation.INSERT:
                        _, (time, comp) = table.insert(key)
                    else:
                        _, (time, comp) = table.search(key)

                    comparisons += comp
                    total_time += time

                results[table_type][hash_func_name].append({
                    "size":
                    table_size,
                    "time":
                    total_time,
                    "comparisons":
                    comparisons
                })

    return results


hash_tables = {
```

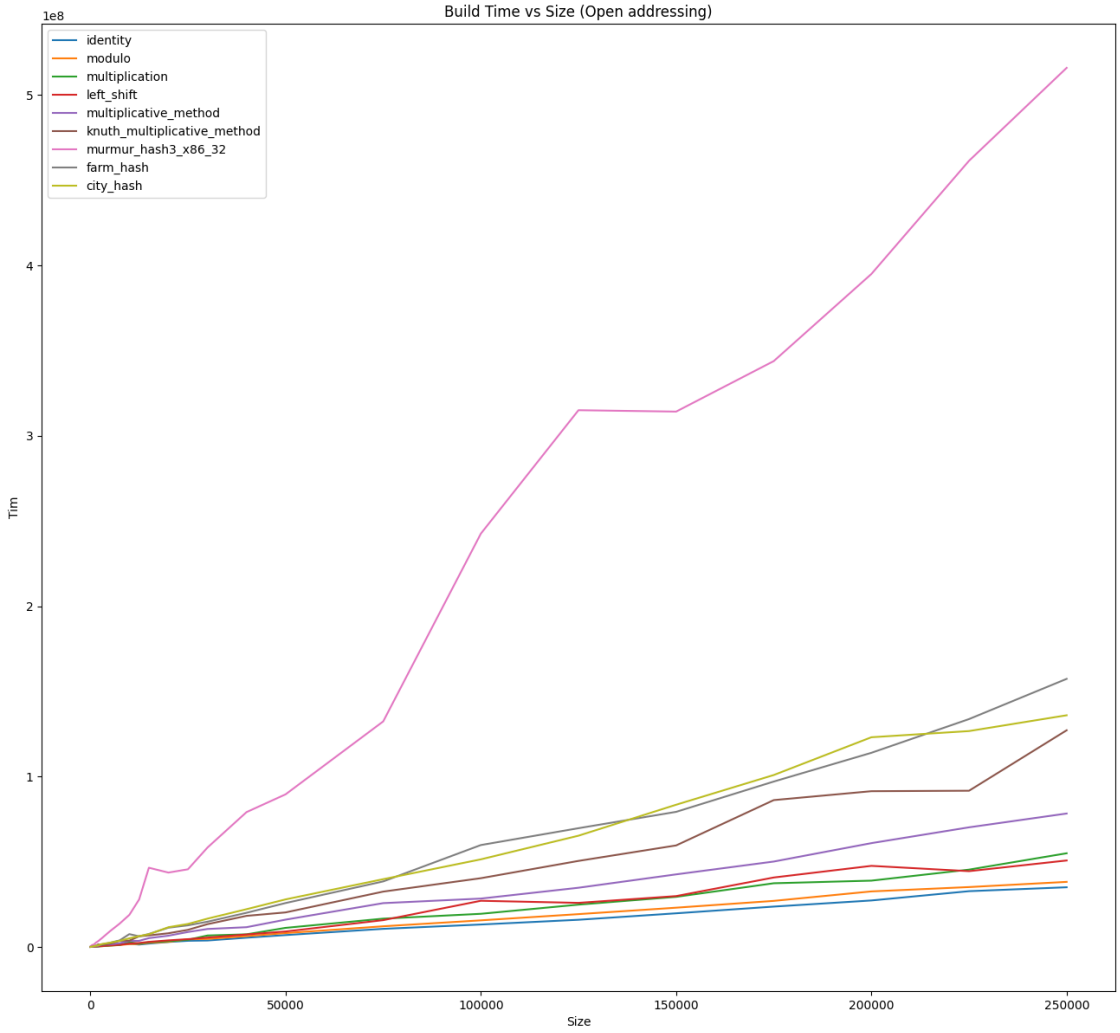
```
    "open_addressing": open_addressing_hash_table,
    "separate_chaining": separate_chaining_hash_table,
}

output = {
    "build":
    process_data(data_build,
                 hash_tables,
                 hash_functions,
                 operation=Operation.INSERT),
    "search":
    process_data(data_search,
                 hash_tables,
                 hash_functions,
                 operation=Operation.SEARCH),
}
```

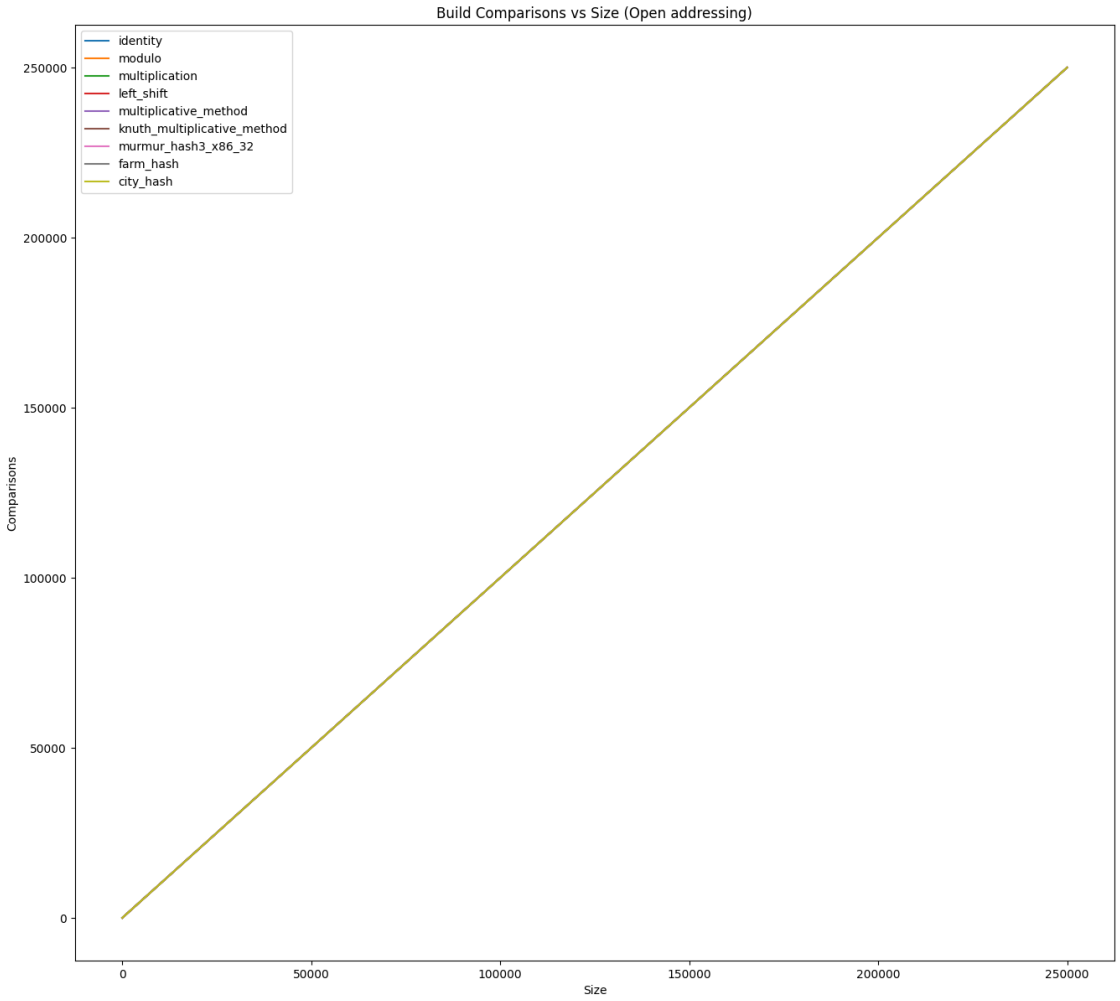
1.7 Analise dos Dados e Discussão

Assim, tendo computado os valores, as visualizações sao geradas a seguir.

[ ]:
```
def plot_results(output: Dict[str, Dict[str, Dict[str, List[Dict[str,
                                                                  Any]]]]]):
    for operation in output:
        for table_type in output[operation]:
            # Collect data for all hash functions to plot them in a single chart
            x_values_list = []
            y_time_values_list = []
            y_comparisons_values_list = []

            for hash_func_name in output[operation][table_type]:
                x_values = [
                    entry["size"]
                    for entry in output[operation][table_type][hash_func_name]
                ]
                time_values = [
                    entry["time"]
                    for entry in output[operation][table_type][hash_func_name]
                ]
                comparisons_values = [
                    entry["comparisons"]
                    for entry in output[operation][table_type][hash_func_name]
                ]

                x_values_list.append(x_values)
                y_time_values_list.append(time_values)
                y_comparisons_values_list.append(comparisons_values)
```

```
            hash_func_names = list(hash_functions.keys())

            # Plot time vs size
            picoplot(
                f"{operation.capitalize()} Time vs Size ({table_type.
 ↪replace('_', ' ').capitalize()})",
                x_values_list,
                y_time_values_list,
                hash_func_names,
                plt.cm.tab10.colors,  # Default matplotlib color cycle
                "Size",
                "Tim")

            # Plot comparisons vs size
            picoplot(
                f"{operation.capitalize()} Comparisons vs Size ({table_type.
 ↪replace('_', ' ').capitalize()})",
                x_values_list,
                y_comparisons_values_list,
                hash_func_names,
                plt.cm.tab10.colors,  # Default matplotlib color cycle
                "Size",
                "Comparisons")

            # Create table for each hash function
            for i, hash_func_name in enumerate(hash_func_names):
                table_data = [
                    x_values_list[i], y_time_values_list[i],
                    y_comparisons_values_list[i]
                ]

                # Transpose the table_data
                table_data_transposed = list(map(list, zip(*table_data)))

                picotable(
                    f"{operation.capitalize()} - {table_type.replace('_', ' ').
 ↪capitalize()} - {hash_func_name}",
                    table_data_transposed, ["Size", "Time", "Comparisons"],
                    [f"Data {j+1}" for j in range(len(x_values_list[i]))],
                    round_digits=2,
                    color="#1ea5c1")


# Call the plot_results function with the output dictionary
plot_results(output)
```


Build Time vs Size (Open addressing)
1e8
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
Tim
Size
0
1
2
3
4
5
0
50000
100000
150000
200000
250000


Build Comparisons vs Size (Open addressing)
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
Comparisons
Size
0
50000
100000
150000
200000
250000

Build - Open addressing - identity

	Size	Time	Comparisons
Data 1	50	5858	50
Data 2	100	9981	100
Data 3	200	22317	200
Data 4	300	32424	300
Data 5	500	57813	500
Data 6	750	84919	750
Data 7	1000	117787	1000
Data 8	1500	196258	1500
Data 9	2000	267192	2000
Data 10	3000	683217	3000
Data 11	5000	1675578	5000
Data 12	7500	1633492	7500
Data 13	10000	1842644	10000
Data 14	12500	1388406	12500
Data 15	15000	1952113	15000
Data 16	20000	2906816	20000
Data 17	25000	3593582	25000
Data 18	30000	3765868	30000
Data 19	40000	5416027	40000
Data 20	50000	6952075	50000
Data 21	75000	10612560	75000
Data 22	100000	13156545	100000
Data 23	125000	15929950	125000
Data 24	150000	19721917	150000
Data 25	175000	23668420	175000
Data 26	200000	27258307	200000
Data 27	225000	32713793	225000
Data 28	250000	35009481	250000

Build - Open addressing - modulo

	Size	Time	Comparisons
Data 1	50	7398	50
Data 2	100	14101	100
Data 3	200	27494	200
Data 4	300	40587	300
Data 5	500	68794	500
Data 6	750	114934	750
Data 7	1000	150720	1000
Data 8	1500	377805	1500
Data 9	2000	488512	2000
Data 10	3000	421258	3000
Data 11	5000	724233	5000
Data 12	7500	1091356	7500
Data 13	10000	1657731	10000
Data 14	12500	1718489	12500
Data 15	15000	2404420	15000
Data 16	20000	2978314	20000
Data 17	25000	4472760	25000
Data 18	30000	4702495	30000
Data 19	40000	6343920	40000
Data 20	50000	8111528	50000
Data 21	75000	12109517	75000
Data 22	100000	15589224	100000
Data 23	125000	19241997	125000
Data 24	150000	23021860	150000
Data 25	175000	27013392	175000
Data 26	200000	32578217	200000
Data 27	225000	35126405	225000
Data 28	250000	38174312	250000

Build - Open addressing - multiplication

	Size	Time	Comparisons
Data 1	50	7681	50
Data 2	100	13310	100
Data 3	200	50743	200
Data 4	300	64634	300
Data 5	500	113244	500
Data 6	750	202298	750
Data 7	1000	305066	1000
Data 8	1500	272113	1500
Data 9	2000	293729	2000
Data 10	3000	455752	3000
Data 11	5000	803518	5000
Data 12	7500	1151220	7500
Data 13	10000	2748219	10000
Data 14	12500	1912877	12500
Data 15	15000	2747963	15000
Data 16	20000	3355155	20000
Data 17	25000	4186766	25000
Data 18	30000	6704212	30000
Data 19	40000	7387383	40000
Data 20	50000	11182649	50000
Data 21	75000	16594329	75000
Data 22	100000	19433929	100000
Data 23	125000	24815218	125000
Data 24	150000	29381785	150000
Data 25	175000	37363403	175000
Data 26	200000	38897450	200000
Data 27	225000	45360399	225000
Data 28	250000	54963858	250000

Build - Open addressing - left_shift

	Size	Time	Comparisons
Data 1	50	9474	50
Data 2	100	15963	100
Data 3	200	32711	200
Data 4	300	50292	300
Data 5	500	94793	500
Data 6	750	258157	750
Data 7	1000	228214	1000
Data 8	1500	261114	1500
Data 9	2000	365728	2000
Data 10	3000	514409	3000
Data 11	5000	910889	5000
Data 12	7500	1264355	7500
Data 13	10000	1922877	10000
Data 14	12500	2319103	12500
Data 15	15000	2965999	15000
Data 16	20000	3852926	20000
Data 17	25000	4548343	25000
Data 18	30000	5479082	30000
Data 19	40000	7302863	40000
Data 20	50000	9064617	50000
Data 21	75000	15707241	75000
Data 22	100000	27078357	100000
Data 23	125000	25850388	125000
Data 24	150000	29829374	150000
Data 25	175000	40740413	175000
Data 26	200000	47568937	200000
Data 27	225000	44459534	225000
Data 28	250000	50729609	250000

Build - Open addressing - multiplicative_method

	Size	Time	Comparisons
Data 1	50	16099	50
Data 2	100	26305	100
Data 3	200	52030	200
Data 4	300	78143	300
Data 5	500	135658	500
Data 6	750	218324	750
Data 7	1000	320578	1000
Data 8	1500	443869	1500
Data 9	2000	646499	2000
Data 10	3000	1488616	3000
Data 11	5000	1435793	5000
Data 12	7500	2208201	7500
Data 13	10000	3490991	10000
Data 14	12500	3585393	12500
Data 15	15000	5158991	15000
Data 16	20000	6587976	20000
Data 17	25000	8754965	25000
Data 18	30000	10522278	30000
Data 19	40000	11580080	40000
Data 20	50000	16003820	50000
Data 21	75000	25695803	75000
Data 22	100000	28370048	100000
Data 23	125000	34726575	125000
Data 24	150000	42574623	150000
Data 25	175000	50102798	175000
Data 26	200000	60908712	200000
Data 27	225000	70187036	225000
Data 28	250000	78290864	250000

Build - Open addressing - knuth_multiplicative_method

	Size	Time	Comparisons
Data 1	50	18941	50
Data 2	100	33493	100
Data 3	200	69412	200
Data 4	300	107615	300
Data 5	500	175667	500
Data 6	750	267053	750
Data 7	1000	360715	1000
Data 8	1500	718003	1500
Data 9	2000	1481833	2000
Data 10	3000	1112526	3000
Data 11	5000	1795921	5000
Data 12	7500	3647493	7500
Data 13	10000	3708944	10000
Data 14	12500	6231086	12500
Data 15	15000	6761460	15000
Data 16	20000	8015506	20000
Data 17	25000	10026693	25000
Data 18	30000	13350754	30000
Data 19	40000	18229226	40000
Data 20	50000	20251017	50000
Data 21	75000	32462469	75000
Data 22	100000	40332512	100000
Data 23	125000	50476140	125000
Data 24	150000	59558667	150000
Data 25	175000	86164923	175000
Data 26	200000	91394037	200000
Data 27	225000	91660650	225000
Data 28	250000	127142785	250000

Build - Open addressing - murmur_hash3_x86_32

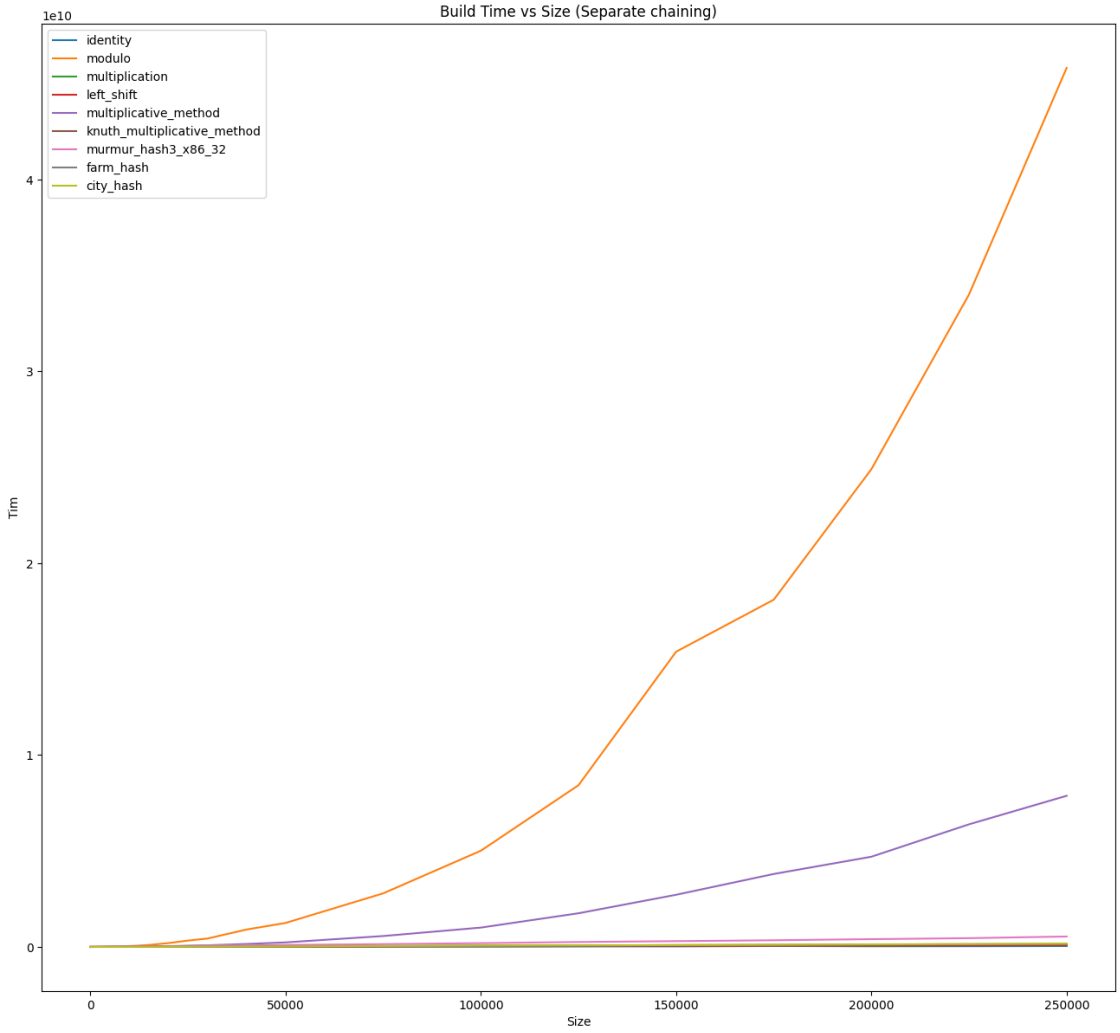
	Size	Time	Comparisons
Data 1	50	155319	50
Data 2	100	173744	100
Data 3	200	648222	200
Data 4	300	726385	300
Data 5	500	1215390	500
Data 6	750	1173276	750
Data 7	1000	1620407	1000
Data 8	1500	2610932	1500
Data 9	2000	3355498	2000
Data 10	3000	5213748	3000
Data 11	5000	9148911	5000
Data 12	7500	13652962	7500
Data 13	10000	18814292	10000
Data 14	12500	27854697	12500
Data 15	15000	46468007	15000
Data 16	20000	43627225	20000
Data 17	25000	45559482	25000
Data 18	30000	58307998	30000
Data 19	40000	79114704	40000
Data 20	50000	89546108	50000
Data 21	75000	132343543	75000
Data 22	100000	242552039	100000
Data 23	125000	314996819	125000
Data 24	150000	314169777	150000
Data 25	175000	343812702	175000
Data 26	200000	394938796	200000
Data 27	225000	461369624	225000
Data 28	250000	515911660	250000

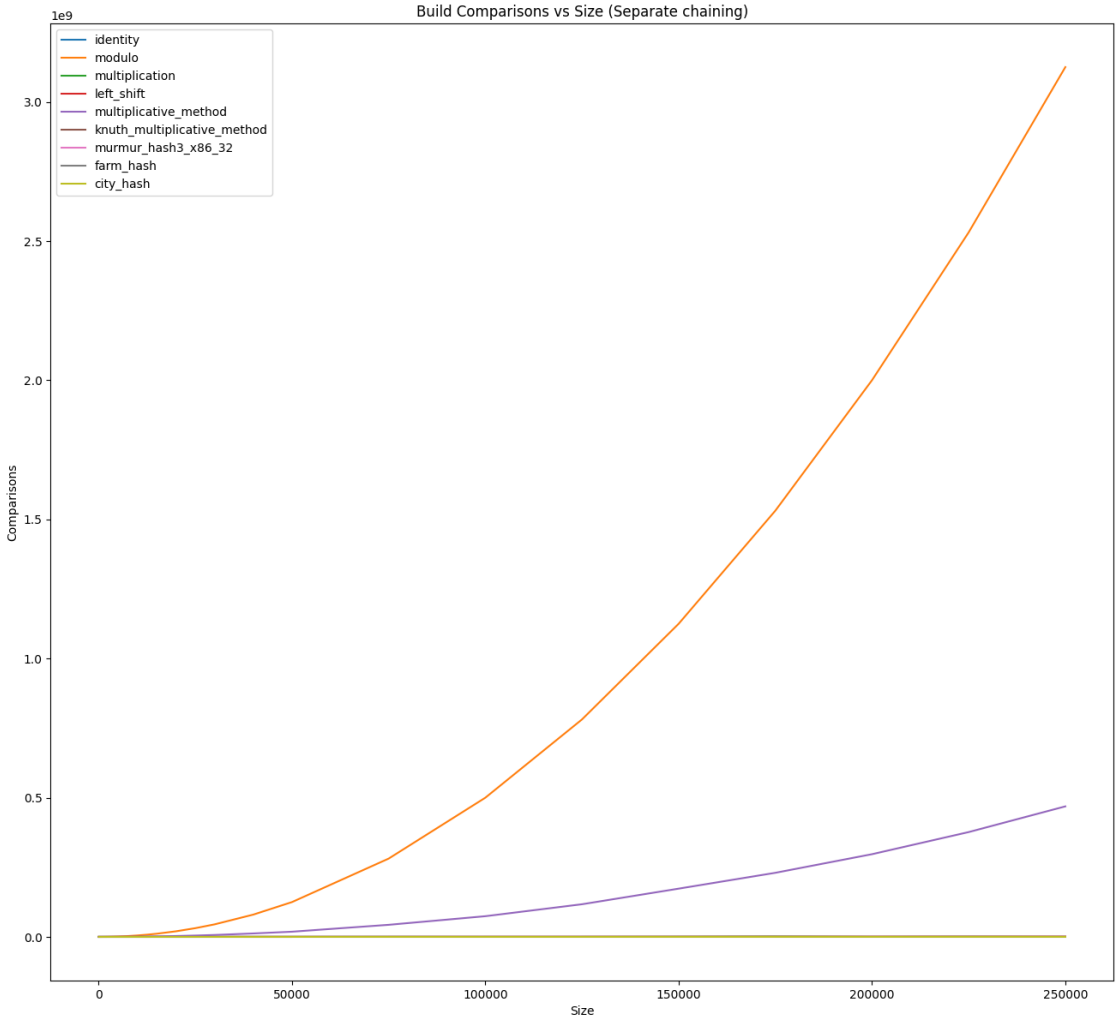
Build - Open addressing - farm_hash

	Size	Time	Comparisons
Data 1	50	26529	50
Data 2	100	42816	100
Data 3	200	86306	200
Data 4	300	131666	300
Data 5	500	225954	500
Data 6	750	335627	750
Data 7	1000	535077	1000
Data 8	1500	781334	1500
Data 9	2000	964414	2000
Data 10	3000	1500652	3000
Data 11	5000	2415045	5000
Data 12	7500	3819123	7500
Data 13	10000	7511470	10000
Data 14	12500	6218165	12500
Data 15	15000	7594908	15000
Data 16	20000	11276874	20000
Data 17	25000	12670041	25000
Data 18	30000	14794923	30000
Data 19	40000	20031205	40000
Data 20	50000	25751962	50000
Data 21	75000	38435514	75000
Data 22	100000	59783550	100000
Data 23	125000	69671264	125000
Data 24	150000	79267296	150000
Data 25	175000	97049090	175000
Data 26	200000	113921302	200000
Data 27	225000	133716253	225000
Data 28	250000	157321170	250000

Build - Open addressing - city_hash

	Size	Time	Comparisons
Data 1	50	24029	50
Data 2	100	43627	100
Data 3	200	87334	200
Data 4	300	133849	300
Data 5	500	226118	500
Data 6	750	343025	750
Data 7	1000	464044	1000
Data 8	1500	694150	1500
Data 9	2000	934501	2000
Data 10	3000	1735971	3000
Data 11	5000	2437256	5000
Data 12	7500	3540971	7500
Data 13	10000	5059304	10000
Data 14	12500	6328740	12500
Data 15	15000	7439900	15000
Data 16	20000	11562790	20000
Data 17	25000	13480738	25000
Data 18	30000	16587991	30000
Data 19	40000	22136209	40000
Data 20	50000	27889875	50000
Data 21	75000	39763579	75000
Data 22	100000	51390610	100000
Data 23	125000	65269419	125000
Data 24	150000	83446138	150000
Data 25	175000	100880370	175000
Data 26	200000	123052979	200000
Data 27	225000	126680798	225000
Data 28	250000	135949771	250000


Build Time vs Size (Separate chaining)
1e10
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
Tim
Size
0
1
2
3
4
0
50000
100000
150000
200000
250000


Build Comparisons vs Size (Separate chaining)
1e9
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
Comparisons
Size
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
50000
100000
150000
200000
250000

Build - Separate chaining - identity

	Size	Time	Comparisons
Data 1	50	15277	12
Data 2	100	18628	27
Data 3	200	33502	53
Data 4	300	50562	78
Data 5	500	86465	128
Data 6	750	120256	183
Data 7	1000	166607	255
Data 8	1500	254405	382
Data 9	2000	314214	503
Data 10	3000	508937	740
Data 11	5000	844880	1247
Data 12	7500	1762970	1871
Data 13	10000	1736057	2516
Data 14	12500	2618172	3158
Data 15	15000	2555609	3759
Data 16	20000	4106599	4679
Data 17	25000	5975518	4538
Data 18	30000	5835135	2350
Data 19	40000	7808064	0
Data 20	50000	8674533	0
Data 21	75000	12687847	0
Data 22	100000	18829580	0
Data 23	125000	26493183	0
Data 24	150000	27879542	0
Data 25	175000	48787594	0
Data 26	200000	33950671	0
Data 27	225000	37202792	0
Data 28	250000	44777555	0

Build - Separate chaining - modulo

	Size	Time	Comparisons
Data 1	50	26692	106
Data 2	100	20745	484
Data 3	200	47757	1938
Data 4	300	77588	4445
Data 5	500	154776	12369
Data 6	750	378252	27844
Data 7	1000	819227	49790
Data 8	1500	1317199	112167
Data 9	2000	1631818	199713
Data 10	3000	3460906	448964
Data 11	5000	10449430	1248584
Data 12	7500	22744717	2810073
Data 13	10000	44340594	4998666
Data 14	12500	68511420	7811583
Data 15	15000	101855477	11244286
Data 16	20000	197990110	19996054
Data 17	25000	330207064	31242014
Data 18	30000	436782826	44985464
Data 19	40000	905623487	79980000
Data 20	50000	1249430789	124975000
Data 21	75000	2791911165	281212500
Data 22	100000	5013221924	499950000
Data 23	125000	8427622800	781187500
Data 24	150000	15389275550	1124925000
Data 25	175000	18099801166	1531162500
Data 26	200000	24904676724	1999900000
Data 27	225000	34013835136	2531137500
Data 28	250000	45825494456	3124875000

Build - Separate chaining - multiplication

	Size	Time	Comparisons
Data 1	50	32239	37
Data 2	100	42767	79
Data 3	200	36184	155
Data 4	300	63416	828
Data 5	500	100469	378
Data 6	750	154662	2059
Data 7	1000	234031	740
Data 8	1500	639844	4110
Data 9	2000	663160	1482
Data 10	3000	1035190	8273
Data 11	5000	1095280	3751
Data 12	7500	1896018	20571
Data 13	10000	3027626	7485
Data 14	12500	3146396	9393
Data 15	15000	4268178	41381
Data 16	20000	5570808	14199
Data 17	25000	7296681	16334
Data 18	30000	10137511	76816
Data 19	40000	11494751	20000
Data 20	50000	18971880	25000
Data 21	75000	30614914	187500
Data 22	100000	28891203	50000
Data 23	125000	38569687	62500
Data 24	150000	57586586	375000
Data 25	175000	76781673	1137500
Data 26	200000	60017830	100000
Data 27	225000	89036872	562500
Data 28	250000	78616291	125000

Build - Separate chaining - left_shift

	Size	Time	Comparisons
Data 1	50	27893	37
Data 2	100	21193	183
Data 3	200	44809	753
Data 4	300	64820	539
Data 5	500	132683	849
Data 6	750	162880	564
Data 7	1000	257368	3777
Data 8	1500	386903	2628
Data 9	2000	823495	7544
Data 10	3000	1014440	11177
Data 11	5000	1443325	18716
Data 12	7500	1966395	13094
Data 13	10000	4023142	37583
Data 14	12500	3960458	21939
Data 15	15000	4511295	56356
Data 16	20000	7027702	74029
Data 17	25000	9466993	90718
Data 18	30000	14393318	106571
Data 19	40000	16717385	140000
Data 20	50000	19290524	175000
Data 21	75000	28351620	262500
Data 22	100000	36770957	350000
Data 23	125000	45811656	437500
Data 24	150000	53313403	525000
Data 25	175000	61075403	612500
Data 26	200000	68675290	700000
Data 27	225000	76269770	787500
Data 28	250000	87620733	875000

Build - Separate chaining - multiplicative_method

	Size	Time	Comparisons
Data 1	50	38755	23
Data 2	100	28552	50
Data 3	200	59695	238
Data 4	300	93415	646
Data 5	500	164622	1787
Data 6	750	259162	3915
Data 7	1000	369002	7181
Data 8	1500	631068	16317
Data 9	2000	859897	29482
Data 10	3000	1507369	66364
Data 11	5000	4363843	191109
Data 12	7500	6475444	415525
Data 13	10000	12429975	758597
Data 14	12500	15590737	1151104
Data 15	15000	21919292	1673793
Data 16	20000	38629876	3013500
Data 17	25000	61639513	4727107
Data 18	30000	85871322	6776164
Data 19	40000	153154123	12274451
Data 20	50000	235037618	18540827
Data 21	75000	567041093	43291385
Data 22	100000	1007127202	74218153
Data 23	125000	1753536465	117159360
Data 24	150000	2717609268	173253285
Data 25	175000	3801965738	230135764
Data 26	200000	4701062547	296995819
Data 27	225000	6390818074	376455486
Data 28	250000	7877844205	468800111

Build - Separate chaining - knuth_multiplicative_method

	Size	Time	Comparisons
Data 1	50	170713	15
Data 2	100	126639	29
Data 3	200	231178	47
Data 4	300	311982	87
Data 5	500	542796	143
Data 6	750	840914	234
Data 7	1000	537387	308
Data 8	1500	576158	450
Data 9	2000	796525	576
Data 10	3000	1224004	899
Data 11	5000	3356407	1450
Data 12	7500	3270096	2276
Data 13	10000	5685218	2914
Data 14	12500	6747097	3789
Data 15	15000	8094072	3991
Data 16	20000	10550529	5267
Data 17	25000	14348047	4721
Data 18	30000	16844792	4865
Data 19	40000	20598741	4210
Data 20	50000	26150018	6571
Data 21	75000	39181294	7919
Data 22	100000	54842574	11995
Data 23	125000	63735694	14689
Data 24	150000	79896181	20734
Data 25	175000	94232415	19048
Data 26	200000	101143860	22642
Data 27	225000	119147812	31667
Data 28	250000	130314137	33243

Build - Separate chaining - murmur_hash3_x86_32

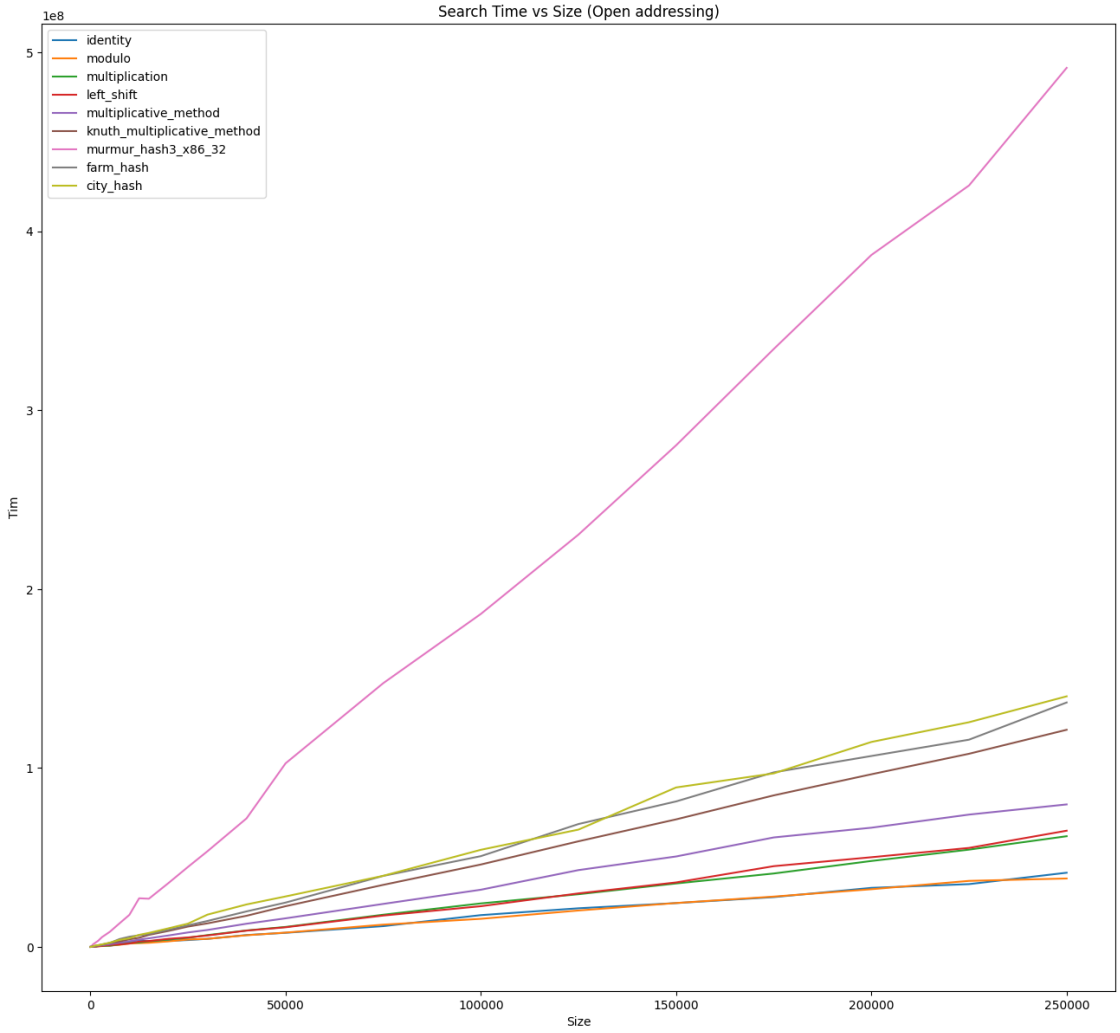
	Size	Time	Comparisons
Data 1	50	149423	26
Data 2	100	139117	57
Data 3	200	285530	107
Data 4	300	483140	155
Data 5	500	758749	247
Data 6	750	1122008	381
Data 7	1000	1649733	499
Data 8	1500	2473661	777
Data 9	2000	3906857	1001
Data 10	3000	5251361	1491
Data 11	5000	9417335	2435
Data 12	7500	14182185	3723
Data 13	10000	20589462	5034
Data 14	12500	23829431	6254
Data 15	15000	28482578	7383
Data 16	20000	38153232	10028
Data 17	25000	47680158	12637
Data 18	30000	77917865	15102
Data 19	40000	78767112	20197
Data 20	50000	100752239	25101
Data 21	75000	145376631	37392
Data 22	100000	196591481	49905
Data 23	125000	255862571	62973
Data 24	150000	300544501	74951
Data 25	175000	347374934	87729
Data 26	200000	404243774	99750
Data 27	225000	456242332	111789
Data 28	250000	541538389	125609

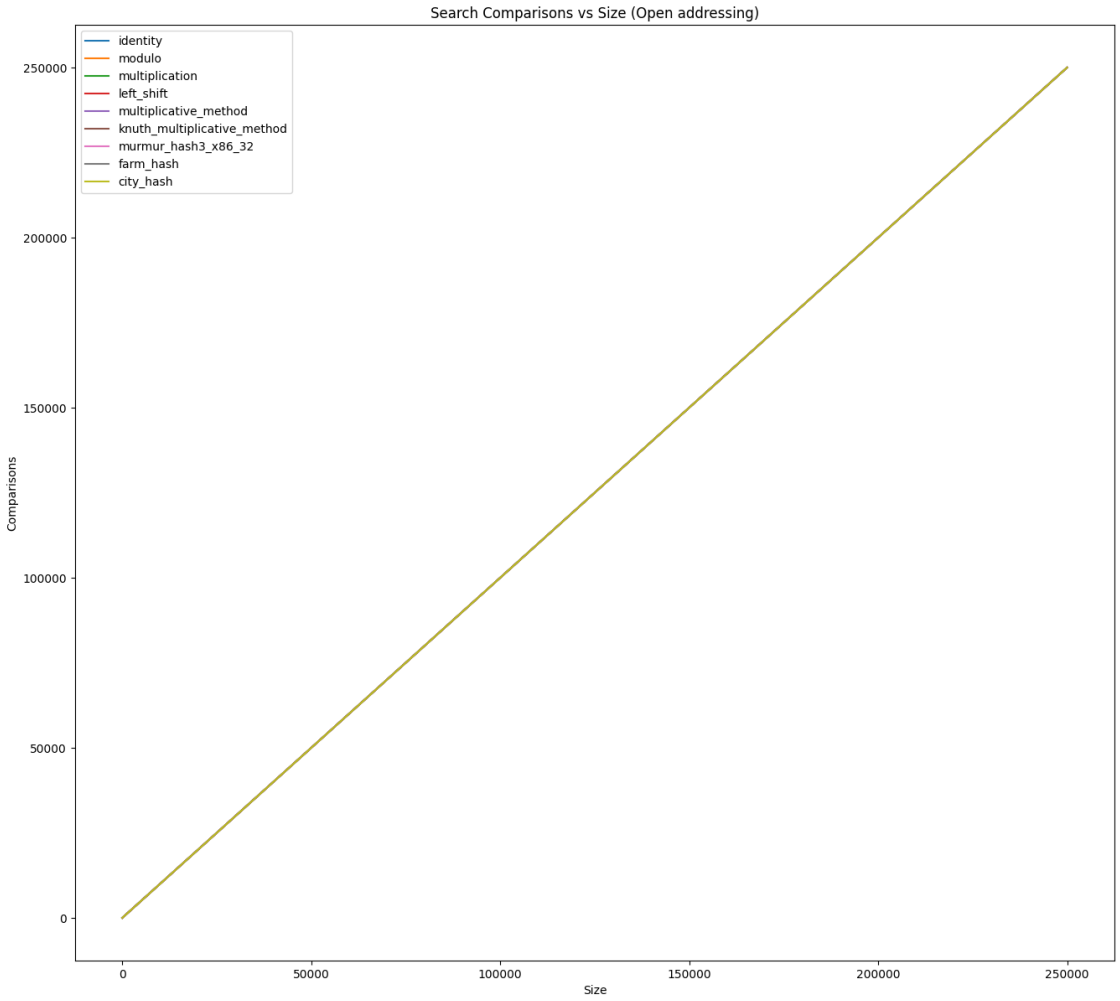
Build - Separate chaining - farm_hash

	Size	Time	Comparisons
Data 1	50	45703	41
Data 2	100	38980	49
Data 3	200	99739	102
Data 4	300	147063	156
Data 5	500	234711	234
Data 6	750	309354	378
Data 7	1000	438860	499
Data 8	1500	646897	734
Data 9	2000	1253154	998
Data 10	3000	1619830	1543
Data 11	5000	2422578	2577
Data 12	7500	4079074	3810
Data 13	10000	5102528	4999
Data 14	12500	6911604	6382
Data 15	15000	8310162	7515
Data 16	20000	20336049	10096
Data 17	25000	21262846	12373
Data 18	30000	19661617	14981
Data 19	40000	24803376	20079
Data 20	50000	32871099	25070
Data 21	75000	48903153	37693
Data 22	100000	65579289	50068
Data 23	125000	75564292	62434
Data 24	150000	96035792	75156
Data 25	175000	106663226	87006
Data 26	200000	124109383	100490
Data 27	225000	142382108	112517
Data 28	250000	155037585	124636

Build - Separate chaining - city_hash

	Size	Time	Comparisons
Data 1	50	44519	22
Data 2	100	42012	52
Data 3	200	85724	89
Data 4	300	126492	144
Data 5	500	215758	238
Data 6	750	331644	372
Data 7	1000	471466	498
Data 8	1500	713765	741
Data 9	2000	922500	1050
Data 10	3000	1555961	1482
Data 11	5000	3024987	2444
Data 12	7500	4022742	3700
Data 13	10000	5908517	4983
Data 14	12500	8002986	6333
Data 15	15000	8602500	7445
Data 16	20000	11531691	9936
Data 17	25000	14339604	12732
Data 18	30000	18117853	15044
Data 19	40000	23201973	19811
Data 20	50000	29720905	24904
Data 21	75000	47082877	37319
Data 22	100000	67802408	50217
Data 23	125000	79460539	62721
Data 24	150000	93103104	75119
Data 25	175000	110974168	87618
Data 26	200000	133740928	99961
Data 27	225000	148858097	112517
Data 28	250000	170023473	125544


Search Time vs Size (Open addressing)
1e8
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
Tim
Size
0
1
2
3
4
5
0
50000
100000
150000
200000
250000


Search Comparisons vs Size (Open addressing)
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
Comparisons
Size
0
50000
100000
150000
200000
250000

Search - Open addressing - identity

	Size	Time	Comparisons
Data 1	50	5873	50
Data 2	100	9503	100
Data 3	200	19917	200
Data 4	300	30486	300
Data 5	500	55046	500
Data 6	750	86494	750
Data 7	1000	104324	1000
Data 8	1500	196970	1500
Data 9	2000	582094	2000
Data 10	3000	532726	3000
Data 11	5000	572797	5000
Data 12	7500	1657465	7500
Data 13	10000	2924139	10000
Data 14	12500	3789026	12500
Data 15	15000	3018627	15000
Data 16	20000	3379326	20000
Data 17	25000	3850873	25000
Data 18	30000	4519116	30000
Data 19	40000	6636585	40000
Data 20	50000	7898023	50000
Data 21	75000	11612201	75000
Data 22	100000	17774870	100000
Data 23	125000	21603169	125000
Data 24	150000	24501367	150000
Data 25	175000	27836352	175000
Data 26	200000	33011828	200000
Data 27	225000	35111338	225000
Data 28	250000	41477497	250000

Search - Open addressing - modulo

	Size	Time	Comparisons
Data 1	50	7518	50
Data 2	100	13545	100
Data 3	200	27545	200
Data 4	300	42200	300
Data 5	500	71001	500
Data 6	750	106330	750
Data 7	1000	145362	1000
Data 8	1500	255344	1500
Data 9	2000	329606	2000
Data 10	3000	465714	3000
Data 11	5000	728813	5000
Data 12	7500	1178797	7500
Data 13	10000	1894075	10000
Data 14	12500	2113605	12500
Data 15	15000	2309108	15000
Data 16	20000	3106431	20000
Data 17	25000	4131597	25000
Data 18	30000	4517105	30000
Data 19	40000	6523717	40000
Data 20	50000	7993356	50000
Data 21	75000	12453726	75000
Data 22	100000	15706967	100000
Data 23	125000	20364675	125000
Data 24	150000	24542015	150000
Data 25	175000	28141295	175000
Data 26	200000	32248108	200000
Data 27	225000	36917316	225000
Data 28	250000	38252962	250000

Search - Open addressing - multiplication

	Size	Time	Comparisons
Data 1	50	8667	50
Data 2	100	17148	100
Data 3	200	39884	200
Data 4	300	58335	300
Data 5	500	137551	500
Data 6	750	182210	750
Data 7	1000	224012	1000
Data 8	1500	315464	1500
Data 9	2000	449245	2000
Data 10	3000	540511	3000
Data 11	5000	897804	5000
Data 12	7500	1377319	7500
Data 13	10000	2265391	10000
Data 14	12500	2456377	12500
Data 15	15000	3183592	15000
Data 16	20000	4146669	20000
Data 17	25000	5100151	25000
Data 18	30000	6602917	30000
Data 19	40000	9193205	40000
Data 20	50000	11168050	50000
Data 21	75000	17998703	75000
Data 22	100000	24320970	100000
Data 23	125000	29453656	125000
Data 24	150000	35479185	150000
Data 25	175000	41048772	175000
Data 26	200000	48045678	200000
Data 27	225000	54389287	225000
Data 28	250000	61884015	250000

Search - Open addressing - left_shift

	Size	Time	Comparisons
Data 1	50	9724	50
Data 2	100	16106	100
Data 3	200	36835	200
Data 4	300	58668	300
Data 5	500	101666	500
Data 6	750	148106	750
Data 7	1000	180525	1000
Data 8	1500	293864	1500
Data 9	2000	375798	2000
Data 10	3000	589078	3000
Data 11	5000	1129707	5000
Data 12	7500	1427516	7500
Data 13	10000	1895256	10000
Data 14	12500	2986018	12500
Data 15	15000	3315681	15000
Data 16	20000	4529633	20000
Data 17	25000	5284810	25000
Data 18	30000	6393564	30000
Data 19	40000	9101633	40000
Data 20	50000	10980996	50000
Data 21	75000	17532583	75000
Data 22	100000	22756278	100000
Data 23	125000	29923948	125000
Data 24	150000	36051145	150000
Data 25	175000	45143984	175000
Data 26	200000	50137467	200000
Data 27	225000	55339408	225000
Data 28	250000	64971173	250000

Search - Open addressing - multiplicative_method

	Size	Time	Comparisons
Data 1	50	29400	50
Data 2	100	32898	100
Data 3	200	78691	200
Data 4	300	87177	300
Data 5	500	156043	500
Data 6	750	224894	750
Data 7	1000	311170	1000
Data 8	1500	451069	1500
Data 9	2000	649149	2000
Data 10	3000	923493	3000
Data 11	5000	1539713	5000
Data 12	7500	2420366	7500
Data 13	10000	3153364	10000
Data 14	12500	4051346	12500
Data 15	15000	4828084	15000
Data 16	20000	6399327	20000
Data 17	25000	8085235	25000
Data 18	30000	9473869	30000
Data 19	40000	12966168	40000
Data 20	50000	15981604	50000
Data 21	75000	24038602	75000
Data 22	100000	31975645	100000
Data 23	125000	42946714	125000
Data 24	150000	50569300	150000
Data 25	175000	61203099	175000
Data 26	200000	66641948	200000
Data 27	225000	73973482	225000
Data 28	250000	79664831	250000

Search - Open addressing - knuth_multiplicative_method

	Size	Time	Comparisons
Data 1	50	19710	50
Data 2	100	34936	100
Data 3	200	76295	200
Data 4	300	113827	300
Data 5	500	195690	500
Data 6	750	294889	750
Data 7	1000	381985	1000
Data 8	1500	555824	1500
Data 9	2000	786405	2000
Data 10	3000	1196193	3000
Data 11	5000	2206394	5000
Data 12	7500	3063025	7500
Data 13	10000	4255331	10000
Data 14	12500	5119920	12500
Data 15	15000	6720586	15000
Data 16	20000	8927058	20000
Data 17	25000	11378245	25000
Data 18	30000	13164871	30000
Data 19	40000	17405158	40000
Data 20	50000	22758987	50000
Data 21	75000	34686229	75000
Data 22	100000	46071591	100000
Data 23	125000	59047081	125000
Data 24	150000	71308439	150000
Data 25	175000	84715172	175000
Data 26	200000	96513874	200000
Data 27	225000	107981155	225000
Data 28	250000	121383657	250000

Search - Open addressing - murmur_hash3_x86_32

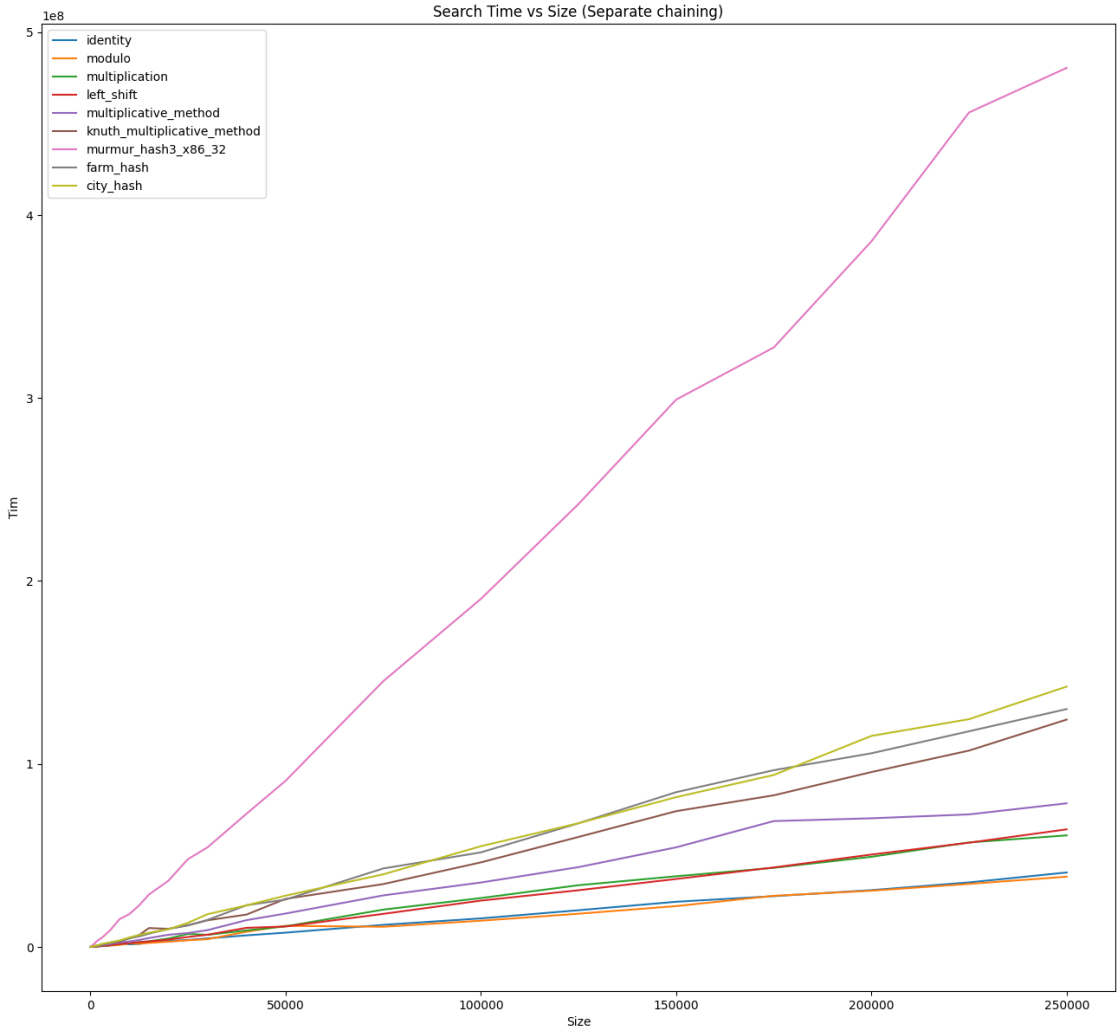
	Size	Time	Comparisons
Data 1	50	93869	50
Data 2	100	159489	100
Data 3	200	322312	200
Data 4	300	515500	300
Data 5	500	865513	500
Data 6	750	1557769	750
Data 7	1000	1871103	1000
Data 8	1500	2659979	1500
Data 9	2000	3405093	2000
Data 10	3000	5488200	3000
Data 11	5000	8454325	5000
Data 12	7500	13214765	7500
Data 13	10000	17901947	10000
Data 14	12500	27193847	12500
Data 15	15000	26941172	15000
Data 16	20000	35679655	20000
Data 17	25000	44700513	25000
Data 18	30000	53483402	30000
Data 19	40000	71841082	40000
Data 20	50000	102634261	50000
Data 21	75000	147429355	75000
Data 22	100000	186145257	100000
Data 23	125000	230502747	125000
Data 24	150000	280423891	150000
Data 25	175000	334249146	175000
Data 26	200000	386805472	200000
Data 27	225000	425675096	225000
Data 28	250000	491420873	250000

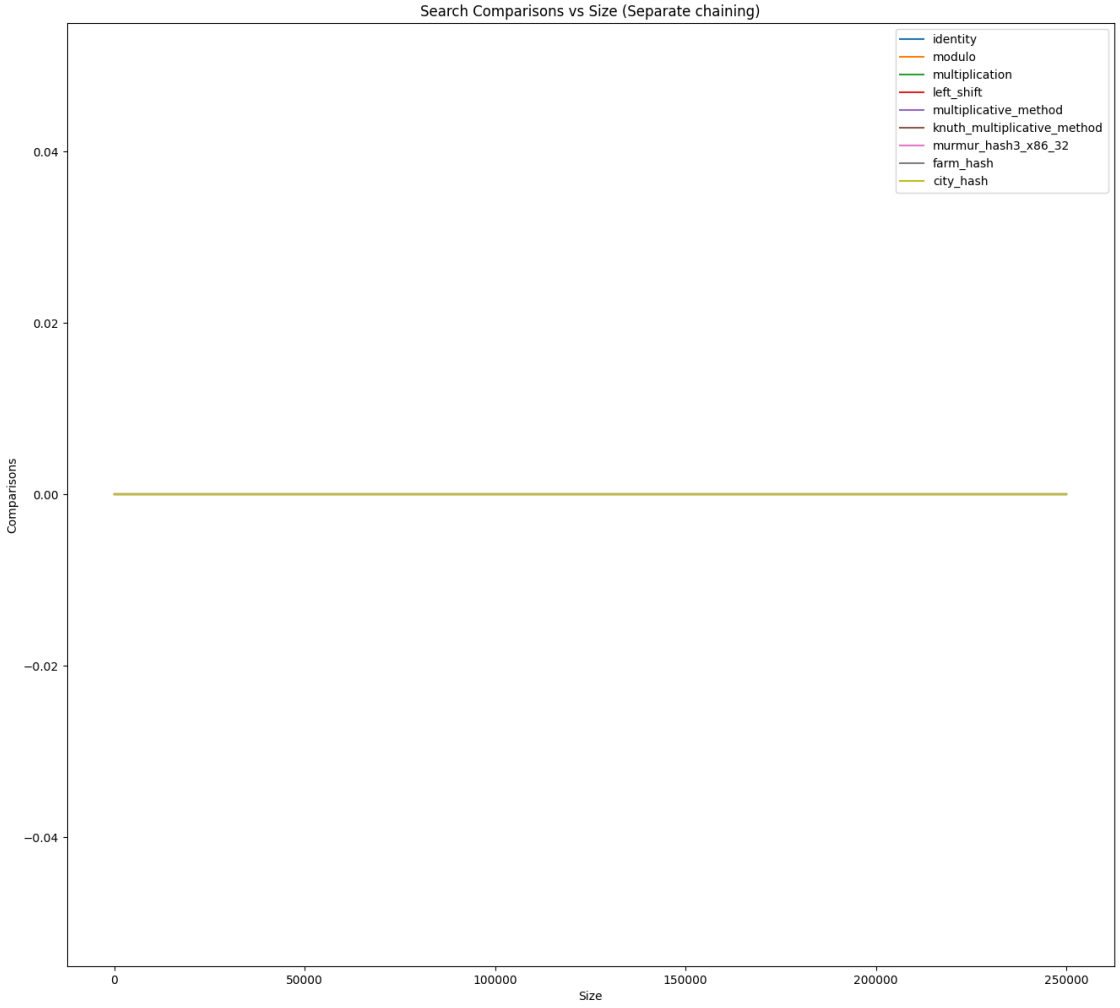
Search - Open addressing - farm_hash

	Size	Time	Comparisons
Data 1	50	38475	50
Data 2	100	43449	100
Data 3	200	128575	200
Data 4	300	198917	300
Data 5	500	477366	500
Data 6	750	331855	750
Data 7	1000	460570	1000
Data 8	1500	714215	1500
Data 9	2000	938149	2000
Data 10	3000	1364257	3000
Data 11	5000	2215921	5000
Data 12	7500	4411698	7500
Data 13	10000	5689487	10000
Data 14	12500	6526316	12500
Data 15	15000	7222794	15000
Data 16	20000	9542003	20000
Data 17	25000	12060347	25000
Data 18	30000	14502076	30000
Data 19	40000	19849684	40000
Data 20	50000	24783296	50000
Data 21	75000	39702786	75000
Data 22	100000	50745711	100000
Data 23	125000	68697534	125000
Data 24	150000	81338159	150000
Data 25	175000	97584476	175000
Data 26	200000	106730584	200000
Data 27	225000	115852088	225000
Data 28	250000	136646035	250000

Search - Open addressing - city_hash

	Size	Time	Comparisons
Data 1	50	24013	50
Data 2	100	45805	100
Data 3	200	90040	200
Data 4	300	138966	300
Data 5	500	234714	500
Data 6	750	352400	750
Data 7	1000	597297	1000
Data 8	1500	888747	1500
Data 9	2000	952377	2000
Data 10	3000	1417162	3000
Data 11	5000	2375010	5000
Data 12	7500	3931385	7500
Data 13	10000	4808909	10000
Data 14	12500	6736747	12500
Data 15	15000	7825103	15000
Data 16	20000	10387606	20000
Data 17	25000	13006119	25000
Data 18	30000	18047186	30000
Data 19	40000	23742440	40000
Data 20	50000	28215697	50000
Data 21	75000	39757268	75000
Data 22	100000	54292338	100000
Data 23	125000	65582131	125000
Data 24	150000	89142335	150000
Data 25	175000	97008055	175000
Data 26	200000	114577794	200000
Data 27	225000	125607603	225000
Data 28	250000	140088287	250000


Search Time vs Size (Separate chaining)
1e8
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
Tim
Size
0
1
2
3
4
5
0
50000
100000
150000
200000
250000


Search Comparisons vs Size (Separate chaining)
identity
modulo
multiplication
left_shift
multiplicative_method
knuth_multiplicative_method
murmur_hash3_x86_32
farm_hash
city_hash
0.04
0.02
0.00
−0.02
−0.04
Comparisons
0
50000
100000
150000
200000
250000
Size

Search - Separate chaining - identity

	Size	Time	Comparisons
Data 1	50	5671	0
Data 2	100	10633	0
Data 3	200	23578	0
Data 4	300	34738	0
Data 5	500	66512	0
Data 6	750	100805	0
Data 7	1000	126499	0
Data 8	1500	206848	0
Data 9	2000	362232	0
Data 10	3000	428303	0
Data 11	5000	744970	0
Data 12	7500	2288792	0
Data 13	10000	1431009	0
Data 14	12500	1618759	0
Data 15	15000	2454848	0
Data 16	20000	3247957	0
Data 17	25000	3803116	0
Data 18	30000	4655515	0
Data 19	40000	6319120	0
Data 20	50000	7795419	0
Data 21	75000	12058897	0
Data 22	100000	15587900	0
Data 23	125000	20050342	0
Data 24	150000	24680620	0
Data 25	175000	27734651	0
Data 26	200000	31036050	0
Data 27	225000	35251027	0
Data 28	250000	40684310	0

Search - Separate chaining - modulo

	Size	Time	Comparisons
Data 1	50	8365	0
Data 2	100	13410	0
Data 3	200	28812	0
Data 4	300	52939	0
Data 5	500	87841	0
Data 6	750	121252	0
Data 7	1000	151024	0
Data 8	1500	211211	0
Data 9	2000	317694	0
Data 10	3000	448760	0
Data 11	5000	706835	0
Data 12	7500	1116513	0
Data 13	10000	1870413	0
Data 14	12500	1767763	0
Data 15	15000	2211894	0
Data 16	20000	2843922	0
Data 17	25000	3624940	0
Data 18	30000	4224271	0
Data 19	40000	8284332	0
Data 20	50000	11539227	0
Data 21	75000	11013700	0
Data 22	100000	14323519	0
Data 23	125000	18142124	0
Data 24	150000	22313645	0
Data 25	175000	27939062	0
Data 26	200000	30714026	0
Data 27	225000	34439176	0
Data 28	250000	38373972	0

Search - Separate chaining - multiplication

	Size	Time	Comparisons
Data 1	50	12208	0
Data 2	100	20187	0
Data 3	200	66832	0
Data 4	300	140641	0
Data 5	500	239096	0
Data 6	750	363702	0
Data 7	1000	318509	0
Data 8	1500	254710	0
Data 9	2000	348603	0
Data 10	3000	533699	0
Data 11	5000	876217	0
Data 12	7500	1741257	0
Data 13	10000	1913783	0
Data 14	12500	2636807	0
Data 15	15000	3086182	0
Data 16	20000	4583059	0
Data 17	25000	7095927	0
Data 18	30000	6594520	0
Data 19	40000	9026636	0
Data 20	50000	11236627	0
Data 21	75000	20382874	0
Data 22	100000	26698305	0
Data 23	125000	33726314	0
Data 24	150000	38616400	0
Data 25	175000	43224277	0
Data 26	200000	49255358	0
Data 27	225000	57100489	0
Data 28	250000	60949663	0

Search - Separate chaining - left_shift

	Size	Time	Comparisons
Data 1	50	11056	0
Data 2	100	16186	0
Data 3	200	33262	0
Data 4	300	52661	0
Data 5	500	100390	0
Data 6	750	138845	0
Data 7	1000	182318	0
Data 8	1500	272956	0
Data 9	2000	365569	0
Data 10	3000	542447	0
Data 11	5000	930718	0
Data 12	7500	1615029	0
Data 13	10000	2077960	0
Data 14	12500	2582287	0
Data 15	15000	2942000	0
Data 16	20000	4020702	0
Data 17	25000	5425719	0
Data 18	30000	6671454	0
Data 19	40000	10442378	0
Data 20	50000	11149781	0
Data 21	75000	18052581	0
Data 22	100000	25307749	0
Data 23	125000	30976248	0
Data 24	150000	37126402	0
Data 25	175000	43432590	0
Data 26	200000	50491346	0
Data 27	225000	56965669	0
Data 28	250000	64253153	0

Search - Separate chaining - multiplicative_method

	Size	Time	Comparisons
Data 1	50	18215	0
Data 2	100	24367	0
Data 3	200	53824	0
Data 4	300	84520	0
Data 5	500	142126	0
Data 6	750	213128	0
Data 7	1000	288306	0
Data 8	1500	444678	0
Data 9	2000	595358	0
Data 10	3000	871376	0
Data 11	5000	1508475	0
Data 12	7500	2518632	0
Data 13	10000	2976858	0
Data 14	12500	3816005	0
Data 15	15000	4833260	0
Data 16	20000	6643836	0
Data 17	25000	7592088	0
Data 18	30000	9171479	0
Data 19	40000	14627227	0
Data 20	50000	18247711	0
Data 21	75000	28151994	0
Data 22	100000	35197120	0
Data 23	125000	43600819	0
Data 24	150000	54426857	0
Data 25	175000	68778991	0
Data 26	200000	70308442	0
Data 27	225000	72425716	0
Data 28	250000	78480271	0

Search - Separate chaining - knuth_multiplicative_method

	Size	Time	Comparisons
Data 1	50	22566	0
Data 2	100	33310	0
Data 3	200	69784	0
Data 4	300	136266	0
Data 5	500	173722	0
Data 6	750	269374	0
Data 7	1000	370569	0
Data 8	1500	545203	0
Data 9	2000	755350	0
Data 10	3000	1138902	0
Data 11	5000	2288353	0
Data 12	7500	3034643	0
Data 13	10000	4653780	0
Data 14	12500	6479383	0
Data 15	15000	10298163	0
Data 16	20000	9889365	0
Data 17	25000	11838391	0
Data 18	30000	14547005	0
Data 19	40000	17683733	0
Data 20	50000	26257147	0
Data 21	75000	34340209	0
Data 22	100000	46198975	0
Data 23	125000	60099965	0
Data 24	150000	74194814	0
Data 25	175000	82856764	0
Data 26	200000	95573249	0
Data 27	225000	107305280	0
Data 28	250000	124256011	0

Search - Separate chaining - murmur_hash3_x86_32

	Size	Time	Comparisons
Data 1	50	92455	0
Data 2	100	154579	0
Data 3	200	311976	0
Data 4	300	482328	0
Data 5	500	823111	0
Data 6	750	1245202	0
Data 7	1000	1649778	0
Data 8	1500	2763048	0
Data 9	2000	3686312	0
Data 10	3000	5045140	0
Data 11	5000	8909984	0
Data 12	7500	15201762	0
Data 13	10000	17954573	0
Data 14	12500	22692450	0
Data 15	15000	28608077	0
Data 16	20000	36190093	0
Data 17	25000	47981446	0
Data 18	30000	54399017	0
Data 19	40000	72792439	0
Data 20	50000	90795820	0
Data 21	75000	145218737	0
Data 22	100000	190183579	0
Data 23	125000	242059365	0
Data 24	150000	299116025	0
Data 25	175000	327634899	0
Data 26	200000	385626969	0
Data 27	225000	456096031	0
Data 28	250000	480464105	0

Search - Separate chaining - farm_hash

	Size	Time	Comparisons
Data 1	50	25920	0
Data 2	100	41724	0
Data 3	200	83098	0
Data 4	300	127910	0
Data 5	500	217214	0
Data 6	750	333127	0
Data 7	1000	443455	0
Data 8	1500	650005	0
Data 9	2000	845809	0
Data 10	3000	1289655	0
Data 11	5000	2401163	0
Data 12	7500	3344620	0
Data 13	10000	4721850	0
Data 14	12500	5713744	0
Data 15	15000	7231523	0
Data 16	20000	9820373	0
Data 17	25000	11753319	0
Data 18	30000	14741451	0
Data 19	40000	22774638	0
Data 20	50000	25894278	0
Data 21	75000	42871445	0
Data 22	100000	51668684	0
Data 23	125000	67581914	0
Data 24	150000	84568799	0
Data 25	175000	96589649	0
Data 26	200000	105827149	0
Data 27	225000	117866172	0
Data 28	250000	129985530	0

Search - Separate chaining - city_hash

	Size	Time	Comparisons
Data 1	50	25348	0
Data 2	100	43168	0
Data 3	200	86202	0
Data 4	300	131171	0
Data 5	500	264726	0
Data 6	750	337792	0
Data 7	1000	450657	0
Data 8	1500	706388	0
Data 9	2000	941668	0
Data 10	3000	1691887	0
Data 11	5000	2383082	0
Data 12	7500	3515530	0
Data 13	10000	5186727	0
Data 14	12500	6486708	0
Data 15	15000	7755237	0
Data 16	20000	9616242	0
Data 17	25000	13181227	0
Data 18	30000	17875261	0
Data 19	40000	22734188	0
Data 20	50000	28019521	0
Data 21	75000	39674197	0
Data 22	100000	55034103	0
Data 23	125000	67625550	0
Data 24	150000	81846878	0
Data 25	175000	93981034	0
Data 26	200000	115327147	0
Data 27	225000	124462985	0
Data 28	250000	142282878	0

1.8 Conclusões

TODO.