1/19/2022

Hash Tables Assignment

Data Structures and Algorithms

Mashal Ashfaque
337203
CS 10-A

Using netflix-2 dataset, and parsing all the movie titles and storing them into hash tables.

We have to use two collision resolution techniques. I have used linear probing, and double hashing.

1- Linear probing:

Code:

```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#define size 6235
using namespace std;

class hashtable {
public:
	string arr[size];
	hashtable()
	{
		for (int i = 0; i < size; i++)
			arr[i] = "";
	}

	int hashFunction(string x) // For string
	{
		int hash =0;
		for (int i = 0; i < x.length(); i++)
		{
			hash += (int(x[i])); // taking sum of ascii values
		}
		return abs(hash % size);
	}

	void insertstring(string s)
	{ //done by linear probing
		int index = hashFunction(s);
		for (int i = 0; i < size; i++)
		{
			if (arr[(index + i) % size] == "")  //to check if cell is empty or
not.
			{
				arr[(index + i) % size] = s;
				return;
			}
		}
	}
	void findindex(string s)
	{
		int index=hashFunction(s);
		while (arr[index] != s)
		{
			index++;
		}
```

```
		cout << s << " is at index # " << index;
	}
	void printhashtable()
	{
		cout << "ALL MOVIES AND SHOWS: " << endl;
		for (int i = 0; i < size; i++)
		{
			cout<<i<<"\t" << arr[i] << endl;
		}
	}
};
hashtable titles;
void readfile()
{
	string data;
	fstream netflix; //File Pointer
	netflix.open("C:\\Users\\HP\\Downloads\\countries.csv", ios::in);
	if (netflix.fail()) {
		cout << "COULD NOT READ FILE\n";
		exit(1);
	}
	getline(netflix, data);
	int line = 0;
	string value;
	while (!netflix.eof())
	{
		line++;
		getline(netflix, value);
		titles.insertstring(value);
	}
}
int main()
{
	readfile();
	//titles.printhashtable();
	while (1) {
		cout << "\nWELCOME TO NETFLIX IMPLEMENTATION ON HASHTABLES\nBy Mashal
Ashfaque\n\n\t1-Search movie by location (index)\n\t2-Search location of a movie\n\t3-
Show all movies.";
		int choice;

		cin >> choice; getchar();
		while (choice > 3 || choice < 1)
		{
			cout << "Invalid choice. Please select 1, 2 or 3.\n";
			cin >> choice;
		}
		switch (choice)
		{
		case 1: {
			cout << "\nWe have a total of 6235 movies and shows.\nEnter a number
0-6235 and we'll show what movie is over there.\n";
			int s; cin >> s;
			cout <<"\""<< titles.arr[s] << "\" is stored at your entered index."
<< endl;
			break;
		}
```

```
            case 2: { cout << "\nTo find location of a movie, please enter its title. "
<< endl;
                  string m;
                  getline(cin, m);
                  titles.findindex(m);
                  cout << endl;
                  break; }
            case 3:
                  titles.printhashtable();
            }
      }
      }
```

Output:

```
WELCOME TO NETFLIX IMPLEMENTATION ON HASHTABLES
By Mashal Ashfaque

        1-Search movie by location (index)
        2-Search location of a movie
        3-Show all movies.1

We have a total of 6235 movies and shows.
Enter a number 0-6235 and we'll show what movie is over there.
2000
"Inside The Freemasons" is stored at your entered index.
```

Selecting Option 1, and entering location 6212:

```
We have a total of 6235 movies and shows.
Enter a number 0-6235 and we'll show what movie is over there.
6212
"Stranger Things" is stored at your entered index.
```

Selecting Option 2, and entering “Stranger Things”:

```
WELCOME TO NETFLIX IMPLEMENTATION ON HASHTABLES
By Mashal Ashfaque

        1-Search movie by location (index)
        2-Search location of a movie
        3-Show all movies.2

To find location of a movie, please enter its title.
Stranger Things
Stranger Things is at index # 6212
```

Selecting option 3:

```
        2-Search location of a movie
        3-Show all movies.3
ALL MOVIES AND SHOWS:
0       La casa de papel
1       Last Chance U
2       Queer Eye
3       The Epic Tales of Captain Underpants
4       Can't Cope|Won't Cope
5       Intelligence
6       Glee
7       Pinky Malinky
8       Wynonna Earp
9       The Joel McHale Show with Joel McHale
10      H2O: Mermaid Adventures
11      3Below: Tales of Arcadia
12      Bonus Family
13      Nightmare Tenants|Slum Landlords
14      Mossad 101
15      Baby Ballroom
16      Drug Lords
17      Hatchimals |Adventures in Hatchtopia
18      Tayo the Little Bus
19      Zombie Dumb
20      Being Mary Jane: The Series
21      Cheers
```

```
6211    Kakegurui
6212    Stranger Things
6213    Kuromukuro
6214    The Letdown
6215    A Very Secret Service
6216    SWORDGAI The Animation
6217    Yummy Mummies
6218    LEGO Bionicle: The Journey to One
6219    Kath and Kim
6220    El Chapo
6221    Big Dreams|Small Spaces
6222    My First First Love
6223    Orange Is the New Black
6224    Sugar Rush
6225    The Worst Witch
6226    Popples
6227    Beat Bugs
6228    Deep Undercover Collection
6229    Fix It and Finish It
6230    Home: Adventures with Tip & Oh
6231    Luna Petunia: Return to Amazia
6232    Origins Collection
6233    Good Witch
6234    Comedians in Cars Getting Coffee
```

2- Double Hashing:

Code:

```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#define size 6235
using namespace std;

class hashtable {
public:
	string arr[size];
	hashtable()
	{
		for (int i = 0; i < size; i++)
			arr[i] = "";
	}

	int hashFunction(string x) // For string
	{
		int hash =0;
		for (int i = 0; i < x.length(); i++)
		{
			hash += (int(x[i]));
		}
		return abs(hash % size);
	}
	int hash2(string x) // For string
	{
		int hash = 0;
		for (int i = 0; i < x.length(); i++)
		{
			hash += (int(x[i]));
		}
		int a= abs(hash % 6229); //6229 is the last prime n
		return 6229 - a;
	}

	void insertstring(string s)
	{ //done by linear probing
		int index = hashFunction(s);
		for (int i = 0; i < size; i++)
		{
			if (arr[(index + i*hash2(s)) % size] == "")  //to check if cell is
empty or not.
			{
				arr[(index + i * hash2(s)) % size] = s;
				return;
			}
		}
	}
	void findindex(string s)
```

```
	{
		int index=hashFunction(s);
		while (arr[index] != s)
		{
			index=(index+ hash2(s))%size;
		}

		cout << s << " is at index # " << index;
	}
	void printhashtable()
	{
		cout << "ALL MOVIES AND SHOWS: " << endl;
		for (int i = 0; i < size; i++)
		{
			cout<<i<<"\t" << arr[i] << endl;
		}
	}
};
hashtable titles;
void readfile()
{
	string data;
	fstream netflix; //File Pointer
	netflix.open("C:\\Users\\HP\\Downloads\\countries.csv", ios::in);
	if (netflix.fail()) {
		cout << "COULD NOT READ FILE\n";
		exit(1);
	}
	getline(netflix, data);
	int line = 0;
	string value;
	while (!netflix.eof())
	{
		line++;
		getline(netflix, value);
		titles.insertstring(value);
	}
}
int main()
{
	readfile();
	//titles.printhashtable();
	while (1) {
		cout << "\nWELCOME TO NETFLIX IMPLEMENTATION ON HASHTABLES\nBy Mashal
Ashfaque\n\n\t1-Search movie by location (index)\n\t2-Search location of a movie\n\t3-
Show all movies.";
		int choice;

		cin >> choice; getchar();
		while (choice > 3 || choice < 1)
		{
			cout << "Invalid choice. Please select 1, 2 or 3.\n";
			cin >> choice;
		}
		switch (choice)
		{
		case 1: {
```

```
                    cout << "\nWe have a total of 6235 movies and shows.\nEnter a number
0-6235 and we'll show what movie is over there.\n";
                    int s; cin >> s;
                    cout <<"\""<< titles.arr[s] << "\" is stored at your entered index."
<< endl;
                    break;
             }
             case 2: { cout << "\nTo find location of a movie, please enter its title. "
<< endl;
                    string m;
                    getline(cin, m);
                    titles.findindex(m);
                    cout << endl;
                    break; }
             case 3:
                    titles.printhashtable();
             }
      }
}
```

```
WELCOME TO NETFLIX IMPLEMENTATION ON HASHTABLES
By Mashal Ashfaque

        1-Search movie by location (index)
        2-Search location of a movie
        3-Show all movies.2

To find location of a movie, please enter its title.
Stranger Things
Stranger Things is at index # 4223
```

```
We have a total of 6235 movies and shows.
Enter a number 0-6235 and we'll show what movie is over there.
4223
"Stranger Things" is stored at your entered index.
```