Project 3: hashing

CPSC 335 - Algorithm Engineering
Fall 2018
Instructors: Doina Bein (dbein@fullerton.edu), Kevin Wortman (kwortman@fullerton.edu)

Abstract

In this project you will design, implement, and analyze one algorithm for the hashing problem. The algorithm is called Cuckoo Hashing, presented in class. For this problem, you will design and implement one algorithm in C++, test it on various inputs and complete a hash table with a given input. No algorithm analysis is needed for this project.

The Cuckoo Hashing Algorithm

There are several versions of cuckoo hashing. The version we learned in class is the simplest, where there are two hash functions, and thus only two places where any given item could be stored in the table. Let us consider the set of keys to be printable ASCII strings of length no more than 80. Let us consider the hash table size to be 17 .

If key is the string representing the key, then let keysize be the size of the string and key[i] be the ASCII code of the (i+1)th character in the string key read from left to right: $key = key_0\, key_1 \ldots key_{keysize-1}$
Java uses the following hash function on strings:

$$val = 31^{keysize-1} \cdot key_0 + 31^{keysize-2} \cdot key_1 + \ldots + 31^1 \cdot key_{keysize-2} + key_{keysize-1}$$

Let us consider two hash functions, f_1 and f_2. Function f_2 will compute the hash value using Java's hash function formula, while the function f_1 computes a different hash value using a different hash function. Function f_1 computes first a large number then it brings the result into the proper range using the formulas below:

$$val = \sum_{i=0}^{keysize-1} key[i] \cdot 31^i$$

$$f_1 = val \,\%\, tablesize$$

if $f_1 < 0$ then $f_1 = f_1 +$ tablesize

Function f_2 computes first a large number then it brings the result into the proper range using the formulas below:

$$val = \sum_{i=0}^{keysize-1} key[keysize - i - 1] \cdot 31^i$$

$$f_2 = val \,\%\, tablesize$$

if $f_2 < 0$ then $f_2 = f_2$ + tablesize
Both functions f_1 and f_2 compute first a large number then it brings the result into the proper range 0..tablesize-1. But we bring the intermediate results into the proper range after each calculation, we do not need to wait until we compute the final result. Also, we can ring the power term 31^{index} into the proper range before multiplying it with key_{index}

You need to insert the strings below (also given in the input file in6.txt) into the hash table provided next. Please put an empty line at the end of the file.

Algorithm Engineering
California State University
Fullerton
College of Engineering
and Computer Science
Department of Computer
Science
Dynamic Programming
Monge Properties
String Matching
Matrix Searching
Optimal Tree Construction
Online algorithms
emphasis on
Server Problem
Some related problem
Self-Stabilization
One of the greatest
mysteries in science
Quantum Nature of Universe
In physics and
are known
Cuckoo hashing is fun

into the hash table (next page) using f_1 for the first table and f_2 for the second table. Show the result of the insertion in the table shown on next page.

Hint: consider a two-dimensional table of strings t, where t[0] is T1 and t[1] is T2. Consider a variable index that oscillates between 0 and 1as it would have oscillated between T1 and T2. In C++, the value of index could be changed using the tertiary operator: index = index? 0:1. Depending on the value of index, either apply hash function f_1 (index == 0) or f_2 (index == 1).

	Table T1	Table T2
[0]		
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		
[10]		
[11]		
[12]		
[13]		
[14]		
[15]		
[16]		

Obtaining and Submitting Code

This document explains how to obtain and submit your work:

GitHub Education / Tuffix Instructions

Here is the invitation link for this project:

https://classroom.github.com/g/o4D1VDnC

Implementation

You are provided with the following files.

1. `cuckoo.cpp` is is a C++ file that contains the functions and definitions needed for this project. The function definitions are incomplete skeletons; you will need to rewrite them to actually work properly.
2. `in4.txt` is a text file that you can use to match the output available in problem description. You can use this file to see whether your algorithm implementation is working correctly. See sample runs.
3. `in5.txt` is another text file that you can use to match the output available in problem description. You can use this file to see whether your algorithm implementation is working correctly. See sample runs.
4. `in6.txt` is a text file for which you need to complete the table required in problem description.
5. `rubrictest.hpp` is the unit test library used for the test program; you can ignore this file.

`README.md` contains a brief description of the project, and a place to write the names and CSUF email addresses of the group members. You need to modify this file to identify your group members.

What to Do

Decide on who will be in your team, or decide to work alone; have one of your team members accept the GitHub assignment by following the invitation link; have any other team members join your team by following the invitation link; and add your group member names to `README.md`.

Then, implement each of the two algorithms in C++ using the provided skeleton code from `cuckoo.cxx`. Test your code using the provided text files `in4.txt` and `in5.txt`.

Once you are confident that your algorithm implementation is correct, complete the table and submit ***in PDF format***. Submit your PDF by committing it to your GitHub repository along with your code. Your report should include the following:

1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 2.
2. Completed table

Grading Rubric

Your grade will be comprised of three parts: *Form, Function,* and *Analysis.*

Function refers to whether your code works properly by comparing it with the test files `in4.txt`, `in5.txt` and the output of `in6.txt`.

Form refers to the design, organization, and presentation of your code. A grader will read your code and evaluate these aspects of your submission.

Analysis refers to the correctness of your mathematical and empirical analyses, scatter plots, question answers, and the presentation of your report document.

The grading rubric is below.

1. Function = 6 points, scored by the unit test program
2. Form = 9 points, divided as follows:
    a. README.md completed clearly = 3 points
    b. Style (whitespace, variable names, comments, helper functions, etc.) = 3 points
    c. C++ Craftsmanship (appropriate handling of encapsulation, memory management, avoids gross inefficiency and taboo coding practices, etc.) = 3 points
3. Analysis = 20 points, divided as follows
    a. Report document presentation = 3 points
    b. Correct table = 17 points

Legibility standard: As stated on the syllabus, submissions that cannot compile in the Tuffix environment are considered unacceptable and will be assigned an "F" (50%) grade.

Deadline

The project deadline is Friday, November 30, 1 pm.

You will be graded based on what you have pushed to GitHub as of the deadline. Commits made after the deadline will not be considered. Late submissions will not be accepted.

Sample Runs for the Cuckoo Hashing Algorithm:

<u>Example #1:</u>

```
K:\202> ast4
CPSC 335-x – Programming Assignment #4: Cuckoo Hashing algorithm

Input the file name (no spaces)!
in4.txt

String <Algorithm Engineering> will be placed at t[11][0]
String <California> will be placed at t[16][0]
String <State University> will be placed at t[5][0]
String <Fullerton> will be placed at t[15][0]
String <College of Engineering and Computer Science> will be placed at t[10][0]
String <Department of Computer Science> will be placed at t[5][0] replacing <State University>
String <State University> will be placed at t[7][1]
String <Dynamic Programming> will be placed at t[3][0]
String <Monge Properties> will be placed at t[9][0]
String <String Matching> will be placed at t[16][0] replacing <California>
String <California> will be placed at t[2][1]
String <Matrix Searching> will be placed at t[5][0] replacing <Department of Computer Science>
String <Department of Computer Science> will be placed at t[12][1]
String <Optimal Tree Construction> will be placed at t[5][0] replacing <Matrix Searching>
String <Matrix Searching> will be placed at t[11][1]
String <Online algorithms> will be placed at t[0][0]
String <emphasis on> will be placed at t[15][0] replacing <Fullerton>
String <Fullerton> will be placed at t[3][1]
String <Server Problem> will be placed at t[9][0] replacing <Monge Properties>
String <Monge Properties> will be placed at t[2][1] replacing <California>
String <California> will be placed at t[16][0] replacing <String Matching>
String <String Matching> will be placed at t[16][1]
```

<u>Example #2:</u>

```
K:\202> ast4
CPSC 335-x – Programming Assignment #4: Cuckoo Hashing algorithm

Input the file name (no spaces)!
in5.txt

String <Algorithm Engineering> will be placed at t[11][0]
String <California> will be placed at t[16][0]
String <State University> will be placed at t[5][0]
String <Fullerton> will be placed at t[15][0]
String <College of Engineering and Computer Science> will be placed at t[10][0]
String <Department of Computer Science> will be placed at t[5][0] replacing <State University>
String <State University> will be placed at t[7][1]
String <Dynamic Programming> will be placed at t[3][0]
String <Monge Properties> will be placed at t[9][0]
String <String Matching> will be placed at t[16][0] replacing <California>
String <California> will be placed at t[2][1]
String <Matrix Searching> will be placed at t[5][0] replacing <Department of Computer Science>
String <Department of Computer Science> will be placed at t[12][1]
String <Optimal Tree Construction> will be placed at t[5][0] replacing <Matrix Searching>
String <Matrix Searching> will be placed at t[11][1]
String <Online algorithms> will be placed at t[0][0]
String <emphasis on> will be placed at t[15][0] replacing <Fullerton>
String <Fullerton> will be placed at t[3][1]
String <Server Problem> will be placed at t[9][0] replacing <Monge Properties>
String <Monge Properties> will be placed at t[2][1] replacing <California>
String <California> will be placed at t[16][0] replacing <String Matching>
String <String Matching> will be placed at t[16][1]
String <Some related problem> will be placed at t[11][0] replacing <Algorithm Engineering>
String <Algorithm Engineering> will be placed at t[2][1] replacing <Monge Properties>
String <Monge Properties> will be placed at t[9][0] replacing <Server Problem>
String <Server Problem> will be placed at t[4][1]
String <Self-Stabilization> will be placed at t[2][0]
String <One of the greatest> will be placed at t[6][0]
```