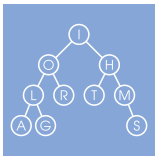

Module: Hash Tables (Week 3 out of 4)
Course: Data Structures (Course 2 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 3: Hash Tables and Hash Functions

Revision: April 4, 2016

Introduction

In this programming assignment, you will practice implementing hash functions and hash tables and using them to solve algorithmic problems. In some cases you will just implement an algorithm from the lectures, while in others you will need to invent an algorithm to solve the given problem using hashing.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. Apply hashing to solve the given algorithmic problems.
2. Implement a simple phone book manager.
3. Implement a hash table using the chaining scheme.
4. Find all occurrences of a pattern in text using Rabin–Karp's algorithm.

Passing Criteria: 2 out of 3

Passing this programming assignment requires passing at least 2 out of 3 code problems from this assignment. In turn, passing a code problem requires implementing a solution that passes all the tests for this problem in the grader and does so under the time and memory limits specified in the problem statement.

Contents

1 Problem: Phone book

Problem Introduction

In this problem you will implement a simple phone book manager.

Problem Description

Task. In this task your goal is to implement a simple phone book manager. It should be able to process the following types of user's queries:

- **add** `number` `name`. It means that the user adds a person with name `name` and phone number `number` to the phone book. If there exists a user with such number already, then your manager has to overwrite the corresponding name.
- **del** `number`. It means that the manager should erase a person with number `number` from the phone book. If there is no such person, then it should just ignore the query.
- **find** `number`. It means that the user looks for a person with phone number `number`. The manager should reply with the appropriate name, or with string "not found" (without quotes) if there is no such person in the book.

Input Format. There is a single integer N in the first line — the number of queries. It's followed by N lines, each of them contains one query in the format described above.

Constraints. $1 \le N \le 10^5$. All phone numbers consist of decimal digits, they don't have leading zeros, and each of them has no more than 7 digits. All names are non-empty strings of latin letters, and each of them has length at most 15. It's guaranteed that there is no person with name "not found".

Output Format. Print the result of each **find** query — the name corresponding to the phone number or "not found" (without quotes) if there is no person in the phone book with such phone number. Output one result per line in the same order as the **find** queries are given in the input.

Time Limits. `C`: 3 sec, `C++`: 3 sec, `Java`: 6 sec, `Python`: 6 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

```
12
add 911 police
add 76213 Mom
add 17239 Bob
find 76213
find 910
find 911
del 910
del 911
find 911
find 76213
add 76213 daddy
find 76213
```

Output:

```
Mom
not found
police
not found
Mom
daddy
```

Explanation:
76213 is Mom's number, 910 is not a number in the phone book, 911 is the number of police, but then it was deleted from the phone book, so the second search for 911 returned "not found". Also, note that when the daddy was added with the same phone number 76213 as Mom's phone number, the contact's name was rewritten, and now search for 76213 returns "daddy" instead of "Mom".

Sample 2.

Input:

```
8
find 3839442
add 123456 me
add 0 granny
find 0
find 123456
del 0
del 0
find 0
```

Output:

```
not found
granny
me
not found
```

Explanation:
Recall that deleting a number that doesn't exist in the phone book doesn't change anything.

Starter Files

The starter solutions for C++, Java and Python3 in this problem read the input, implement a naive algorithm to look up names by phone numbers and write the output. You need to use a fast data structure to implement a better algorithm. If you use other languages, you need to implement the solution from scratch.

What to Do

Use the direct addressing scheme.

2 Problem: Hashing with chains

Problem Introduction

In this problem you will implement a hash table using the chaining scheme.

Problem Description

Task. In this task your goal is to implement a hash table with lists chaining. You are already given the number of buckets m and the hash function. It is a polynomial hash function

$$h(S) = \left(\sum_{i=0}^{|S|-1} S[i]x^i \bmod p \right) \bmod m\,,$$

where $S[i]$ is the ASCII code of the i-th symbol of S, $p = 1\ 000\ 000\ 007$ and $x = 263$. Your program should support the following kinds of queries:

- **add** `string` insert `string` into the table. If there is already such string in the hash table, then just ignore the query.
- **del** `string` remove `string` from the table. If there is no such string in the hash table, then just ignore the query.
- **find** `string` output "yes" or "no" (without quotes) depending on whether the table contains `string` or not.
- **check** i output the content of the i-th list in the table. Use spaces to separate the elements of the list.

When inserting a new string into a hash chain, you must insert it in the beginning of the chain.

Input Format. There is a single integer m in the first line — the number of buckets you should have. The next line contains the number of queries N. It's followed by N lines, each of them contains one query in the format described above.

Constraints. $1 \le N \le 10^5$; $\frac{N}{5} \le m \le N$. All the strings consist of latin letters. Each of them is non-empty and has length at most 15.

Output Format. Print the result of each of the **find** and **check** queries, one result per line, in the same order as these queries are given in the input.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 5 sec, `Python`: 7 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

```
5
12
add world
add HellO
check 4
find World
find world
del world
check 4
del HellO
add luck
add GooD
check 2
del good
```

Output:

```
HellO world
no
yes
HellO
GooD luck
```

Explanation:
The ASCII code of 'w' is 119, for 'o' it is 111, for 'r' it is 114, for 'l' it is 108, and for 'd' it is 100. Thus, $h(\texttt{``world"}) = (119 + 111 \times 263 + 114 \times 263^2 + 108 \times 263^3 + 100 \times 263^4 \bmod 1\,000\,000\,007) \bmod 5 = 4$. It turns out that the hash value of *HellO* is also 4. Recall that we always insert in the beginning of the chain, so after adding "world" and then "HellO" in the same chain index 4, first goes "HellO" and then goes "world". Of course, "World" is not found, and "world" is found, because the strings are case-sensitive, and the codes of 'W' and 'w' are different. After deleting "world", only "HellO" is found in the chain 4. Similarly to "world" and "HellO", after adding "luck" and "GooD" to the same chain 2, first goes "GooD" and then "luck".

Sample 2.

Input:

```
4
8
add test
add test
find test
del test
find test
find Test
add Test
find Test
```

Output:

```
yes
no
no
yes
```

Explanation:
Adding "test" twice is the same as adding "test" once, so first **find** returns "yes". After del, "test" is

no longer in the hash table. First time **find** doesn't find "Test" because it was not added before, and strings are case-sensitive in this problem. Second time "Test" can be found, because it has just been added.

Starter Files

There are starter solutions only for C++, Java and Python3, and if you use other languages, you need to implement solution from scratch. Starter solutions read the input, do a full scan of the whole table to simulate each **find** operation and write the output. This naive simulation algorithm is too slow, so you need to implement the real hash table.

What to Do

Follow the explanations about the chaining scheme from the lectures. Remember to always insert new strings in the beginning of the chain.

3 Problem: Find pattern in text

Problem Introduction

In this problem, your goal is to implement the Rabin–Karp's algorithm.

Problem Description

Task. In this problem your goal is to implement the Rabin–Karp's algorithm for searching the given pattern in the given text.

Input Format. There are two strings in the input: the pattern P and the text T.

Constraints. $1 \leq |P| \leq |T| \leq 5 \cdot 10^5$. The total length of all occurrences of P in T doesn't exceed 10^8. The pattern and the text contain only latin letters.

Output Format. Print all the positions of the occurrences of P in T in the ascending order. Use 0-based indexing of positions in the the text T.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 5 sec, `Python`: 5 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

```
aba
abacaba
```

Output:

```
0 4
```

Explanation:
The pattern *aba* can be found in positions 0 (**aba**caba) and 4 (abac**aba**) of the text *abacaba*.

Sample 2.

Input:

```
Test
testTesttesT
```

Output:

```
4
```

Explanation:
Pattern and text are case-sensitive in this problem. Pattern $Test$ can only be found in position 4 in the text $testTesttesT$.

Sample 3.

Input:

```
aaaaa
baaaaaaa
```

Output:

```
1 2 3
```

Explanation:
Note that the occurrences of the pattern in the text can be overlapping, and that's ok, you still need to output all of them.

Starter Files

The starter solutions in C++, Java and Python3 read the input, apply the naive $O(|T||P|)$ algorithm to this problem and write the output. You need to implement the Rabin–Karp's algorithm instead of the naive algorithm and thus significantly speed up the solution. If you use other languages, you need to implement a solution from scratch.

What to Do

Implement the fast version of the Rabin–Karp's algorithm from the lectures.

4 General Instructions and Recommendations on Solving Algorithmic Problems

Your main goal in an algorithmic problem is to implement a program that solves a given computational problem in just few seconds even on massive datasets. Your program should read a dataset from the standard input and write an answer to the standard output.

Below we provide general instructions and recommendations on solving such problems. Before reading them, go through readings and screencasts in the first module that show a step by step process of solving two algorithmic problems: link.

4.1 Reading the Problem Statement

You start by reading the problem statement that contains the description of a particular computational task as well as time and memory limits your solution should fit in, and one or two sample tests. In some problems your goal is just to implement carefully an algorithm covered in the lectures, while in some other problems you first need to come up with an algorithm yourself.

4.2 Designing an Algorithm

If your goal is to design an algorithm yourself, one of the things it is important to realize is the expected running time of your algorithm. Usually, you can guess it from the problem statement (specifically, from the subsection called constraints) as follows. Modern computers perform roughly 10^8–10^9 operations per second. So, if the maximum size of a dataset in the problem description is $n = 10^5$, then most probably an algorithm with quadratic running time is not going to fit into time limit (since for $n = 10^5$, $n^2 = 10^{10}$) while a solution with running time $O(n \log n)$ will fit. However, an $O(n^2)$ solution will fit if n is up to $10^3 = 1000$, and if n is at most 100, even $O(n^3)$ solutions will fit. In some cases, the problem is so hard that we do not know a polynomial solution. But for n up to 18, a solution with $O(2^n n^2)$ running time will probably fit into the time limit.

To design an algorithm with the expected running time, you will of course need to use the ideas covered in the lectures. Also, make sure to carefully go through sample tests in the problem description.

4.3 Implementing Your Algorithm

When you have an algorithm in mind, you start implementing it. Currently, you can use the following programming languages to implement a solution to a problem: `C`, `C++`, `Java`, `Python2`, `Python3`. For all problems, we will be providing starter solutions for `C++`, `Java`, and `Python3`. If you are going to use one of these programming languages, use these starter files. If you are going to use `C` or `Python2`, you need to implement a solution from scratch.

4.4 Compiling Your Program

For solving programming assignments, you can use any of the following programming languages: `C`, `C++`, `Java`, `Python2`, or `Python3`. However, we will only be providing starter solution files for `C++`, `Java`, and `Python3`. The programming language of your submission is detected automatically, based on the extension of your submission.

Your solution will be compiled as follows. We recommend that when testing your solution locally, you use the same compiler flags for compiling. This will increase the chances that your program behaves in the same way on your machine and on the testing machine (note that a buggy program may behave differently when compiled by different compilers, or even by the same compiler with different flags).

- `C` (`gcc 5.2.1`). File extensions: `.c`. Flags:

```
gcc -pipe -O2 -std=c11
```

- `C++` (`g++ 5.2.1`). File extensions: `.cc`, `.cpp`. Flags:

```
g++ -pipe -O2 -std=c++11
```

  If your `C/C++` compiler does not recognize `-std=c++11` flag, try replacing it with `-std=c++0x` flag or compiling without this flag at all (all starter solutions can be compiled without it). On Linux and MacOS, you most probably have the required compiler. On Windows, you may use your favorite compiler or install, e.g., `cygwin`.

- `Java (Open JDK 8)`. File extensions: `.java`. Flags:

```
javac -encoding UTF-8
```

- `Python 2 (CPython 2.7)`. File extensions: `.py2` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python2"). No flags:

```
python2
```

- `Python 3 (CPython 3.4)`. File extensions: `.py3` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python3"). No flags:

```
python3
```

4.5 Testing Your Program

When your program is ready, you start testing it. It makes sense to start with small datasets — for example, sample tests provided in the problem description. Ensure that your program produces a correct result.

You then proceed to checking how long does it take your program to process a massive dataset. For this, it makes sense to implement your algorithm as a function like `solve(dataset)` and then implement an additional procedure `generate()` that produces a large dataset. For example, if an input to a problem is a sequence of integers of length $1 \le n \le 10^5$, then generate a sequence of length exactly 10^5, pass it to your `solve()` function, and ensure that the program outputs the result quickly.

Also, check the boundary values. Ensure that your program processes correctly sequences of size $n = 1, 2, 10^5$. If a sequence of integers from 0 to, say, 10^6 is given as an input, check how your program behaves when it is given a sequence $0, 0, \ldots, 0$ or a sequence $10^6, 10^6, \ldots, 10^6$. Check also on randomly generated data. For each such test check that you program produces a correct result (or at least a reasonably looking result).

In the end, we encourage you to stress test your program to make sure it passes in the system at the first attempt. See the readings and screencasts from the first week to learn about testing and stress testing: link.

4.6 Submitting Your Program to the Grading System

When you are done with testing, you submit your program to the grading system. For this, you go the submission page, create a new submission, and upload a file with your program. The grading system then compiles your program (detecting the programming language based on your file extension, see Subsection 4.4) and runs it on a set of carefully constructed tests to check that your program always outputs a correct result and that it always fits into the given time and memory limits. The grading usually takes no more than a minute, but in rare cases when the servers are overloaded it might take longer. Please be patient. You can safely leave the page when your solution is uploaded.

As a result, you get a feedback message from the grading system. The feedback message that you will love to see is: `Good job!` This means that your program has passed all the tests. On the other hand, the three messages `Wrong answer`, `Time limit exceeded`, `Memory limit exceeded` notify you that your program failed due to one these three reasons. Note that the grader will not show you the actual test you program have failed on (though it does show you the test if your program have failed on one of the first few tests; this is done to help you to get the input/output format right).

4.7 Debugging and Stress Testing Your Program

If your program failed, you will need to debug it. Most probably, you didn't follow some of our suggestions from the section 4.5. See the readings and screencasts from the first week to learn about debugging your program: link.

You are almost guaranteed to find a bug in your program using stress testing, because the way these programming assignments and tests for them are prepared follows the same process: small manual tests, tests for edge cases, tests for large numbers and integer overflow, big tests for time limit and memory limit checking, random test generation. Also, implementation of wrong solutions which we expect to see and stress testing against them to add tests specifically against those wrong solutions.

Go ahead, and we hope you pass the assignment soon!

5 Frequently Asked Questions

5.1 I submit the program, but nothing happens. Why?

You need to create submission and upload the file with your solution in one of the programming languages `C`, `C++`, `Java`, or `Python` (see Subsections 4.3 and 4.4). Make sure that after uploading the file with your solution you press on the blue "Submit" button in the bottom. After that, the grading starts, and the submission being graded is enclosed in an orange rectangle. After the testing is finished, the rectangle disappears, and the results of the testing of all problems is shown to you.

5.2 I submit the solution only for one problem, but all the problems in the assignment are graded. Why?

Each time you submit any solution, the last uploaded solution for each problem is tested. Don't worry: this doesn't affect your score even if the submissions for the other problems are wrong. As soon as you pass the sufficient number of problems in the assignment (see in the pdf with instructions), you pass the assignment. After that, you can improve your result if you successfully pass more problems from the assignment. We recommend working on one problem at a time, checking whether your solution for any given problem passes in the system as soon as you are confident in it. However, it is better to test it first, please refer to the reading about stress testing: link.

5.3 What are the possible grading outcomes, and how to read them?

Your solution may either pass or not. To pass, it must work without crashing and return the correct answers on all the test cases we prepared for you, and do so under the time limit and memory limit constraints specified in the problem statement. If your solution passes, you get the corresponding feedback "Good job!" and get a point for the problem. If your solution fails, it can be because it crashes, returns wrong answer, works for too long or uses too much memory for some test case. The feedback will contain the number of the test case on which your solution fails and the total number of test cases in the system. The tests for the problem are numbered from 1 to the total number of test cases for the problem, and the program is always tested on all the tests in the order from the test number 1 to the test with the biggest number.

Here are the possible outcomes:

`Good job!` Hurrah! Your solution passed, and you get a point!

`Wrong answer.` Your solution has output incorrect answer for some test case. If it is a sample test case from the problem statement, or if you are solving Programming Assignment 1, you will also see the input data, the output of your program and the correct answer. Otherwise, you won't know the input, the output, and the correct answer. Check that you consider all the cases correctly, avoid integer overflow, output the required white space, output the floating point numbers with the required precision, don't output anything in addition to what you are asked to output in the output specification of the problem statement. See this reading on testing: link.

`Time limit exceeded.` Your solution worked longer than the allowed time limit for some test case. If it is a sample test case from the problem statement, or if you are solving Programming Assignment 1, you will also see the input data, the output of your program and the correct answer. Otherwise, you won't know the input, the output and the correct answer. Check again that your algorithm has good enough running time estimate. Test your program locally on the test of maximum size allowed by the problem statement and see how long it works. Check that your program doesn't wait for some input from the user which makes it to wait forever. See this reading on testing: link.

`Memory limit exceeded.` Your solution used more than the allowed memory limit for some test case. If it is a sample test case from the problem statement, or if you are solving Programming Assignment 1,

you will also see the input data, the output of your program and the correct answer. Otherwise, you won't know the input, the output and the correct answer. Estimate the amount of memory that your program is going to use in the worst case and check that it is less than the memory limit. Check that you don't create too large arrays or data structures. Check that you don't create large arrays or lists or vectors consisting of empty arrays or empty strings, since those in some cases still eat up memory. Test your program locally on the test of maximum size allowed by the problem statement and look at its memory consumption in the system.

`Cannot check answer. Perhaps output format is wrong.` This happens when you output something completely different than expected. For example, you are required to output word "Yes" or "No", but you output number 1 or 0, or vice versa. Or your program has empty output. Or your program outputs not only the correct answer, but also some additional information (this is not allowed, so please follow exactly the output format specified in the problem statement). Maybe your program doesn't output anything, because it crashes.

`Unknown signal 6 (or 7, or 8, or 11, or some other).` This happens when your program crashes. It can be because of division by zero, accessing memory outside of the array bounds, using uninitialized variables, too deep recursion that triggers stack overflow, sorting with contradictory comparator, removing elements from an empty data structure, trying to allocate too much memory, and many other reasons. Look at your code and think about all those possibilities. Make sure that you use the same compilers and the same compiler options as we do. Try different testing techniques from this reading: link.

`Internal error: exception...` Most probably, you submitted a compiled program instead of a source code.

`Grading failed.` Something very wrong happened with the system. Contact Coursera for help or write in the forums to let us know.

5.4 How to understand why my program fails and to fix it?

If your program works incorrectly, it gets a feedback from the grader. For the Programming Assignment 1, when your solution fails, you will see the input data, the correct answer and the output of your program in case it didn't crash, finished under the time limit and memory limit constraints. If the program crashed, worked too long or used too much memory, the system stops it, so you won't see the output of your program or will see just part of the whole output. We show you all this information so that you get used to the algorithmic problems in general and get some experience debugging your programs while knowing exactly on which tests they fail.

However, in the following Programming Assignments throughout the Specialization you will only get so much information for the test cases from the problem statement. For the next tests you will only get the result: passed, time limit exceeded, memory limit exceeded, wrong answer, wrong output format or some form of crash. We hide the test cases, because it is crucial for you to learn to test and fix your program even without knowing exactly the test on which it fails. In the real life, often there will be no or only partial information about the failure of your program or service. You will need to find the failing test case yourself. Stress testing is one powerful technique that allows you to do that. You should apply it after using the other testing techniques covered in this reading.

5.5 Why do you hide the test on which my program fails?

Often beginner programmers think by default that their programs work. Experienced programmers know, however, that their programs almost never work initially. Everyone who wants to become a better programmer needs to go through this realization.

When you are sure that your program works by default, you just throw a few random test cases against it, and if the answers look reasonable, you consider your work done. However, mostly this is not enough. To make one's programs work, one must test them really well. Sometimes, the programs still don't work although you tried really hard to test them, and you need to be both skilled and creative to fix your bugs. Solutions to algorithmic problems are one of the hardest to implement correctly. That's why in this Specialization you will gain this important experience which will be invaluable in the future when you write programs which you really need to get right.

It is crucial for you to learn to test and fix your programs yourself. In the real life, often there will be no or only partial information about the failure of your program or service. Still, you will have to reproduce the failure to fix it (or just guess what it is, but that's rare, and you will still need to reproduce the failure to make sure you have really fixed it). When you solve algorithmic problems, it is very frequent to make subtle mistakes. That's why you should apply the testing techniques described in this reading to find the failing test case and fix your program.

5.6 My solution does not pass the tests? May I post it in the forum and ask for a help?

No, please do not post any solutions in the forum or anywhere on the web, even if a solution does not pass the tests (as in this case you are still revealing parts of a correct solution). Recall the third item of the Coursera Honor Code: "I will not make solutions to homework, quizzes, exams, projects, and other assignments available to anyone else (except to the extent an assignment explicitly permits sharing solutions). This includes both solutions written by me, as well as any solutions provided by the course staff or others" (link).

5.7 Are you going to support my favorite language in programming assignments?

Currently, we are going to support `C`, `C++`, `Java`, and `Python` only, but we may add other programming languages later if there appears a huge need. To express your interest in a particular programming language, please post its name in this thread or upvote the corresponding option if it is already there.

5.8 My implementation always fails in the grader, though I already tested and stress tested it a lot. Would not it be better if you give me a solution to this problem or at least the test cases that you use? I will then be able to fix my code and will learn how to avoid making mistakes. Otherwise, I do not feel that I learn anything from solving this problem. I am just stuck.

First of all, it is just not true that you do not learn by trying to fix your implementation.

The process of trying to invent new test cases that might fail your program and proving them wrong is often enlightening. This thinking about the invariants which you expect your loops, ifs, etc. to keep and proving them wrong (or right) makes you understand what happens inside your program and in the general algorithm you're studying much more.

Also, it is important to be able to find a bug in your implementation without knowing a test case and without having a reference solution. Assume that you designed an application and an annoyed user reports that it crashed. Most probably, the user will not tell you the exact sequence of operations that led to a crash. Moreover, there will be no reference application. Hence, once again, it is important to be able to locate a bug in your implementation yourself, without a magic oracle giving you either a test case that your program fails or a reference solution. We encourage you to use programming assignments in this class as a way of practicing this important skill.

If you have already tested a lot (considered all corner cases that you can imagine, constructed a set of manual test cases, applied stress testing), but your program still fails and you are stuck, try to ask for help

on the forum. We encourage you to do this by first explaining what kind of corner cases you have already considered (it may happen that when writing such a post you will realize that you missed some corner cases!) and only then asking other learners to give you more ideas for tests cases.