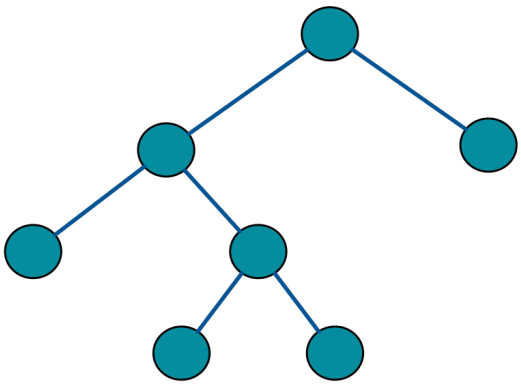
Lab 2



Group 26
Klas Mannberg klaman-8@student.ltu.se
D0012E Algorithms and data structures

LULEÅ UNIVERSITY OF TECHNOLOGY

December 9, 2019

Abstract

We analyze a modified version of probing a hash table. The probe method and the testing functions used for analysis are then developed using Python.

Contents

1 Introduction

2 Theory

2.1 Linear probing

Function used = $h(x,k) = (f(x) + i) mod n$ Time complexity = $\theta(1)$
Successful search : $1/2 * (1 + 1/(1-a))$ probes
Unsuccessful search : $1/2 * (1 + 1/(1-a)^2)$ probes
per search assuming each slot is equally likely to accessed. For example if the list is 90%full we would get $1/2 * (1 + 1/(1-a) = 1/2 * (1 + 1/0.1) = 5.5$ probes at most. a is load factor and < 1 since $\frac{elements}{totalsize} < 1$.

2.2 Modified probing

Function used ldown = $h(x,k) = (f(x) + i) mod n$ if $ldown <= lup$ else lup = $h(x,k) = (f(x) - i) mod n$ is used. Both with a time complexity of $\theta(1)$
I suspect by the nature of the normal linear probing stacking up when a lot of collisions happen that this method would result in less total probes per search when we allow the probing to switch direction efficiently.

3 Implementation

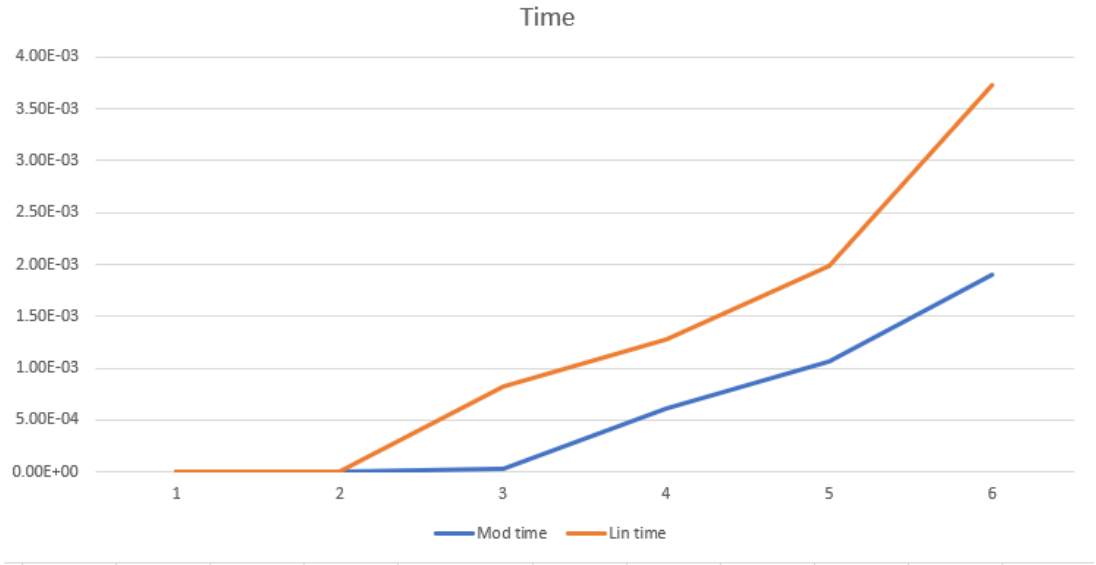
The implementation of the algorithms were created in Python 3. The source code for all algorithms are included in the file lab2.py

3.1 Experimentation

Array length n is 10 000. All values to be inserted are randomly generated with a range of 1-1000.
Test data results:

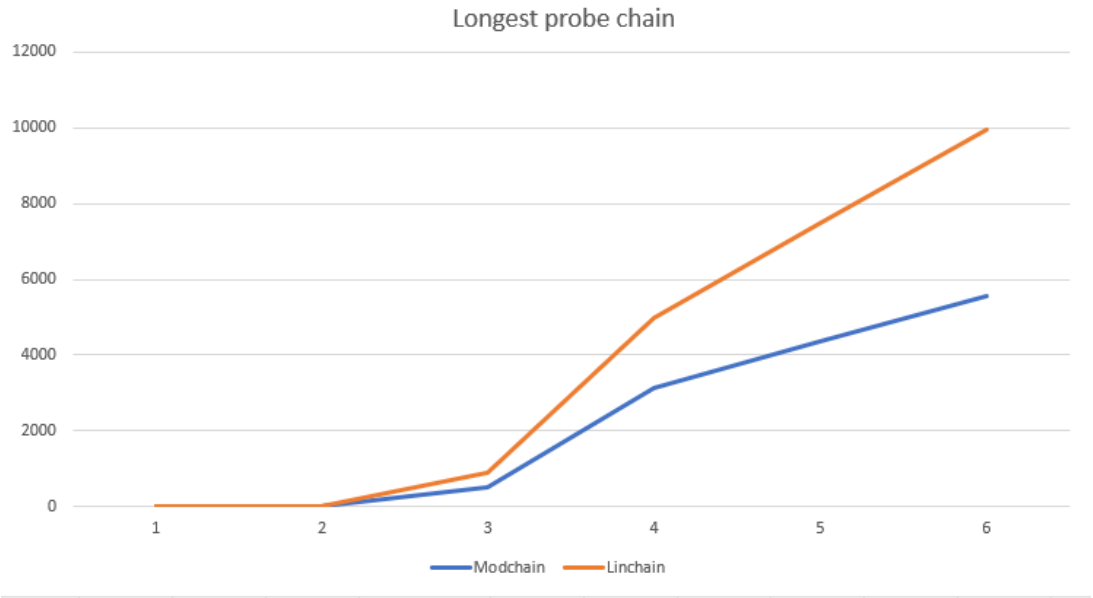
		Array length 10 000. Varying number of values inserted. Range 1-1000 per value					
		MOD	LIN	MOD	LIN	Longest probe chain	
# Values INS	Load factc	Mod time	Lin time	Mod Prob	Lin Probes	Modchain	Linchain
10	0.001	1.30E-06	1.40E-06	0	0	0	0
100	0.01	1.30E-06	1.40E-06	3	3	1	1
1000	0.1	2.41E-05	0.000817	12096	15591	523	889
5000	0.5	0.000605	0.001276	6012439	10003523	3110	4990
7500	0.75	1.06E-03	0.001992	13836801	24328907	4350	7470
10000	1	0.001901	0.003724	24781627	44997209	5542	9943

Graph 1 Time:
The total running time of our modified method (blue) versus the running time of the default linear probing (orange).
The time is our y axis and the x axis is the test number ranging from {1,2,3,4,5,6} = Load factor {0.001,0.01,0.1,0.5,0.75,1}



Graph 2 Probes:
The total amount of probes of our modified method (blue) versus the total amount of probes of the default linear probing (orange). The probe count is our y axis and the x axis is the test number ranging from {1,2,3,4,5,6} = Load factor {0.001,0.01,0.1,0.5,0.75,1}

Graph 3 Longest probe chain:
The longest probe chain of our modified method (blue) versus the longest probe chain of the default linear probing (orange). The longest probe chain is our y axis and the x axis is the test number ranging from {1,2,3,4,5,6} = Load factor {0.001,0.01,0.1,0.5,0.75,1}



4 Result

The modified version seems to get exponentially faster once we start getting more and more collisions. The resulting amount of probes, running time and longest probe chain is almost half of the linear probing when we reach higher load factors.