LECTURE-15

HT Advanced Topics and Priority Queue Introduction

Hash Tables Advanced and Priority Queue Introduction

For Poll Ev

CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan
Department of Computer Science, SBASSE

Agenda

- Double Hashing
- Chained and Open Addressing HashTables performance analysis
- Comparision in performance of AVL and HashTables

RECAP: Deletion OA HT --- Issue resolved

Keep a flag “deleted” in the struct (which takes the value 0 or 1)

– For insert, treat “deleted=1” as “none” and do the insertion

– For search, treat “deleted=1” as “continue looking head”

How can we address clustering?

- Reduce rate of collisions

    1. Associate a good hash function

    2. Monitor the load factor: take action.

    3. Double hashing!

Double Hashing

- Uses a primary **h(key)** and secondary function **d(key)** to handle collisions by placing an item in the first available cell of the series:

$$\textbf{Index} = \{ [h(key) + j*d(key)] \bmod m \} \text{ for } j = 0, 1, \ldots, m-1$$

  - The secondary hash function d(key) is not allowed to have zero values
  - q and m must be coprime.

- Secondary hash function is:

$$d(key) = q - [hc(key) \bmod q]$$

where, $q < m$, and q is a prime and the possible values for $d_2(k)$ are 1, 2, ..., q

Double Hashing in Action!

- $h(k) = k$, $m = 13$ $q = 7$
- $d(k) = 7 - (k \bmod 7)$
- Index = [h(k) + (j * d(k))] mod 13
- **Insert keys 18, 41, 22, 44, 59, 32, 31, and 73 (left to right)**

0	1	2	3	4	5	6	7	8	9	10	11	12
		41			18				22			

How many probes for 44?

Index = [h(44) + (j * d(44))] mod 13

h(44) = 44 **d(44) = 5**

- **For j = 0 index (44 + 0 %13) comes out to be 5**
- **For j = 1 index ((44 + 5) %13) comes out to be 10**

0	1	2	3	4	5	6	7	8	9	10	11	12
		41			18	32	59		22	44		

Double Hashing in Action!

- $h(k) = k$, m = 13 q = 7
- $d(k) = 7 - (k \bmod 7)$
- Index = [h(k) + (j * d(k))] mod 13
- Insert keys 18, 41, 22, 44, 59, 32, 31, and 73 (left to right)

0	1	2	3	4	5	6	7	8	9	10	11	12
		41			18	32	59		22	44		

How many probes for 31?

Index = [h(31) + (j * d(31))] mod 13

h(31) = 31 **d(31) = 4**

- **For j = 0 index (31 + 0 %13) comes out to be 5**
- **For j = 1 index ((31 + 4) %13) comes out to be 9**
- **For j = 2 index (31 + [4 +4] %13) comes out to be 0**

0	1	2	3	4	5	6	7	8	9	10	11	12
31		41			18	32	59		22	44		

Recall: Load Factor ($\boldsymbol{\alpha}$) = n/m.

- α can never be greater than 1. Why?

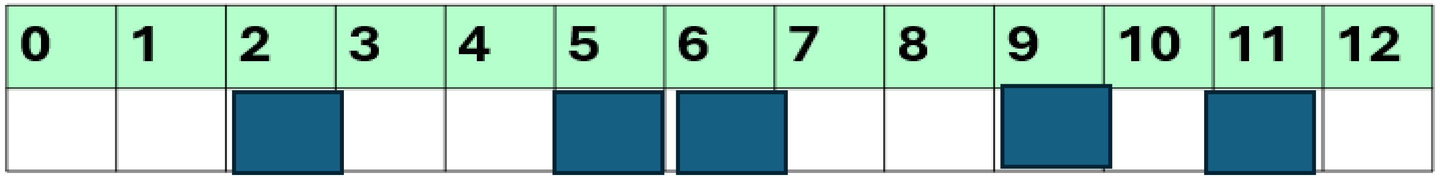
- if α is small, less slots of the array have to be probed to search an element in an Open Addressing HT. Why?

- if α is large, in a Chained HT chains are longer because of more collisions and search time is increased. Why?

Performance Analysis of Hashing

If Hash Function is independent!

- Suppose we want to **insert** an item with key **k** and there are already **n** items in a table of size **m**



- What is the probability of finding a free slot without linear probing? **8/13**

- What is the probability of finding a free slot on first probe?

[1 – (8/13)] *[8/13]

Performance Analysis of Hashing

How do we analyse the performance for an average number of probes into a OA HT with a certain α.

We need the expected average of number of probes!

Expected Value: Average of all possible outcomes, weighted by their probability of occurrence.

$$\mathbb{E}[X] = \sum_{k=1}^{\infty} k(1-p)^{k-1} p = \frac{1}{p}$$

From the Geometric Series
http://bit.ly/47yRssU
Link to Khan Academy calculating sum of geometric series

Numbers to our analysis

What is the Expected Value or average number of probes if $\boldsymbol{\alpha} = \frac{\mathbf{50}}{\mathbf{100}}$ when E(x) = 1/p = 1/(1- $\boldsymbol{\alpha}$)?

E(x) = 1/ (50/100) = 100/50 = 2 probes

What is the Expected Value or average number of probes if $\boldsymbol{\alpha} = \frac{\mathbf{90}}{\mathbf{100}}$ when E(x) = 1/p = 1/(1- $\boldsymbol{\alpha}$)?

E(x) = 1/ (10/100) = 100/10 = 10 probes

Which is performing better?

Chaining vs Open Addressing in HTs

- **Chaining** Pros:
  - **NO CLUSTERING.** Chains of other keys cannot mix to increase length of chain.

    A higher load factor is not as adverse to Chained HT as it is to OA HT!

- **Open Addressing** Pros:
  - Array is **BETTER CACHE LOCALITY.**

    A Chained Hashtable has all elements scattered in memory like all linked structures

Comparing AVLs and Hash Tables

- What is the relationship of a data item to its placement in these data structures?

- AVL allows us to exploit order of data items!

- HT arbitrarily assigns data to index dictated by the associated HT!