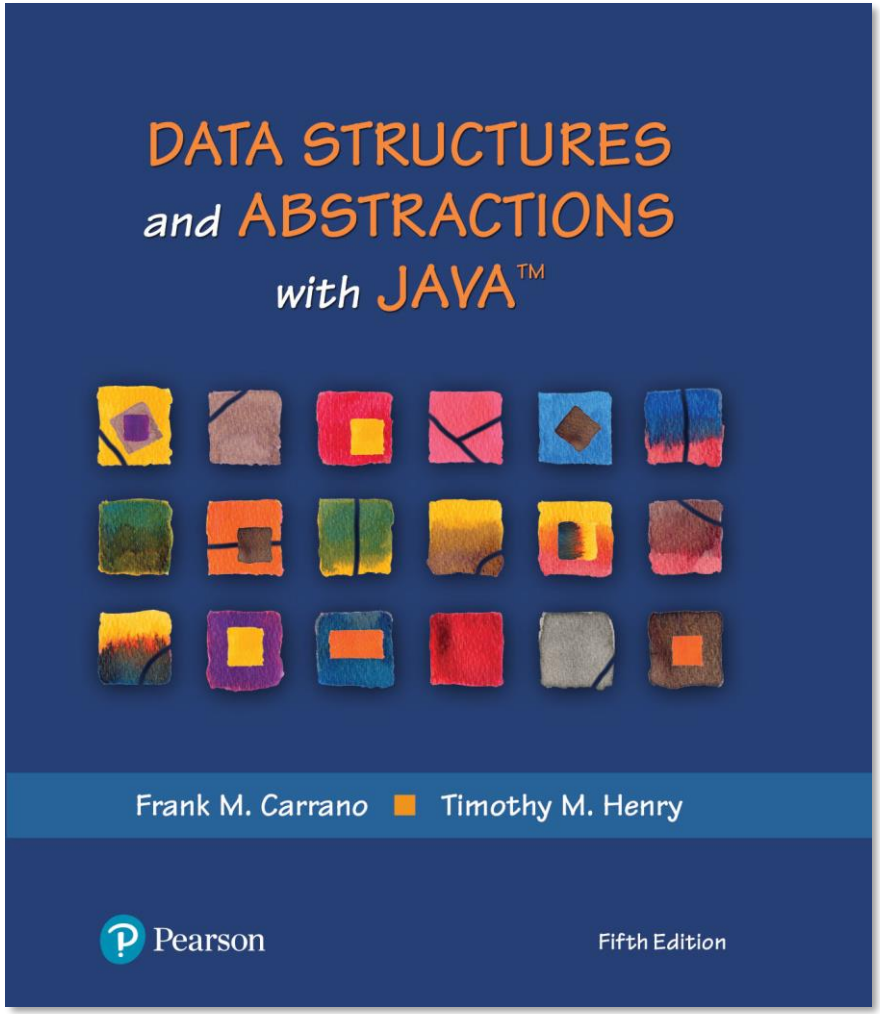
Data Structures and Abstractions with Java™

5th Edition



Chapter 23

Hashing as a Dictionary Implementation

Efficiency of Hashing

- Observations about the time efficiency of these operations
  - Successful retrieval/removal has same efficiency as successful search
  - Unsuccessful retrieval/removal has same efficiency as unsuccessful search
  - Successful addition has same efficiency as unsuccessful search
  - Unsuccessful addition has same efficiency as successful search

Load Factor

- Definition of load factor:

$$\lambda = \frac{Number\ of\ entries\ in\ the\ dictionary}{Number\ of\ locations\ in\ the\ hash\ table}$$

  - Never negative
  - For open addressing, $1 \geq \lambda$
  - For separate chaining, λ has no maximum value
  - Restricting size of λ improves performance

Cost of Open Addressing

- Average number of searches for linear probing

For unsuccessful search: $$\frac{1}{2}\left\{1+\frac{1}{(1-\lambda)^2}\right\}$$

For successful search: $$\frac{1}{2}\left\{1+\frac{1}{(1-\lambda)}\right\}$$

Cost of Open Addressing

λ	Unsuccessful Search	Successful Search
0.1	1.1	1.1
0.3	1.5	1.2
0.5	2.5	1.5
0.7	6.1	2.2
0.9	50.5	5.5

FIGURE 23-1 The average number of comparisons required by a search of the hash table for given values of the load factor λ when using linear probing

Quadratic Probing and Double Hashing

- Average number of comparisons needed

For unsuccessful search: $\frac{1}{(1-\lambda)}$

For successful search: $\frac{1}{\lambda}\log\left(\frac{1}{1-\lambda}\right)$

Quadratic Probing and Double Hashing

λ	Unsuccessful Search	Successful Search
0.1	1.1	1.1
0.3	1.5	1.2
0.5	2.0	1.4
0.7	3.3	1.7
0.9	10.0	2.6

FIGURE 23-2 The average number of comparisons required by a search of the hash table for given values of the load factor λ when using either quadratic probing or double hashing

Cost of Separate Chaining

- Average number of comparisons during a search when separate chaining is used

For unsuccessful search: λ

For successful search: $1 + \lambda/2$

To maintain reasonable efficiency, you should keep $\lambda < 1$.

Cost of Separate Chaining

λ	Unsuccessful Search	Successful Search
0.1	0.1	1.1
0.3	0.3	1.2
0.5	0.5	1.3
0.7	0.7	1.4
0.9	0.9	1.5
1.1	1.1	1.6
1.3	1.3	1.7
1.5	1.5	1.8
1.7	1.7	1.9
1.9	1.9	2.0
2.0	2.0	2.0

FIGURE 23-3 The average number of comparisons required by a search of the hash table for given values of the load factor λ when using separate chaining

Maintaining the Performance of Hashing

- To maintain efficiency, restrict the size of λ as follows:
  - $\lambda < 0.5$ for open addressing
  - $\lambda < 1.0$ for separate chaining
- Should the load factor exceed these bounds
  - Increase the size of the hash table

Rehashing

- When the load factor λ becomes too large must resize the hash table
- Compute the table's new size
  - Double its present size
  - Increase the result to the next prime number
  - Use method add to add the current entries in dictionary to new hash table

Comparing Schemes for Collision Resolution

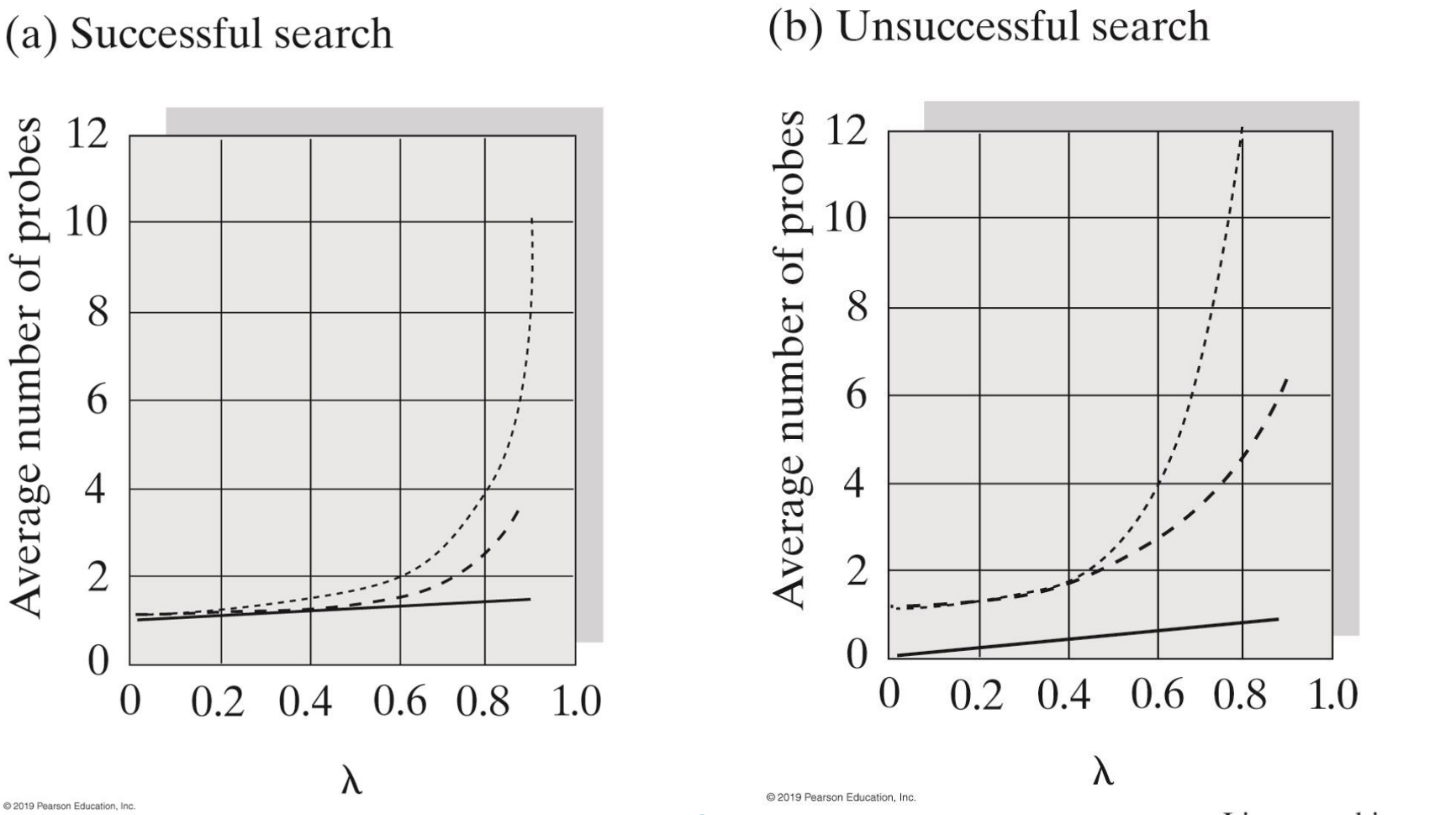


FIGURE 23-4 The average number of comparisons required by a search of the hash table versus the load factor λ for four collision resolution techniques

Java Class Library: The Class `HashMap`

- Hash table is a collection of buckets
  - Each bucket contains several entries
- Variety of constructors provided
- Default maximum load factor of 0.75
  - When limit exceeded, size of table increased by rehashing
- Possible to avoid rehashing by setting number of buckets initially larger

Java Class Library: The Class `HashSet`

- Implements the interface **`java.util.Set`**
- **`HashSet`** uses an instance of the class **`HashMap`**
- Variety of constructors provided in class

End

Chapter 23