University of Pittsburgh

Algorithms and Data Structures 1
CS 0445

Fall 2022

Sherif Khattab

ksm73@pitt.edu

(Slides adapted from slides of Textbook, Dr. Ramirez's CS 0445, and Dr. Farnan's CS 1501)

Announcements

- Upcoming Deadlines
  - Homework 11: This Friday 12/9 @ 11:59 pm
  - Lab 11: next Monday 12/12
  - Lab 12 and Homework 12: Monday 12/19
  - Assignment 5 is now for extra credit ONLY
  - We have 4 programming assignments
    - the lowest is dropped
    - each worth 13.3%
  - Assignment 3: Friday 12/16 @ 11:59 pm
  - Assignment 4: Friday 12/16 @ 11:59 pm

Bonus Opportunities

- Bonus Lab due on 12/19
- Bonus Homework due on 12/19
- Bonus Assignment due on 12/19
- 1 bonus point for entire class when OMETs response rate >= 80%
  - Currently at 23%
  - Deadline is Sunday 12/11

Final Exam

- Same format as midterm
- Non-cumulative
- Date, time and location on PeopleSoft
  - Thursday 12/15 8-9:50 am (coffee served!)
- Same classroom as lectures
- Study guide and practice test to be posted soon

Previous Lecture …

- Hashing!
  - what makes a good hash function
    - Horner's method + modular hashing
  - Handling collisions
    - Open addressing
      - Linear probing

This Lecture …

- Hashing!
  - Handling collisions
    - Open addressing
      - Double hashing
    - Closed addressing
- String matching

Muddiest Points

- **Q: why do we have iterable interface and iterator interface. As only iterator works here**
- Iterator interface is used to implement iterators
- Iterable interface is used to implement containers that have iterators
  - allows us to use the for-each loop structure

```
IterableLinkedList<Integer> list = new …..
for(Integer x : list){
   //do something with x
}
```

Muddiest Points

- **Q: Can we please get more in class tophat questions? It would be a very helpful way to boost our grades.**
- Sure. Let's have a couple today and next lecture!

Double hashing

- After a collision, instead of attempting to place the key x in i+1 mod m, look at i+h2(x) mod m
  - h2() is a second, different hash function
    - Should still follow the same general rules as h() to be considered good, but needs to be different from h()
      - h(x) == h(y) AND h2(x) == h2(y) should be very unlikely
        - Hence, it should be unlikely for two keys to use the same increment

Double hashing

- h(x) = x mod 11
- h2(x) = (x mod 7) +1
- Insert 14, 17, 25, 37, 34, 16, 26

0	1	2	3	4	5	6	7	8	9	10
	34		14	37	16	17		25		26

- Why could we not use h2(x) = x mod 7?
  - Try to insert 2401

A few extra rules for h2()

- Second hash function cannot map a value to 0
- You should try all indices once before trying one twice

- Were either of these issues for linear probing?

As $\alpha \rightarrow 1$...

- Meaning n approaches m...
- Both linear probing and double hashing degrade to O(n)
  - How?
    - Multiple collisions will occur in both schemes
    - Consider inserts and misses...
      - Both continue until an empty index is found
        - With few indices available, close to m probes will need to be performed
          - O(m)
        - n is approaching m, so this turns out to be O(n)

Horner's method

```
public long horners_hash(String key, int n) {
        long h = 0;
        for (int j = 0; j < n; j++)
                h = (R * h + key.charAt(j)) % m;
        return h;
}
```

horners_hash("abcd", 4) =

- 'a' * R^3 + 'b' * R^2 + 'c' * R + 'd' % m
- **h = 'a' % m**
- **h = (h * R + 'b') % m**
- = (('a' % m) * R + 'b') % m
- **h = (h * R + 'c') % m**
- = ((('a' % m) * R + 'b' % m) * R + 'c') % m
- **h = (h * R + 'd') % m**
- = ((('a' % m) * R + 'b' % m) * R + 'c' % m) * R + 'd') % m

Open addressing issues

- Must keep a portion of the table empty to maintain respectable performance
  - For linear hashing ½ is a good rule of thumb
    - Can go higher with double hashing
- What do we do when the hash table is more than half full?
  - resizing!
  - How?

String Matching

- Have a pattern string p of length m
- Have a text string t of length n
- Can we find an index i of string t such that each of the m characters in the substring of t starting at i matches each character in p
  - Example: can we find the pattern "fox" in the text "the quick brown fox jumps over the lazy dog"?
    - Yes! At index 16 of the text string!

Simple approach

- BRUTE FORCE
  - Start at the beginning of both pattern and text
  - Compare characters left to right
  - Mismatch?
  - Start again at the 2nd character of the text and the beginning of the pattern...

Brute force code

```
public static int bf_search(String pat, String txt) {
    int m = pat.length();
    int n = txt.length();
    for (int i = 0; i <= n - m; i++) {
        int j;
        for (j = 0; j < m; j++) {
            if (txt.charAt(i + j) != pat.charAt(j))
                break;
        }
        if (j == m)
            return i; // found at offset i
    }
    return n; // not found
}
```