CS 124: Course Review for Final Exam

- Grades So Far (with Top Students)
- Exam Format: Similar to Midterms
- Class Quizzes
- Some Topic Highlights

Illustrations of $O$, $\Omega$, $\theta$ (Class Quiz 1)

True or false?
1. $g_1(N) = O(f_1(N))$
2. $g_2(N) = O(f_1(N))$
3. $g_3(N) = O(f_1(N))$
4. $g_1(N) = \Omega(f_2(N))$
5. $g_2(N) = \Omega(f_2(N))$
6. $g_3(N) = \Omega(f_2(N))$

True in all cases.

Class Quiz 1 Outcome
- 100%: 22 (out of 58)
Arithmetic expression tree

Class Quiz 2:
Traverse this tree into (a) prefix, (b) infix, and (c) postfix expressions.

What if the access path length is an odd number?

- One rotation must be a single rotation.
- Do it either at the bottom or at the top.
  - The resulting tree structures may be different. Both are fine!

Class Quiz 3:
Let’s do it at the bottom in this class (with T,S,A,M,O,N).

Class Quiz 2 Outcome
• 100%: 37 (out of 57)

Class Quiz 3 Outcome
• 100%: 12 (out of 51)
Double hashing: example

After inserting the first 10 items
\( k = 11, 43, 16, 10, 21, 18, 26, 35, 19, 27 \).

Inserting the next 3 items: \( k = 15, 36, 57 \), using
\[
    h_i(k) = (h(k) + i \cdot (k \% 11 + 1) \% 20)
\]

Class Quiz 4 Outcome
• 100%: 34 (out of 48)

Shellsort: Class Quiz 5

• Sort the file ILOVERECURSION using the increment sequence 4,1.

• Total #comps:
• Total #swaps:

• Phase 1 (h=4): ELECINOORSVUR (14;7)
• Phase 2 (h=1): CEEILNOORRSUV (24;12)
• Answers: 38; 19

Class Quiz 5 Outcome
• 100%: 17 (out of 41)
Quicksort: Class Quiz 6

- Sort the file ILOVERECURSION.
  - Choose the median-of-three as the pivot
    (and show it in each recursion).
  - I L O V E R E C U R S I O N
    - pivot: {I,N,E} -> I
  - I C E E I R O L U R S N O V
  - I C E E I R O L U R S N O V
    - I C E E pivot: {I,E,C} -> E
    - R O L U R S N O V pivot: {R, V, R} -> R
  - C E E I I L N O O R R S U V

Class Quiz 6 Outcome
- 100%: 13 (out of 44)

Class Quiz 7: BFS and DFS

- Breadth-first search and depth-first search starting with vertex 2: List the orders of vertices to be visited on (a).
  (Hint: when there is more than one adjacent node to choose, pick the smallest numbered node.)
  - BSF: {2,1,3,4,5,7,8,6}
  - DFS: {2,1,3,5,4,6,7,8}

Class Quiz 7 Outcome
- 100%: 24 (out of 45)
CS 124: Course Review for Final Exam

- Grades So Far (with Top Students)
- Exam Format: Similar to Midterms
- Class Quizzes
- Some Topic Highlights

Exercise: \[ \sum_{i=1}^{n} i - \frac{n(n+1)}{2} \] (Chapter 1, Slide 14)

- Base case (n=1):
  - Verify LFS = RHS
- Inductive hypothesis (n=k-1):
  - Assume LFS = RHS
- When n=k,
  - Prove LFS = RHS
Example: binary search (Chapter 2)

```cpp
// Search for x in an array a[i..j] sorted in the ascending order.
// If found then return the array index m such that a[m] = x
// else return NOT_FOUND;
int a[];
binsearch(x, i, j) {
    m = floor((i+j)/2);
    if (x == a[m]) then return m;
    else if (i == j) then return NOT_FOUND;
    else if (x < a[m]) then return binsearch(x, i, m-1);
    else return binsearch(x, m+1, j);
}
```

Run-time: the number of comparisons between array elements
Problem size: the number of elements in the array
Recurrence equation:
\[ T(N) = T(N/2) + 2 \text{ if } N >= 2 \]
\[ = 1 \text{ if } N = 1. \]

Let \( N = 2^n \). Then, recursively,
\[ T(2^n) = T(2^{n-1}) + 2 \]
\[ T(2^{n-1}) = T(2^{n-2}) + 2 \]
\[ . \]
\[ T(2^2) = T(2) + 2 \]
\[ T(2) = T(1) + 2 \]
\[ = 1 + 2n \]
Hence, \( T(N) = 1 + 2 \log N = O(\log N) \)
2-D arrays vs. 2-D multi-lists for a matrix (Chapter 3)

- Let \( m \), \( n \), and \( k \) be the numbers of columns, rows, and non-zero elements, respectively.

Then:

<table>
<thead>
<tr>
<th></th>
<th>2-D array</th>
<th>Multi-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage space</td>
<td>( \Theta(m \cdot n) )</td>
<td>( \Theta(m+n+k) )</td>
</tr>
<tr>
<td>Row access time</td>
<td>( \Theta(m) )</td>
<td>( O(m) )</td>
</tr>
<tr>
<td>Column access time</td>
<td>( \Theta(n) )</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>Element access time</td>
<td>( \Theta(1) )</td>
<td>( O(k) )</td>
</tr>
</tbody>
</table>

A multi-list is better if the matrix is **sparse**.

---

Summary: rotations in AVL and splay (Chapter 4)

<table>
<thead>
<tr>
<th>Case</th>
<th>AVL tree</th>
<th>Splay tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>zig-zig</td>
<td>single rotation</td>
<td>top-down double rotation (the roots)</td>
</tr>
<tr>
<td>( \text{left-left or right-right} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zig-zag</td>
<td>bottom-up double rotation</td>
<td>bottom-up double rotation (the current node)</td>
</tr>
<tr>
<td>( \text{otherwise} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison of collision resolution schemes (Chapter 5)

- **Separate chaining**
  - Use of links - costs memory and calls on memory allocation
  - Hence typically is expensive.
- **Linear probing**
  - Easily implementable
  - Performance degrades severely as the load factor increases, due to primary clustering.
- **Quadratic probing**
  - Slightly more difficult to implement with good performance
  - An insertion can (but not likely) fail if the table is half empty.
- **Double hashing**
  - Eliminates primary and secondary clustering;
  - The computation of a second hash function can be costly.

Sorting: Summary (Chapter 7)

1. Selection sort
2. Bubble sort
3. Insertion sort (Section 7.1) $O(N^2)$
4. Shellsort (Section 7.4)
5. Heapsort (Section 7.5) $O(N \log N)$
6. Mergesort (Section 7.6) $O(N \log N)$
7. Quicksort (Section 7.7) $O(N \log N)$
8. Counting sort
9. Radix sort (Section 7.11) $O(N) – one of the most creative CS topics!$
10. External sorting (Section 7.12)
Graph traversal: algorithms (Chapter 9)

- **Breadth-first** search (or traversal)
- **Depth-first** search (or traversal)

(We consider connected graphs here.)

![Graph traversal images]

Dijkstra’s algorithm: example (1/2)

- **Start vertex:** A

![Dijkstra’s algorithm images]
Dijkstra’s algorithm: example (2/2)