Lecture 15: Asymptotics 2

9/30/2020

Loops

Loops Example 1: Based on Exact Count

- Find order of growth of worst case runtime:

```
int N = A.length;
for (int i = 0; i < N; i += 1)
    for (int j = i + 1; j < N; j += 1)
        if (A[i] == A[j])
            return true;
return false;
```

- Worst case number of `==` operations:
    - Given by area of right triangle of side length N-1
    - Area is Theta(N^2)

Loops Example 2

```
int N = A.length;
for (int i = 1; i < N; i = i * 2)
    for (int j = 0; j < i; j += 1)
        System.out.println("hello");
        int ZUG = 1 + 1;
return false;
```

- C(N) = 1 + 2 + 4 + ... + N = 2N - 1, if N is a power of 2
- Number of prints lies between 0.5N and 2N
- The runtime complexity is, in fact, Theta(N)

No Magic Shortcut

Repeat After Me...

- There is no magic shortcut for these problems (well... usually)
    - Runtime analysis often requires careful thought
    - CS70 and CS170 will cover this in much more detail
    - This is not a math class, we expect you to know these:
        - 1 + 2 + 3 + ... + N = N(N+1)/2 = Theta(N^2)
        - 1 + 2 + 4 + ... + N = 2N - 1 = Theta(N) (Where N is a power of 2)
    - Strategies:

- Find exact sum
- Write out examples
- Draw pictures
- Use geometric intuition

Recursion

Recursion (Intuitive)

```
public static int f3(int n) {
    if (n <= 1)
        return 1;
    return f3(n-1) + f3(n-1);
}
```

- Our time complexity is Theta(2^N)
- Every time we increase N by 1, we double the work!

Recursion and Exact Counting

- Another approach: count number of calls to f3, given by C(N)
  - C(1) = 1
  - C(2) = 1 + 2
  - C(N) = 1 + 2 + 4 + ... + 2^(N-1) = 2(2^(N-1)) - 1 = 2^N - 1
- Since work during each call is constant:
  - R(N) = Theta(2^N)

Recursion and Recurrence Relations

- Count number of calls to f3, by a "recurrence relation"
  - C(1) = 1
  - C(N) = 2C(N-1) + 1
- More technical to solve. Won't do this in our course

Binary Search

Binary Search Intuitive

- Finding a key in a sorted array
  - Compare key against middle entry
    - Too small, go left
    - Too big, go right
    - Equal, found
- The runtime of binary search is Theta(log_2(N))
- Why? Problem size halves over and over until it gets down to 1

Binary Search Exact Count

- Find worst case runtime for binary search
    - What is C(6), number of total calls for N = 6?
        - 3
    - Three total calls, where N = 6, N = 3, and N = 1
    - C(N) = floor(log_2(N)) + 1
    - Since compares take constant time, R(N) = Theta(floor(log_2(N)))
        - This f(N) is way too complicated. Let's simplify.
            - Three useful properties:
                - floor(f(N)) = Theta(f(N))
                    - The floor of f has the same order of growth as f
                - ceiling(f(N)) = Theta(f(N))
                    - The ceiling of f has the same order of growth as f
                - log_p(N) = Theta(log_q(N))
                    - logarithm base does not affect order of growth
            - Hence, floor(log_2(N)) = Theta(log N)
- Since each call takes constant time, R(N) = Theta(log N)

Binary Search (using Recurrence Relations)

- C(0) = 0
- C(1) = 1
- C(N) = 1 + C((N-1)/2)

Log Time is Really Terribly Fast

- In practice, logarithm time algorithms have almost constant runtimes
    - Even for incredibly huge datasets, practically equivalent to constant time

Merge Sort

Selection Sort: A Prelude to Mergesort

- Earlier in class we discussed a sort called selection sort:
    - Find the smallest unfixed item, move it to the front, and "fix" it
    - Sort the remaining unfixed items using selection sort
- Runtime of selection sort is Theta(N^2)
    - Look at all N unfixed items to find smallest
    - The look at N-1 remaining unfixed
    - ...
    - Look at last two unfixed items
    - Done, sum 2+3+4+5+...+N = Theta(N^2)
- Given that runtime is quadratic, for N = 64, we might say the runtime for selection sort is 2048 arbitrary units of time (AU)

The Merge Operation

- Given two sorted arrays, the merge operation combines them into a single sorted array by successively copying the smallest item from the two arrays into a target array

- What is the time complexity of the merge operation?
  - Theta(N)
  - Why? Use array writes as cost model, merge does exactly N writes

Using Merge to Speed Up the Sorting Process

- Merging can give us an improvement over vanilla selection sort:
  - Selection sort the left half: Theta(N^2)
  - Selection sort the right half: Theta(N^2)
  - Merge the results: Theta(N)
- N = 64: ~1088 AU
  - Merge: ~64 AU
  - Selection sort: ~2*512 = ~1024 AU
- Still Theta(N^2), but faster since N + 2*(N/2)^2 < N^2
  - 1088 vs 2048 AU for N=64

Two Merge Layers

- Can do even better by adding a second layer of merges
  - Two layers of merges: ~640 AU

Example 5: Mergesort

- Mergesort does merges all the way down (no selection sort):
  - If array is of size 1, return
  - Mergesort the left half
  - Mergesort the right half
  - Merge the results
- Total runtime to merge all the way down: ~384 AU
  - Top layer: ~64 = 64 AU
  - Second layer: ~32 * 2 = 64 AU
  - Third layer: ~16 * 4 = 64 AU
  - Overall runtime in AU is ~64k, where k is the number of layers
  - k = log_2(64) = 6, so ~384 total AU

Mergesort Order of Growth

- Mergesort has worst case runtime = Theta(N log N)
  - Every level takes ~N AU
    - Top level takes ~N AU
    - Top level takes ~N/2 + N/2 = ~N
    - etc. etc.
  - Thus, total runtime is ~Nk, where k is the number of levels
    - Note that k = log_2(N)
  - Overall runtime is Theta(N log N)

Linear vs. Linearithmic (N log N) vs Quadratic

- N log N is basically as good as N, and is vastly better than N^2

- For N = 1000000, the log N is only 20

Summary

- Theoretical analysis of algorithm performance requires **careful thought**
  - There are **no magic shortcuts** for analyzing code
  - In our course, it's OK to do exact counting or intuitive analysis
    - Know how to sum 1 + 2 + 3 + ... + N and 1 + 2 + 3 + ... + N
    - We won't be writing mathematical proofs in this class
  - Many runtime problems you'll do in this class resemble one of the five problems from today. See textbook, study guide, and discussion for more practice
  - This topic has one of the highest skill ceilings of all topics in the course
- Different solutions to the same problem may have different runtimes
  - N^2 vs. N log N is an enormous difference
  - Going from N log N to N is nice, but not a radical change