CS 209
Data Structures and Mathematical Foundations

03 / 27 / 2024

Instructor: Michael Eckmann

Today's Topics

- Questions?/Comments?
- Divide and Conquer (D&C) technique
  - Look back at mergesort implementation
  - Analyze mergesort runtime
  - Consider applying D&C to MaxCSS

Divide & Conquer

- What is it?

Divide and Conquer

- The divide and conquer technique is a way of
  - converting a problem into smaller problems that can be solved individually and then
  - combining the answers to these subproblems in some way to solve the larger problem
- DIVIDE = divide into smaller problems and solve them recursively, except the base case(s)
- CONQUER = compute the solution to the overall problem by using the solutions to the smaller problems solved in the DIVIDE part.

Analyze runtime of MergeSort

- Let's look at my MergeSort implementation
- And do an example of the merging of two sorted lists
- Then see if we can determine the runtime of the work done in mergesort (independent of the 2 calls).

Analyze runtime of MergeSort

- Because it is recursive, we need to count how many calls are made and add up the amount of work done in each call.
- In other words, if we figure out how much work is done during each call and add all that work up, we will determine the overall running time.

Analyze runtime of MergeSort

- Let's build a tree of all the calls made for a list of size n
- Then let's figure out how much work is done at each "level" of this tree of calls.
- Then add that all up.

Analyze runtime of MergeSort

- Each level of the tree does some constant c times n work (c*n) and
- there are lg(n) + 1 levels
- So c * n * (lg(n) + 1) = c*n*lg(n) + c*n = Theta(n*lg(n))

- What if we divided list list into more than 2 portions each time? How would that affect the analysis?

Log of different bases are off by constant factor

$$\log_b(x) = \frac{\log_a(x)}{\log_a(b)}$$

MaxCSS

- Recall the Maximum contiguous subsequence problem:
  - Given an integer sequence A_1, A_2, ..., A_N, find (and identify the sequence corresponding to) the maximum value of $\sum^{j}_{k=i} A_k$. The maximum contiguous subsequence sum is zero if all are negative. Therefore, an empty sequence may be maximum.

Divide and Conquer for MaxCSS

- Apply divide and conquer to the Maximum contiguous subsequence problem.
- We can divide the seqeuence in half each time, like MergeSort does.

- Don't divide when subsequence is length 1. This is base case and the answer is simply the value of the element or 0 if it is negative.
- We will get an answer for each half.
- The answer to the larger problem (the sequence comprising the two halves) is either
  - The answer to the left half
  - The answer to the right half
  - Or the max that spans the two halves

Divide and Conquer for MaxCSS

- The overall result can be either
  - the max on the left side OR
  - the max on the right side OR
  - the max that spans both sides.

Divide and Conquer for MaxCSS

- Maximum sum of a contiguous subsequence of
  - seq[left .. right]

- Conquer part:
  - compute the maxLeftBorderSum
  - compute the maxRightBorderSum
  - decide which is larger
    - maxLeft or
    - maxRight or
    - maxLeftBorderSum + maxRightBorderSum