CSI33 Data Structures

Sharon Persinger

Fall 2019

Day 12 October 15

MergeSort, a recursive sorting algorithm

- Another divide-and-conquer approach
- Take a list of data – numbers or other data with a defined order.
- Put the list into increasing order by
  - Splitting the list into two (approximately) equal pieces.
  - Sort each piece into increasing order.
  - Merge the two sorted lists into one list.

Look at a visualization.

https://www.hackerearth.com/practice/algorithms/sorting/merge-sort/visualize/

Merging two sorted lists is easy.

- Compare the first item in one list to the first item in the other list, and find the smaller one.
- Append that smaller item to the merged list.
- Continue doing this until one or other of the sorted lists is empty.
- If the other list still has elements, append all of those elements onto the merged list.
- We don’t actually remove the items from the lists, as that would increase the time complexity of the algorithm. Just keep track of the indices.

Look at the merge method.

So how do we sort the smaller lists?

- Recursively, using mergeSort
- What's a base case?
- A list with one element or no elements is in increasing order.

Analysis

- Time complexity $\Theta(n \log n)$, n is the number of items
- Log n levels
- Merging n data items at each level
- Merge is $\Theta(n)$
- Uses $\Theta(n)$additional memory since the lists are copied.
- $\Theta(n \log n)$ is the best achievable time complexity for comparison sorts.

Tower of Hanoi

- Mathematical puzzle
- Move all the disks from one post to another post, subject to these rules:
- Move one disk at a time.
- Disk must be placed on a post.
- Larger disk cannot be placed on top of a smaller disk.

Animation of solution

http://cs.armstrong.edu/liang/animation/web/TowerOfHanoi.html

Problem: Write a program that writes instructions for solving the Tower of Hanoi problem for n disks.

We can see the recursion in the animation.

Look again at the animation for n = 2.

To move the tower of two disks from A to B

Move the top disk from A to C.

Move the bottom disk from A to B.

Move the top disk from C to B.

To move the tower of 3 disks from A to B

Move the top tower of two disks from A to C.

Move the bottom disk from A to B.

Move the top tower of two disks from C to B.

The solution for n = 3 uses the solution for n = 2. That’s recursion.

Problem: Write a program that writes instructions for solving the to the Tower of Hanoi problem for n disks.

Look at the code. Really easy.
We need general labels for the posts – source, dest (for destination), temp (for temporary) – so we can use the same algorithm no matter which post is the source and which the destination.
The base case is moving one disk from the source to the destination.
Time complexity is $\Theta(2^n)$. Each case has two recursive calls and the size of the problem is reduced by 1 each time.

How many operations would it take to solve the problem for 64 disks?

True-False assignment

Programming assignment

Write a program that finds the power set of a set of n elements.
Another 2^n problem.