Homework 1 Writeup

Jackson Hart

April 12th, 2022

Problem 1

A)

Algorithm 1 Shopping Spree

stuff

Algorithm 2 Shopping Spree

procedure SHOPPING SPREE($list$)

$length \leftarrow$ length of the list

if $length > 1$ **then return** $length$

$third \leftarrow \lfloor length/3 \rfloor$

$Bottom \leftarrow MergeSort3(list[0 \,..\, third])$
$Mid \leftarrow MergeSort3(list[third + 1 \,..\, 2 \times third])$
$Top \leftarrow MergeSort3(list[2 \times third + 1 \,..\, length - 1])$

$Merge(list, Bottom, Mid, Top)$

Algorithm 3 Merge

```
procedure MERGE(list, Bottom, Mid, Top)
    idx ← i ← j ← k ← 0
    BotLen ← Lengh of Bottom
    MidtLen ← Lengh of Mid
    TopLen ← Lengh of Top
    while Bottom, Mid, and Top have unvisited elements do
        if Bottom[i] ≤ Mid[j] and Top[k] then
            list[idx] ← Bottom[i]
            iterate idx and i
        else if Mid[j] ≤ Bottom[i] and Top[k] then
            list[idx] ← Mid[j]
            iterate idx and j
        else
            list[idx] ← Top[k]
            iterate idx and k
    while Bottom and Mid have unvisited elements do
        if Bottom[i] ≤ Mid[j] then
            list[idx] ← Bottom[i]
            iterate idx and i
        else
            list[idx] ← Mid[j]
            iterate idx and j
```

Algorithm 4 Merge Cont.

```
while Top and Mid have unvisited elements do
    if Top[k] ≤ Mid[j] then
        list[idx] ← Top[k]
        iterate idx and k
    else
        list[idx] ← Mid[j]
        iterate idx and j
while Bottom and Top have unvisited elements do
    if Bottom[i] ≤ Top[k] then
        list[idx] ← Bottom[i]
        iterate idx and i
    else
        list[idx] ← Top[k]
        iterate idx and k
while Bottom or Mid or Top have remaining elements do
    place respective values in the remainder of the list
```

B)

$T(n) = 3T(n/3) + \Theta(n)$

C)

Using the master theorem,

$$log_b(a) = log_3(3) = 1$$

Because $f(n) = n^1$, then we can see this is an example of case 2, so

$$T(n) = \Theta(nlgn)$$

Problem 2

A)

Algorithm 5 Stooge Sort

```
procedure StoogeSort(List, BotIdx, TopIdx)
    if List[BotIdx] > List[TopIdx] then
        Swap bottom and top values
    if (TopIdx − BotIdx + 1) ≥ 3 then
        Third ← ⌊(TopIdx − BotIdx + 1)/3⌋

        StoogeSort(List, BotIdx, TopIdx - Third)
        StoogeSort(List, BotIdx + Third, Top)
        StoogeSort(List, BotIdx, TopIdx - Third)
```

B)

$T(n) = 3T(3n/2) + \Theta(1)$

C)

Using the master theorem,

$$log_b(a) = log_{3/2}(3) \approx 2.71$$

Because $f(n) = n^0$, we can see this is an example of case 1 where $log_{3/2}(3) > \epsilon > 0$. Which then gives us

$$T(n) = \Theta(n^{log_{3/2}(3)}) \approx \Theta(n^{2.71})$$

Problem 5

A

This is my data for stooge sort

And this is my data for merge sort. This data represents the average case running time. I did not take the average of multiple runs.

B)

This is my graph of stoogesort's running times

$$y = (3.5 \times 10^{-7})n^{log_{1.5}3}$$

This one seems to be fairly close, but I probably could've done with doing an average in this case.

$$y = (9.6 \times 10^{-7})nlgn$$

This one seems to fit a bit closer, but I also probably should've averaged my data.