Largest Subarray

Chen, Daiwei Watts, Joseph

February 14, 2019

Abstract

1 Background and Related Work

1.1 Brute Force Algorithm

Algorithm 1 Insertion Sort

```
function InsertionSort(L)
    for i 1..len(L) do
        j ← i
        while j > 0 and L[j] < L[j − 1] do swap(L[j], L[j-1])
            j ← j − 1
        end while
    end for
end function
```

Explain the function here:

Summation Equation goes here

Explain runtime complexity:

1.2 Kadane Algorithm

Algorithm 2 Kadane Algorithm

```
function Kadane(L)
    maxEnding ← L[0]
    maxAlways ← L[0]
    for i 1..len(L) do
        maxEnding ← max(L[i], maxEnding + L[i])
        maxAlways ← max(maxAlways, maxEnding)
    end for
    return maxAlways
end function
```

Kadane is a very good example of a simple but effective way to write a better algorithm using dynamic programming. It focuses on remembering two very important variables $maxEnding$ and $maxAlways$. $maxAlways$ will always remember the largest sum you've seen up till now out of all the subarrays. But $maxEnding$ will keep track of the largest subset just within that iteration of i. Because it remembers what sort of sums you've checked before, you do not have to check other possible subarrays again. Thus getting rid of the 2 for loops within the brute force algorithm.

$$\sum_{1}^{n} 2 = 2 * n = \Theta(n)$$

Explain big O and big Θ here

2 Experimental Setup

RUST HAS BIG PP OWO

This is how we timed our setup

3 Results

Include our graph visualization of our data here. Brute Force and Kadane Timing

4 Conclusions