Algorithms and Data Structures

Luke Nigro - Homework 2

1. Divide-and-conquer algorithms work by splitting large problems into many smaller and more easily solvable sub-problems using a dividing function. The solutions of the sub-problems are found with a conquer function (also usually an algorithm) and are then merged together using a merge function. Some good examples of using a divide-and-conquer approach would be to sort large lists of number, or to find minimum and maximum values of those lists. It is also a useful framework when designing algorithms in general, since it is the basis of many current algorithms such as the merge sort.

2. $T(n) = 2T(\lfloor \frac{n}{2} \rfloor + 17) + n$ is similar enough to $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$

   We assume that $T(\lfloor \frac{n}{2} \rfloor) \leq c\lfloor \frac{n}{2} \rfloor log(\lfloor \frac{n}{2} \rfloor)$

   $T(n) \leq 2(c\lfloor \frac{n}{2} \rfloor log(\lfloor \frac{n}{2} \rfloor) + n \leq cnlg(\frac{n}{2}) + 2$
   $= cnlg(n) - cnlg(2) + n = cnlg(n) - cn + n \leq cnlg(n)$

   Therefore $T(n) = O(nlgn)$

3. In order to solve the maximum contiguous subarray problem, we can implement Kadane's Algorithm which can find the sum of the maximum contiguous subarray in $\Theta(n)$ time. This algorithm is implemented as follows:

```
max_so_far = A[1]                  Θ(1)
max_of_current_sub = A[1]          Θ(1)
  for each i in A[2 ...n]          2T(n)
    max_of_current_sub = max(i, max_so_far + i)
    max_so_far = max(max_so_far, max_of_current_sub)
retrieve value of max_so_far
```

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