Lecture 12 Ticket

COSC 311: Algorithms, Fall 2022

Name: _____

In class we saw the "Master Theorem" for solving recurrences:
Master Theorem. Suppose the running time $T(n)$ of a (recursively defined) method satisfies $T(n) = aT(n/b) + f(n)$. Define $c = \log_b a$. Then:

1. If $f(n) = O(n^d)$ for $d < c$ then $T(n) = O(n^c)$.
2. If $f(n) = \Theta(n^c \log^k n)$ then $T(n) = O(n^c \log^{k+1} n)$.
3. If $f(n) = \Omega(n^d)$ for $d > c$, then $T(n) = O(f(n))$.

For the following methods, apply the Master Theorem to derive a bound on the running time of the method. In particular, for each method you should compute the values a, b, c, and the function f to determine which (if any) of the three conditions above apply. In both cases, the size n of the input is $n = j - i$.

1. **Binary Search.**

```
BinarySearch(a, val, i, j):
  if j = i then return false
  if j - i = 1 then return a[i] = val
  m <- (j + i) / 2
  if a[m] > val then
    return BinarySearch(a, val, i, m)
  else
    return BinarySearch(a, val, m, j)
  endif
```

2. **Merge Sort.** Assume that the `Merge` procedure runs in time $O(n)$.

```
MergeSort(a, i, j):
  if j - i = 1 then
    return
  endif
  m <- (i + j) / 2
  MergeSort(a,i,m)
  MergeSort(a,m,j)
  Merge(a,i,m,j)
```