Name: Yamchips

Collaborators: None

Problem 2-1.

(a) According to *Master Theorem*, suppose we have a recurrence of the form
$T(n) = aT(n/b) + f(n)$
where n is the size of input, a is the number of subproblems in the recursion, n/b is the size of each subproblem, $f(n)$ is the cost of the work done outside the recursion call, which includes the cost of dividing the problem and cost of merging the solutions.
$T(n)$ has the following asymptotic bounds:

1. if $f(n) = O(n^{log_b a - \epsilon})$, then $T(n) = \Theta(n^{log_b a})$.
2. if $f(n) = O(n^{log_b a})$, then $T(n) = \Theta(n^{log_b a} logn)$.
3. if $f(n) = O(n^{log_b a + \epsilon})$, then $T(n) = \Theta(f(n))$.

where $\epsilon > 0$ is a constant
In this case, $a = 4, b = 2$, that means $n^{log_b a} = n^2$, while $f(n) = O(n^{2-1})$, that makes $T(n) = \Theta(n^2)$ by case 1 of the Master Theorem. This is true no matter the choice of $f(n) \in O(n)$.

(b) According to *Master Theorem*, we have $a = 3, b = \sqrt{2}$ in this case, that means $n^{log_b a} = n^{2log_2 3}$. If $f(n) = n^4$, then $T(n) = O(n^4)$ by case 3 of tHE Master Theorem, since $f(n) = n^4 = n^{2log_2 3 + \epsilon}$. If $f(n) = 0$, then $T(n) =$

(c) According to the assumption, $T(n) \leq 2T(n/3) + \Theta(n)$, let $T_1(n) = 2T(n/3) + \Theta(n)$. Since $f(n) = n = n^{log_3 2 + \epsilon}$, that means Case 3 in Master Theorem applies, $T_1(n) = \Theta(f(n)) = \Theta(n)$, so $T(n) \leq \Theta(n)$. While $T(n) = T(n/3) + T(n/4) + \Theta(n) \geq \Theta(n)$, that makes $T(n) = \Theta(n)$.

(d)

Problem 2-2.

(a)

(b)

(c)

Problem 2-3.

Problem 2-4.

Problem 2-5.

(a)

(b)

(c) Submit your implementation to `alg.mit.edu`.