Problem 4

a) $T(n) = \frac{2\mathrm{n}}{5} + \frac{1}{5}\big(1 + T(n)\big) + \frac{1}{5}\big(3 + T(n)\big) + \frac{1}{5}\big(5 + T(n)\big)$

   $T(n) = \frac{2\mathrm{n}}{5} + \frac{1}{5}\big(9 + 3T(n)\big) = \frac{2\mathrm{n}+9}{5} + \frac{3}{5}T(n)$ ---- recurrence relation

   $\frac{2}{5}\mathrm{T(n)} = \frac{2\mathrm{n}+9}{5}$

   $T(\mathrm{n}) = \mathrm{n} + \frac{9}{2}$

b) The worst-case should be for each time we call the new key, the new key is at the end of the dictionary. And for searching the keys that we already searched, it needs to be as behind as possible.

   Therefore, for k times searching new key, the time we need is $k \cdot n$. Since we need to go through the entire list.

   For (m-k) times we searching the searched key, the time we need is $(m - k) \cdot k$

   Therefore, the exact runtime need is $k \cdot n + (m - k) \cdot k = k(n + m - k)$

   This expression should be $\Theta\big(k(n + m)\big)$, since k < m

   Example searching sequence: $\mathrm{a_n, a_{n\text{-}1}, a_{n\text{-}2}, a_n, a_{n\text{-}1}, a_{n\text{-}2}}$

   This is the case where m = 6, k = 3. For the first search, we go through the entire list. And for the later three searches, each of them are at the farthest updated position (in this example, position 2 if counting from 0).

   So the time need for this example is n * 3 + 3 * 3 = 3(n + 6 - 3)