Ushtrimi 1

Analiza per te gjitha rastet behet per nje input te caktuar me madhesi ***n***, qe ne rastin tone esht ***n*** fjale dhe varion nga 100-1000000. Kompleksitetet totale llogariten bashke me kohen qe i duhet programit te lexoje keto fjale nga fjalori qe eshte O(n).

- Ne rastin e listes se lidhur nje drejtimore, kompleksiteti per te shtuar nje element te ri ne liste eshte O(n) pasi elementet shtohen ne fund dhe ne kemi te ruajtur si reference vetem elementin e pare te listes. Megjithate kompleksiteti per te kerkuar nje element ne liste eshte O(n) ne rastin me te keq pasi elementet nuk jane ruajtur ne vendodhje te njepasnjeshme ne memorien e PC.
    - Kompleksiteti i programit: O(n) * (O(n) + O(n)) = $O(n^2)$

- Ne rastin e pemes binare, kompleksiteti per te shtuar nje element te ri eshte O(n) ne rastin me te keq ku te gjitha elementet shtohen ne te njejtin drejtim dhe ka te njejten strukture si nje liste e lidhur. Per ta kerkuar per te njejtin skenar do jete perseri O(n).
    - Kompleksiteti i programit: O(n) * (O(n) + O(n)) = $O(n^2)$

- Ne rastin e pemes binare te balancuar kemi gjithmone nje kompleksitet O(logn) per te shtuar dhe gjithashtu per te kerkuar nje element ne peme. Prandaj ketu kemi nje rritje ne eficencen e programit.
    - Kompleksiteti i programit: O(n) * (O(logn) + O(logn)) = O(n*logn)

- Ne rastin e tabelave hash, kemi nje kohe konstante O(1) per te shtuar si dhe per te kerkuar nese nje element eshte ne liste apo jo. Ne rastin e kerkimit, mund te ndodhe qe kompleksiteti te rritet disi nese ka perplasje ne kodin hash qe gjenerohet per nje celes te caktuar, po rastin qe kjo te ndodhe eshte i rralle.
    - Kompleksiteti i programit: O(n) * (O(1) + O(1)) = O(n)

Sic mund ta shohim, ne rastin konkret ku meret parasysh dhe fakti qe me e rendesishme per ne eshte shpejtesia sesa memoria, perdorimi i tabelave hash eshte menyra me efikase per te ndertuar fjalorin tone te fjaleve.