Oppgave 1

Binærsøk fungerer ved å repetetivt dele en sortert liste i to halvdeler, der man sjekker midten av listen for å avgjøre om elementet man søker etter er mindre eller større enn midten. Dersom elementet er mindre, søker man videre i venstre halvdel og hvis det er større søker man i høyre halvdel.

Vanlig binærsøk på en **array** har en tidskompleksitet på $O(log(n))$ fordi man halverer antallet mulige posisjoner ved hvert trinn. Dette er mulig fordi man kan få direkte tilgang til et hvilket som helst element i en array på $O(1)$ tid. Dermed blir delingsprosessen svært effektiv.

Når det gjelder en **lenket liste**, er det en fundamental forskjell sammenlignet med en array: Tilgangstiden til et vilkårlig element i en lenket liste er $O(n)$ fordi man må iterere gjennom listen fra starten for å nå en gitt posisjon. I store-O notasjon ser man bort ifra konstante faktorer når man analyserer kjøretiden. Altså, selv om det tar $\frac{n}{2}$ operasjoner for å finne midten, regner man dette fortsatt som $O(n)$ tid. Dette påvirker direkte ytelsen til binærsøk, som er avhengig av rask tilgang til midtpunktet av listen.

Hvert trinn i algoritmen krever at man finner midtpunktet i listen, som i en lenket liste krever lineær tid. For å finne midtpunktet vil det ta $O(n)$ tid, siden man må følge pekerne fra starten av listen til midtpunktet. Siden man gjentar denne operasjonen $O(log(n))$ ganger (hver gang man deler listen), blir total kjøretid $O(nlog(n))$.

Valget av lenket liste som datastruktur har derfor en dramatisk innvirkning på ytelsen til binærsøket, fordi man ikke kan få direkte tilgang til midtelementet i listen like effektivt som i en array. Dette resulterer i en kjøretid på $O(n \log n)$, i stedet for $O(\log n)$ som man ville hatt med en array.