Forklaring av algoritme:

Algoritmen starter ved å lage en ny array med den nåværende prisen til hver dag. Deretter kommer en dobbel for-løkke som sjekker hver enkel pris mot alle prisene som kommer etter den i listen. Gjennom denne prosessen lagres den maksimale differansen mellom salgs- og kjøpspris. Deretter returneres den maksimale profitten.

Hypotese:

Algoritmen får en tidskompleksitet på O(n^2) fordi den inneholder en dobbel for-løkke, og ingen andre elementer øker kompleksiteten ytterligere. Algoritmen har ingen måte for å finne ut om den har funnet den høyeste profitten før begge for-løkkene er ferdig å kjøre, noe som fører til at vi får nedre grense på Ω(n^2), som igjen fører til Θ(n^2).

Resultater:

```
Arraylength: 1000, Amount of arrays: 1000, Completion time:133ms
Arraylength: 2000, Amount of arrays: 1000, Completion time:467ms
Arraylength: 10000, Amount of arrays: 1000, Completion time:9947ms
```

Diskusjon:

Alle testene er kjørt med like mange arrays, noe som gjør at når lengden av hver array dobles, bør test-tiden firedobles. Sammenligner man testen med lengde 1000 med testen med 2000, ser vi at kjøretiden har økt med en faktor på 3.51, noe som ligner på den hypotetiske faktoren 4.

Når man sammenligner testen av lengde 1000 med den tidoblede vil man kunne forvente en kjøretid som blir 100 ganger så stor. I den utførte testen får vi en kjøretid som er 74.8 ganger så lang om man kjører. Dette resultatet gjenspeiler hypotesen om at algoritmen har en tidskompleksitet på o(n^2), og hadde man kjørt en enda lengre test ville man gjerne fått et forhold som går mot 100.