Aflevering uge 1

Sara, Luc og Keinicke

2. september 2020

A

a exists(A, 8, 5) returnerer True

b exists(A, 8, 22) returnerer False

c exists(A, 4, 17) returnerer False, pga. at lo $>$ hi

d Ved kald af exists(A, 8, 22) vil der i sidste ende blive returneret False. Indtil da vil mid antage forskellige værdier for hver gennemkørsel af while loopet:
Floor funktionen returnerer det tætteste hele tal på dets input, men runder altid ned. inputtet er gennemsnittet af hi og lo, så vi får her mid = 3 i første omgang.
Efterfølgende får vi lo = mid + 1 = 3 + 1, så vi får i anden omgang at mid = 5.
Nu bliver lo = 5 + 1 = 6 og vi får derfor at mid = 6.
Nu bliver lo = 7 og vi får derfor at mid = 7.
Nu er x $<$ A[mid], derfor bliver hi = 7 - 1 = 6 $<$ lo, derfor stopper loopet og vi får returneret False.

B

Funktionen exists tager et input array, længden på dette array og et tal x. Funktionen returnerer herefter True hvis tallet x findes i dette array, eller False hvis ikke tallet forekommer heri. Variablerne hi og lo har til opave at indsnævre intervallet vi leder i. Så vi starter med hele arrayet, alle pladser kunne indeholde tallet vi leder efter, så vidt vi ved. Det gør vi ved at lo er det mindste indeks 0 og hi er det største index n - 1. Vi starter nu et while loop som kører så længe lo altid er lavere eller end med hi, ellers leder vi bare arrayet gennem flere gange. Vi husker nu at A er sorteret, så vi starter med at kigge efter tallet x i midten af arrayet, heraf variablen mid, som er et gennemsnit af high og low(ved decimaltal rundes der ned). Vi tjekker om pladse A[mid] indeholder tallet x, eller om x er større eller mindre end dette tal. Hvis x er større, må vi formode at vi skal lede mod den større ende af A, da A er sorteret. Er x derimod mindre end A[mid] må vi lede oppe i den høje ende. Er x derimod lig A[mid], så returneres True, da vi bekræfter at x findes i A Vi definerer derfor et nyt interval ved enten at rykke den lave ende frem til mid + 1, eller ved at rykke den høje ende ned til mid - 1, alt efter hvilken ende vi skal lede i. Vi har altså delt intervallet i 2 dele. Loopet gentager sig nu og vi fortsætter med at dele arrayet op i halvt så store dele, indtil vi enten har fundet stedet i A hvor x befinder sig, eller indtil lo bliver større end hi, for i det tilfælde har vi kigget hele A igennem og må konkludere at x ikke findes. Der returneres False.
Det skal noteres at n skal være en korrekt repræsentation af størrelsen på vores array, ellers vil hi være for lille. Vi indsnævre derfor intervallet vi kigger i, uden at have sikret os at x ikke findes i det kasserede område af A. Er n derimod for stor, rikerer vi en syntaksfejl, da vi potentielt kigger på et indeks der er større end længden af A.

C

a Ja funktionen vil altid returnerer False hvis ikke x existerer i A, under alle omstændigheder, antaget at n er en korrekt repræsentation af længden på A. Ethvert eksempel gør sig gældende.

b Det kan godt ske, men er ikke givet, da A ikke er sorteret. Vi kan derfor blive fanget i et interval hvor x ikke findes, hvorimod x findes i et andet. Dette kan f.eks. ske hvis det første tal vi kigger på er x > A[mid], så kigger vi på et interval i > mid. Da A ikke er sorteret kan det dog godt være at x befinder sig i intervallet i < mid. Det modsatte kan også gøre sig gældende, vi kan være heldige at ramme det rigtige tal fra starten. Tag f.eks. A fra opgavebeskrivelsen, men hvor alle andre positioner end A[3] er fordelt tilfældigt i A. Her vi exists(A, 8, 10) returnerer True, da A[mid] == 10, fordi mid = 3.

D

Vi deler hele tiden A op i to, så hvis n = 64, så har vi 64 mulige pladser at kigge på. Antaget at A er sorteret vil vi derfor kigge på hvor mange gange vi kan dele 64 med 2 for at få 1 mulig plads, altså:

$$\frac{64}{2^m} = 1 \rightarrow m = lg(64) = 6$$

Hvor m er det maksimale antal gange while loopet kan køre igennem ved n = 64