Finn par som summerer til x — 10 poeng

Du er gitt et array med unike heltall `A` og et heltall x. Du skal skrive en prosedyre som skriver ut alle par av heltall y og z i arrayet, der $y + z = x$. Tallet på en gitt posisjon i arrayet kan maksimalt inngå i én utskrift. Rekkefølgen parene (y, z) skrives ut i spiller ingen rolle, og rekkefølgen innad i parene spiller heller ingen rolle.

Input: Et array `A` av heltall, og et heltall x
Output: Skriver ut alle par $y, z \in$ `A` slik at $y + z = x$

```
1 Procedure FindSummands(A, x)
  | // ···
```

For eksempel skal et kall på `FindSummands(`$[0, 2, 4, 6, 8, 10]$`, 10)` skrive ut parene $(0, 10)$, $(2, 8)$ og $(4, 6)$.

For begge deloppgaver: Gi pseudokode for en algoritme som løser problemet **og** oppgi kjøretidskompleksiteten på algoritmen (den trenger ikke begrunnes). Lavere kjøretidskompleksitet er mer poenggivende.

(a) Skriv en prosedyre `FindSummands` som beskrevet over, med antagelsen om at `A` er *sortert fra minst til størst*.

(b) Skriv en prosedyre `FindSummands` som beskrevet over, men du kan *ikke* anta at `A` er sortert. Hint: Du kan anta $O(1)$ for innsetting, oppslag og sletting i hashbaserte datastrukturer.

Både oppgave (a) og (b) kan enkelt løses i kvadratisk tid, slik:

```
Procedure FindSummands(A, x)
    for i ← 0 to |A| − 1 do
        for j ← i + 1 to |A| − 1 do
            if A[i] + A[j] = x then
                print(A[i], A[j])
```

I oppgave (a) kan vi oppnå lineærtid ved å holde styr på to indekser i og j, der i initielt er 0 og økes, og j initielt er $|A|$ og minkes.

```
Procedure FindSummands(A, x)
    i ← 0
    j ← |A| − 1

    while i < j do
        y ← A[i] + A[j]
        if x < y then
            j ← j − 1
        else if x > y then
            i ← i + 1
        else
            print(A[i], A[j])
            i ← i + 1
```

I oppgave (b) kan vi oppnå $O(n)$ ved å sortere med radix-sort først, og deretter bruke løsningen fra (a), eventuelt $O(n \cdot \log(n))$ ved merge sort eller lignende. Vi kan også oppnå forventet linærtid ved å lagre elementer vi har sett i en mengde (implementert ved hashing), og for hvert element i $y \in A$ sjekke om $x - y$ er i mengden.

```
Procedure FindSummands(A, x)
    H ← new HashSet()
    for y in A do
        if x − y is in H then
            print(y, x − y)
            H.remove(x - y)
        else
            H.add(y)
```

For full uttelling forventer vi $O(n)$ i forventet kjøretidskompleksitet for (a) og (b). Dersom noen har en $O(n)$ algoritme i oppgave (b), med en kommentar om at den kan brukes for (a), så gir det full uttelling. Man kan også få full uttelling for a bruke løsningen fra (a) og bruke radix sort først i (b).

For en $O(n \cdot \log n)$ løsning på (b) kan man få opp til 8 poeng.

For kvadratisk tid på begge løsninger kan det gis opp til 3 poeng.