Complexiteit

Uitwerkingen

Opgave 1.

Aanname: vermenigvuldigen van twee getallen is de basis operatie.
a. Reken x^n uit als $x \times x \times x \times \cdots \times x$, geeft $n-1 \in \Theta(n)$ vermenigvuldigingen.
b. Het vinden van de dichtstbijzijnde 2 macht kleiner of gelijk aan n door herhaaldelijk te kwadrateren kost $\lfloor \log_2 n \rfloor$ vermenigvuldigingen. Om vervolgens met de uitgerekende x^1, x^2, x^4, etc. x^n te krijgen kost hooguit nog eens $\lfloor \log_2 n \rfloor$ vermenigvuldigingen. De worst case is dus $2\lfloor \log_2 n \rfloor \in \Theta(\log n)$.
c. Aangenomen dat het delen van twee getallen even veel moeite kost als vermenigvuldigen kan x^{15} sneller uitgerekend worden als x^{16}/x^1. In dit geval blijft de complexiteit wel $\Theta(\log n)$.

Opgave 2.

a.

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix} = \begin{pmatrix} 1 \times 8 + 3 \times 6 & 1 \times 4 + 3 \times 2 \\ 5 \times 8 + 7 \times 6 & 5 \times 4 + 7 \times 2 \end{pmatrix}$$

b. In het algemene geval hebben we voor elk inwendig product n vermenigvuldigingen nodig, en $(n-1)$ optellingen. Voor $n \times n$ matrices worden er n^2 inwendig producten worden uitgerekend. Dus in totaal n^3 vermenigvuldigingen, en $n^2(n-1)$ keer optellen. Voor beide gevallen is dit $\Theta(n^3)$.
c. Noem het aantal elementen in een $n \times n$ array $N = n^2$. We krijgen $n = N^{1/2}$ en dus $N^{3/2}$ vermenigvuldigingen en $\sim N^{3/2}$ optellingen.
d. Ja, het vermenigvuldigen van twee getallen is in dit geval een geschikte basis operatie. Zowel het optellen als vermenigvuldigen van twee getallen zijn in dit geval onafhankelijk van de grootte van n. Ook worden er minstens even vaak getallen vermenigvuldigd als opgeteld, voor elke n.
e. Om twee $n \times n$ matrices te vermenigvuldigen zullen i.i.g. alle invoer waardes gelezen moeten worden. Dit zijn er $2n^2$. Tevens moeten er ook n^2 uitvoer waardes geschreven worden. Dit kost dus i.i.g. $\Theta(n^2)$ stappen.
f. Er bestaan algoritmes om matrices te vermenigvuldigen met een complexiteit lager dan $\Theta(n^3)$, maar de optimale complexiteit is onbekend. Het gebruikelijke $\Theta(n^3)$ algoritme is dus niet optimaal (maar ik zie niet hoe je dit uit de opgave kan concluderen).

Opgave 3.

a. In elk doorlopen van dit algoritme wordt deze operatie minstens even vaak uitgevoerd als elke andere operatie. (Het gaat bij deze vraag dus vooral om de plek van deze operatie in het algoritme.)
b. In het beste geval doen we 1 vergelijking als het element dat we zoeken op de eerste plek in de array staat.
c. In het slechtste geval doen we n vergelijkingen als het element dat we zoeken op de laatste plek staat of helemaal niet in de array voorkomt.
d. Gemiddeld $(n+1)/2$ vergelijkingen als X in de array voorkomt:

$$\frac{1}{n}(1+2+3+\cdots+n) = \frac{1}{n}\sum_{k=1}^{n} k = \frac{n+1}{2}$$

Altijd n vergelijkingen als X niet in de array voorkomt. Het totale gemiddelde hangt af van de kans dat de X die we zoeken wel of niet in de array voorkomt.
e. In het slechtste geval, als X bijvoorbeeld niet in A voorkomt, moeten we altijd alle elementen bekijken om dit vast te stellen. (Bewijs in losse uitwerking.)
f. Zelfde idee als bij **e**. Elk element moet minstens één keer bekeken worden, dan wel in een vergelijking met X, dan wel met een ander element. Immers: in een ongesorteerde array hebben we geen informatie over een element wat we nog niet bekeken hebben, en kunnen dus niet constateren of het wel of niet gelijk is aan X.

Opgave 4.

a. In het beste geval doen we 1 vergelijking in de while loop als het element dat we zoeken op de eerste plek in de array staat, of kleiner is dan het eerste element.
b. In het slechtste geval doen we n vergelijkingen in de while loop als het element dat we zoeken op de laatste plek staat, of groter is dan het laatste element in de array. Wat verschilt met opgave 3 is dat we nu in sommige gevallen niet de hele array door hoeven als we een X zoeken niet in de array voorkomt.
c. In het gemiddelde geval doen we $(n+1)/2$ vergelijkingen, in zowel het geval van X in de array als X niet in de array. We krijgen dus $(n+1)/2 \in \Theta(\frac{n}{2})$. Echter $\Theta(\frac{n}{2}) = \Theta(n)$, dus meestal laten we constante factoren binnen de $O/\Omega/\Theta$ weg.

Opgave 7.

a. In het elk doorlopen van dit algoritme wordt deze operatie minstens even vaak uitgevoerd als elke andere operatie.
b. In het slechtste geval (bijvoorbeeld een omgekeerd gesorteerde array) krijgen we $(n-1)(n-1) \in \Theta(n^2)$ vergelijkingen.
c. De verzameling van worst case inputs bestaat uit alle rijen waar elk element het kleinste of een-na-kleinste is van alle voorgaande elementen. (In het geval dat elk element het kleinste is van alle voorgaande elementen hebben we te maken met de omgekeerde array.)
Het grootste element moet in dit geval op de eerste of tweede plek staan (2 mogelijkheden). Vervolgens kan het een-na-grootste element op plek 1, 2, of 3

staan (waarvan er één bezet is door het grootste element, dus weer 2 mogelijkheden). Uiteindelijk is er dan één plek over voor het laatste element, en zijn er in totaal 2^{n-1} mogelijke rijen.