## Correctheid en complexiteit kortste pad met dynamisch programmeren

Op blz. 89–91 van het boek wordt een algoritme beschreven voor het berekenen van het kortste pad in een gerichte acyclische graaf. Dit algoritme maakt gebruik van dynamisch programmeren.
Ervanuitgaande

- dat de graaf $N \geq 1$ knopen bevat
- dat de buren van een knoop $X$ eenvoudig af te lopen zijn met behulp van de volgende constructie:

```
for alle buren Y van X do
{
}
```

- dat het gewicht van de tak van knoop $X$ naar knoop $Y$ te verkrijgen valt met behulp van de functie Gewicht($X$,$Y$)
- dat de knopen in de graaf al geordend zijn in een array Sort (op posities $1 \ldots N$), zó dat er alleen takken van links naar rechts lopen. Meer precies: als er een tak is van knoop $X$ naar knoop $Y$, dan is $X = \text{Sort}[i]$ en $Y = \text{Sort}[j]$, met $1 \leq i < j \leq N$.

kwamen we eerder al tot de volgende pseudo-code implementatie:

```
L [ Sort[N] ] = 0;
for i=N-1 downto 1 do
{ X = Sort[i];
  Min = MaxInt;
  for alle buren Y van X do
  { if (Gewicht(X,Y) + L[Y] < Min)
      Min = Gewicht(X,Y) + L[Y];
  }
  L[X] = Min;
}
write (L[ Sort[1] ]);
```

**a** Wat kan er mis gaan met dit algoritme als er knopen $X$ zijn vanwaaruit de laatste knoop Sort[$N$] helemaal niet te bereiken is?

Neem aan

- dat de laatste knoop Sort[$N$] vanuit iedere knoop $X$ te bereiken is,
- dat de graaf $M$ takken bevat.

**b** Druk de tijdscomplexiteit van het algoritme uit in $N$ (het aantal knopen) en/of $M$ (het aantal takken).

**c** Wat is het verband tussen $N$ en $M$, en heeft dit gevolgen voor je antwoord bij onderbeel **b**?