1. Om een array A met $N \geq 2$ getallen op posities 1,2,...,N van klein naar groot te sorteren, kunnen we gebruik maken van het volgende algoritme (bubble-sort):

```
for (i=1 to N-1)
{ for (j=1 to N-i)
  { if (A[j]>A[j+1])
      verwissel A[j] en A[j+1];
  }
  (*)
}
```

   (a) Pas het algoritme toe op de rij getallen 5 3 2 6 1 4. Dat wil zeggen: geef de volgorde van de getallen na iedere iteratie van de buitenste for-lus.

   (b) Welke van de volgende beweringen is een invariant voor de buitenste for-lus op positie `(*)`? Dat wil zeggen: welke bewering klopt op positie `(*)` voor iedere iteratie van de buitenste for-lus?

   - $A[1 \ldots (i-1)]$ is van klein naar groot gesorteerd en $A[i \ldots N]$ zijn allemaal minstens zo groot als $A[i-1]$.
   - $A[1 \ldots i]$ is van klein naar groot gesorteerd en $A[(i+1) \ldots N]$ zijn allemaal minstens zo groot als $A[i]$.
   - $A[(N-i+1) \ldots N]$ is van klein naar groot gesorteerd en $A[1 \ldots (N-i)]$ zijn allemaal hoogstens zo groot als $A[N-i+1]$.
   - $A[(N-i) \ldots N]$ is van klein naar groot gesorteerd en $A[1 \ldots (N-i-1)]$ zijn allemaal hoogstens zo groot als $A[N-i]$.

   (Met de notatie $A[x \ldots y]$ bedoelen we de getallen op posities x tot en met y van array A.) Motiveer je antwoord.

   (c) Gebruik de (juiste) invariant van het vorige onderdeel om aan te tonen dat het algoritme partieel correct is. Dat wil zeggen: om aan te tonen dat *als* het algoritme eindigt, dat dan het array A van klein naar groot gesorteerd is.

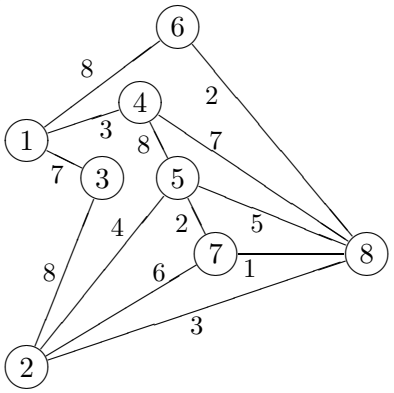
   (d) Wat is het aantal iteraties van de binnenste for-lus in de i-de iteratie van de buitenste for-lus?

   (e) Bereken het totaal aantal iteraties van de binnenste for-lus in alle iteraties van de buitenste for-lus samen. Wat is (dus) de complexiteit van het beschreven algoritme? Motiveer je antwoord.

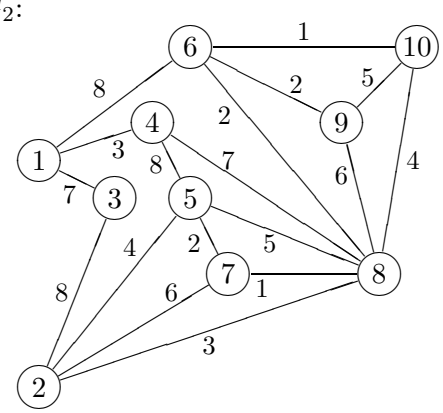
   (f) Is er nog verschil tussen het slechtste geval en het gemiddelde geval bij het beschreven algoritme? Motiveer je antwoord.

2. Beschouw de volgende twee ongerichte grafen met gewichten G_1 en G_2:

G_1:



G_2:



(a) Bepaal met een gretig algoritme een minimale opspannende boom in G_1 (de linker graaf dus). Dat wil zeggen: een boom met minimaal totaal gewicht, die alle knopen met elkaar verbindt. Laat zien hoe je tewerk gaat, en geef ook tussenresultaten.

(b) Bestaat er in G_2 (de rechter graaf dus) een Hamilton pad waarin de tak van knoop 2 naar knoop 5 voorkomt? Zo ja, beschrijf zo'n pad. Zo nee, waarom niet?

(c) Wat is in graaf G_2 de kortste route (of een kortste route als er meerdere zijn) voor een handelsreiziger die begint in knoop 1, vervolgens alle andere knopen precies één keer bezoekt, om daarna in de laatste stap terug te keren naar 1? Wat is de lengte van die route?

(d) De beslissingsvariant van het handelsreizigersprobleem wordt als volgt gedefinieerd:

> Gegeven een ongerichte graaf G met N knopen en met gewichten op de takken, en gegeven een getal K. Bestaat er een route in de graaf die begint bij een knoop, vervolgens alle andere knopen precies één keer bezoekt en dan in de laatste stap terugkeert bij de beginknoop, met een totaal gewicht van hoogstens K?

Geef een niet-deterministisch polynomiaal algoritme voor dit probleem. Neem aan dat de N knopen nummers 1,2,...,N hebben.

3. Orden de volgende vier problemen op 'moeilijkheid' (qua complexiteit / oplosbaarheid), van 'makkelijk' naar 'moeilijk':

- het handelsreizigersprobleem,
- het bepalen van een minimale opspannende boom in een ongerichte graaf,
- zoeken van een element in een gesorteerd array,
- het correspondentieprobleem.

Motiveer de volgorde die je kiest.

4. Het stopprobleem wordt als volgt gedefinieerd:

> Gegeven een computerprogramma R en een mogelijke invoer X voor dat programma. Stopt R als je hem zou draaien voor invoer X?

Toon aan dat het stopprobleem niet oplosbaar is.

Hint: veronderstel dat het stopprobleem wel is op te lossen, met een computerprogramma Q, en leid vervolgens een tegenspraak af.

5. We willen weten of een woord van lengte M in een tekst van lengte N voorkomt. We nemen aan dat het woord is opgeslagen op posities $1, \ldots, M$ van array W, en dat de tekst is opgeslagen op posities $1, \ldots, N$ van array T.

   (a) Beschrijf het naïeve algoritme om te controleren of W in T voorkomt (in woorden of met pseudocode).

   (b) Wat is de tijdscomplexiteit van dit algoritme voor het slechtste geval? Motiveer je antwoord.

   (c) Geef een slechtste geval voor dit algoritme. Dat wil zeggen: een woord W en een tekst T waarvoor het algoritme zoveel mogelijk stappen kost. Gebruik hiervoor $M = 5$ en $N = 15$.

   We gaan nu probabilistisch zoeken, en we gaan er vanuit dat alle letters in het woord en de tekst cijfers 0,1,...,9 zijn. Ieder blok van lengte M is dus een getal van M cijfers. We maken gebruik van vingerafdrukken. We kiezen een random priemgetal K dat ongeveer even groot is als N, en definiëren de vingerafdruk van een blok G van lengte M door:

   $$\text{VingerAfdruk}(G) = G \text{ modulo } K.$$

   (d) In totaal bevat de tekst $N - M + 1$ blokken van lengte M, die met elkaar overlappen. Waarom is het van belang dat we voor het berekenen van de vingerafdruk van een blok gebruik kunnen maken van de vingerafdruk van het vorige blok?

   (e) Stel nu dat $N = 110$, $M = 5$ en $K = 101$. Laat zien hoe we uitgaande van VingerAfdruk(63078), de waarde van VingerAfdruk(30787) kunnen berekenen.

   N.B.: om dit onderdeel te beantwoorden heb je geen rekenmachine nodig! Voor de aardigheid vermelden we dat VingerAfdruk(63078) = 54 en dat VingerAfdruk(30787) = 83.

   (f) Hoe groot is in ons voorbeeld de kans dat een willekeurig blok van lengte M dezelfde vingerafdruk heeft als het woord dat we zoeken? Motiveer je antwoord.

   (g) Waarom is het van belang dat K random is gekozen?