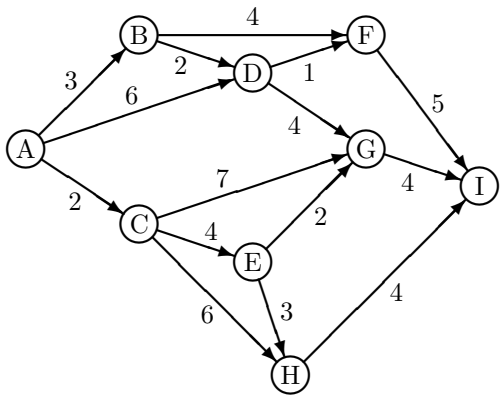
1. Beschouw onderstaande, gerichte, acyclische graaf met gewichten op de takken:



   Voor deze graaf willen we de kortste afstand van knoop A naar knoop I berekenen. Hierbij kunnen we gebruik maken van de functie Gewicht (X,Y) die het gewicht van de tak van X naar Y oplevert.

   (a) Om de kortste afstand van knoop A naar knoop I te berekenen, kunnen we de volgende, recursieve functie gebruiken:

```
int AfstandVanafA (Y)
// bereken de kortste afstand van knoop A naar knoop Y
{
     if (Y==A)
     then return 0;
     else  Minimum = 1000;
           for alle inkomende takken (X,Y) van Y
           do   Afst = AfstandVanafA (X) + Gewicht (X,Y);
                if (Afst < Minimum)
                then Minimum = Afst;
                fi
           od
           return Minimum;
     fi
}
```

   Geef de boom met alle recursieve aanroepen als we de functie AfstandVanafA aanroepen met argument I (we willen tenslotte de kortste afstand van knoop A naar knoop I weten).
   N.B.: je hoeft dus niet AfstandVanafA (I) uit te rekenen, maar je moet alleen laten zien voor welke waardes van Y de functie steeds (recursief) wordt aangeroepen.

   (b) Bereken de kortste afstand van knoop A naar knoop I met behulp van dynamisch programmeren. Maak hierbij gebruik van de ordening A, B, C, D, E, F, G, H, I van de knopen.
   Leg duidelijk uit hoe je te werk gaat, en geef ook tussenresultaten.

2. *Deze opgave bestaat uit veel onderdelen. Dit betekent absoluut niet dat je alle vorige onderdelen goed moet hebben om een bepaald onderdeel te kunnen maken. Dus als je een bepaald onderdeel niet weet, kijk dan toch ook naar de volgende onderdelen!*

   Laat X en Y twee even lange strings zijn, van lengte N, waarbij de N karakters in elk van de strings (gewoon) op posities $0, 1, 2, \ldots, N-1$ staan. We willen nagaan of X en Y aan elkaar gelijk zijn.

   Iemand stelt het volgende algoritme voor, dat de karakters van X en Y van achter naar voren met elkaar vergelijkt:

```
i = N − 1;
NogOK = true;
while (NogOK and (i > 0))
do   if (X[i] ≠ Y[i])
     then NogOK = false;
     fi
     i --;
od

if (i ≥ 0)
then return false;
else return true;
fi
```

(a) Voer het bovenstaande algoritme uit voor X = "abba" en Y = "aaba", waarbij $N = 4$. Geef duidelijk aan welke instructies je achtereenvolgens uitvoert. Is de uitvoer van het algoritme voor deze strings X en Y correct?

(b) Het bovenstaande algoritme is niet correct. Geef een voorbeeld van twee even lange strings X en Y waarvoor het algoritme de verkeerde uitvoer geeft, en leg ook uit waarom het algoritme de verkeerde uitvoer geeft.

Iemand anders stelt het volgende, correcte algoritme voor, dat de karakters van X en Y van voor naar achteren met elkaar vergelijkt:

```
i = 0;
NogOK = true;
while (NogOK and (i < N))
do   if (X[i] ≠ Y[i])
     then NogOK = false;
     else i ++;
     fi
od

return NogOK;
```

(c) Toon aan dat de volgende bewering een invariant is voor de while-lus in het tweede algoritme:

```
( (NogOK = true) and (0 ≤ i ≤ N) and (X[0...(i-1)] = Y[0...(i-1)]) )
or
( (NogOK = false) and (0 ≤ i ≤ N-1) and (X[i] ≠ Y[i]) )
```

Hierin staat $X[0...(i-1)]$ voor de eerste i karakters van de string X, en analoog voor Y.

(d) Gebruik de invariant van het vorige onderdeel om aan te tonen dat het tweede algoritme partieel correct is. Dat wil zeggen: dat *als* het algoritme eindigt, dat de uitvoer dan correct is.

(e) Geef een passende convergent voor de while lus in het tweede algoritme, dat wil zeggen: een uitdrukking die iedere iteratie van de while-lus kleiner wordt, en nooit negatief zal worden. Leg ook uit dat de convergent inderdaad iedere iteratie kleiner wordt.

(f) Hoeveel iteraties doorloopt de while-lus in het tweede algoritme maximaal (als functie van N)? Geef een voorbeeld van twee woorden X en Y van lengte $N = 5$, waarbij dit maximale aantal iteraties gehaald wordt.

(g) Wat is dus de tijdscomplexiteit van het tweede algoritme in het slechtste geval? Motiveer je antwoord.

(h) Stel dat X en Y twee willekeurige woorden van lengte N zijn, met alleen letters 'a' en 'b' erin. Hoe groot is de kans dat de letter $X[i]$ gelijk is aan de letter $Y[i]$ voor een i met $0 \leq i \leq N-1$?

(i) Wat is het gemiddeld aantal iteraties van de while-lus in het tweede algoritme voor twee willekeurige woorden X en Y van lengte N. Motiveer je antwoord.

(j) Wat is dus de tijdscomplexiteit van het tweede algoritme in het gemiddelde geval? Motiveer je antwoord.

3. Deze opgave is uiteindelijk vervallen.

4. Deze opgave gaat over de complexiteit van beslissingsproblmen. We kijken in het bijzonder naar de volgende drie problemen:

   Samengesteldheid: gegeven een geheel, positief getal N. Is N samengesteld?

   Minimale opspannende boom: gegeven een ongerichte graaf met gewichten op de takken, en gegeven een getal K. Bestaat er een minimale opspannende boom van de graaf met een totaal gewicht van hoogstens K?

   Handelsreizigersprobleem: gegeven een graaf 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?

   (a) Wanneer zeggen we dat een beslissingsprobleem in $\mathcal{P}$ zit? Welk probleem (of welke problemen) van de bovenstaande drie problemen, zit (zitten) in $\mathcal{P}$?

   (b) Wanneer zeggen we dat een beslissingsprobleem in $\mathcal{NP}$ zit? Welk probleem (of welke problemen) van de bovenstaande drie problemen, zit (zitten) in $\mathcal{NP}$?

(c) Wanneer zeggen we dat een beslissingsprobleem $\mathcal{NP}$-volledig is? Welk probleem (of welke problemen) van de bovenstaande drie problemen, is (zijn) $\mathcal{NP}$-volledig?

5. Deze opgave gaat over patroonherkennen met het algoritme van Knutt-Morris-Pratt. We zoeken een woord W van lengte M in een tekst T van lengte N. We gaan er hierbij vanuit dat de letters van het woord W op posities 1,...,M van W staan, en dat de letters van de tekst T op posities 1,...,N van T staan. *De letters van het woord en de tekst beginnen dus niet op positie 0.* De failure-links van W staan in het array FLink.

   Wanneer de failure-links bekend zijn, kunnen we de volgende pseudo-code gebruiken om W in T te zoeken:

```
Gevonden = false;
i = 1;
Pos = 1;
while (not Gevonden and (M−i ≤ N−Pos) )
do   if (W[i] == T[Pos])
     then i ++;
          Pos ++;
          if (i > M)
          then Gevonden = true;
          fi
     else i = FLink[i];
          if (i == 0)
          then i ++;
               Pos ++;
          fi
     fi
od

return Gevonden;
```

   (a) Gegeven is dat de failure-links van het woord W = ABAABA als volgt zijn:
   FLink[1] = 0, FLink[2] = 1, FLink[3] = 0, FLink[4] = 2, FLink[5] = 1, FLink[6] = 0.
   Zoek met bovenstaande pseudo-code het woord W in de tekst

   T = ACABACABBABAAABAABAABCA.

   Laat duidelijk zien welke letters van W je steeds met welke letters van T vergelijkt, wat de uitkomst daarvan is, en wat je doet als gevolg van die uitkomst.

   (b) Zoals uit de pseudo-code blijkt, vergelijken we in iedere iteratie van het algoritme van Knuth-Morris-Pratt een letter van het woord W met een letter van de tekst T. Als de letters gelijk zijn, dan gaan we zowel in W als in T een letter naar rechts. Als de letters ongelijk zijn, dan schuiven we het woord W één of meerdere posities naar rechts.

   Leg uit waarom we bij het zoeken nooit meer dan $2N$ iteraties zullen hebben.

   (c) Wat is de tijdscomplexiteit van het algoritme van Knuth-Morris-Pratt? Motiveer je antwoord.

   (d) Leg uit waarom voor W = ABAABA geldt dat FLink[5] = 1.

   (e) Geef de failure-links voor het woord W = AAAAAB.