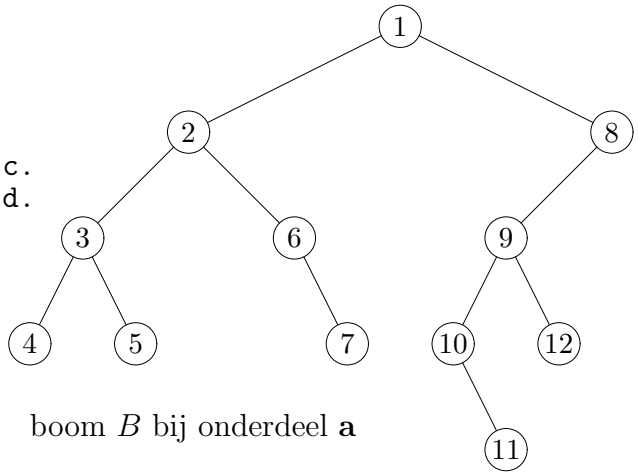
1) De knopen in een symmetrisch bedrade binaire boom zijn van de onderstaande vorm. Hierbij is `TagType` een enumeratie-type dat de waardes `Tak` en `Draad` kent. We gaan onderzoeken met welke alternatieve velden we deze bomen kunnen representeren.

```
struct Knoop
{    int       Info;
     Knoop     *Links, *Rechts;
     TagType   LTag, RTag;
     int       Prenum;   // alleen bij c.
     int       Niveau;   // alleen bij d.
};
```



boom B bij onderdeel **a**

a. Neem bovenstaande binaire boom B over en breng er de symmetrische bedrading (zowel linker- als rechterdraden) in aan.

b. De velden `LTag` en `RTag` geven aan of de pointers `Links` en `Rechts` takken dan wel draden zijn. De wortel van de boom is van het type `Knoop*` en heet `Wortel`. In principe kan de boom leeg zijn.

Geef een algoritme om de Info-velden van de knopen in de boom in de symmetrische (=LWR) volgorde af te drukken. Het algoritme moet gebruik maken van de draden; het moet niet recursief zijn of gebruik maken van een stapel.

c. Veronderstel nu dat iedere knoop in een symmetrisch bedrade boom ook een veld `Prenum` heeft, dat het nummer van de knoop in de pre-orde volgorde bevat.

i. Laat zien dat we de velden `LTag` en `RTag` nu niet meer nodig hebben om te bepalen of een pointer een tak of een draad is.

ii. Waarom is deze representatie (met het veld `Prenum`) niet handig als we de mogelijkheid willen hebben om knopen aan de boom toe te voegen?

d. Veronderstel nu dat iedere knoop in een symmetrisch bedrade boom geen veld `Prenum`, maar een veld `Niveau` heeft, dat het niveau van de knoop in de boom bevat (de wortel zit op niveau 1, zijn kinderen op niveau 2, enzovoort).

i. Laat zien dat we de velden `LTag` en `RTag` ook in dit geval niet meer nodig hebben om te bepalen of een pointer een tak of een draad is.

ii. We gaan er nu vanuit dat iedere knoop in een symmetrisch bedrade boom wél een veld `Niveau` en geen velden `LTag` en `RTag` bevat.
Laat `Deze` naar een knoop in de boom wijzen die nog geen rechter kind heeft (`Deze` is ongelijk aan `NULL`). Schrijf een functie

`void VoegRechtsToe (Knoop *Deze, int NwWaarde)`

die een rechter kind met Info-veld `NwWaarde` toevoegt aan de knoop aangewezen door `Deze`. Uiteraard moet de boom na afloop weer keurig bedraad zijn.

2) **a.** De symmetrische min-max heap (SMM heap) kan op twee (equivalente) manieren gedefinieerd worden. Beide definities beginnen als volgt:

> Een symmetrische min-max heap is een complete binaire boom met een dummy wortel ...

Maak deze definitie op een van de twee manieren af.

b. Construeer een SMM heap met getallen als waardes door, uitgaande van een lege SMM heap, achtereenvolgens de volgende getallen toe te voegen: 17, 11, 31, 5, 41, 3, 59.

Uiteraard dient de boom na iedere toevoeging een SMM heap te zijn, dus borrel zonodig getallen naar hun juiste positie in de heap. Geef met tussenresultaten en een korte toelichting duidelijk aan hoe je aan je antwoord komt.

c. Bij het toevoegen van een getal aan een SMM heap is (soms) een aantal verwisselingen van getallen nodig om weer een SMM heap te verkrijgen.

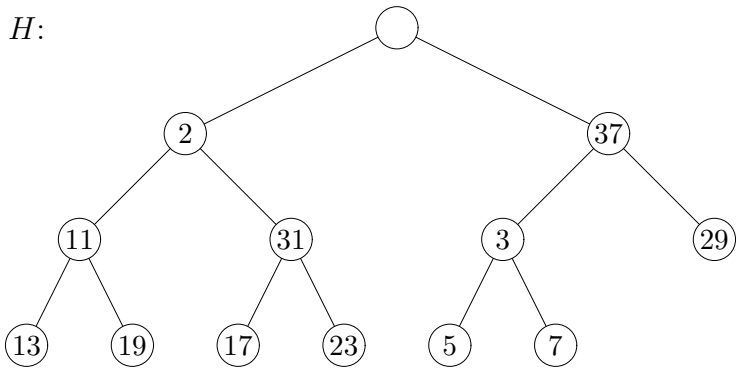
Laat $N \geq 1$ en stel dat we de getallen $1, 2, \ldots, N$ (in een bepaalde volgorde) willen toevoegen aan een aanvankelijk lege SMM heap. Net als in het vorige onderdeel eisen we dat de boom na iedere toevoeging een SMM heap is.

Beschrijf een optimale volgorde voor het toevoegen van de getallen $1, 2, \ldots, N$, d.w.z. een volgorde waarbij in totaal zo weinig mogelijk verwisselingen van getallen nodig zijn. Leg ook uit waarom je volgorde optimaal is.

Als het je niet lukt om een optimale volgorde voor algemene $N \geq 1$ te beschrijven, kun je ook punten verdienen door je te beperken tot $N = 10$. Je geeft dan een optimale volgorde voor het toevoegen van de getallen $1, 2, \ldots, 10$ en je legt uit waarom deze volgorde optimaal is.

d. Beschouw de volgende SMM heap H:

H:



i. Welke SMM heap ontstaat bij uitvoering van de operatie VerwijderMin (DeleteMin) op H?

ii. Welke SMM heap ontstaat bij uitvoering van de operatie VerwijderMax (DeleteMax) op H (de originele SMM heap dus)?

Geef met tussenresultaten en een korte toelichting duidelijk aan hoe je aan je antwoorden komt.

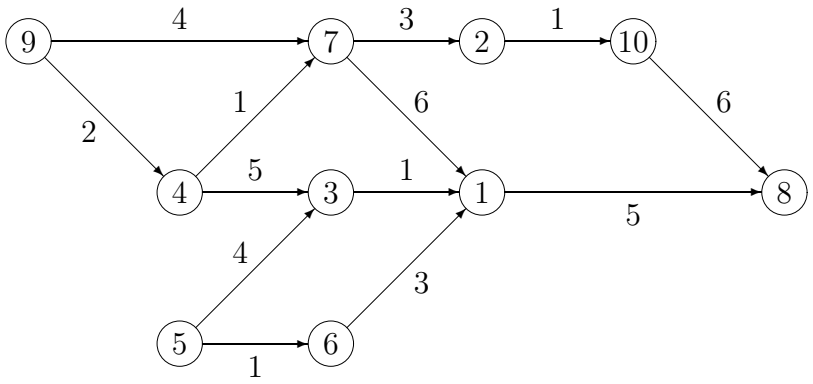
3) **a.** Wat is een topologische ordening op de knopen in een gerichte graaf?

b. Bij topologisch sorteren van een gerichte, acyclische graaf zoeken we steeds naar een knoop zonder inkomende takken. Toon aan dat zo'n knoop altijd bestaat in een gerichte, acyclische graaf met n knopen.

c. Gegeven een gerichte, acyclische graaf met n knopen en m takken in een adjacency-list representatie. Geef een (pseudo-code) algoritme met tijdscomplexiteit $\mathcal{O}(m+n)$ om deze graaf topologisch te sorteren. Je mag aannemen dat eventueel benodigde stapels en rijen reeds geïmplementeerd zijn.

d. In onderstaande gerichte, acyclische graaf representeren de takken projecten. Een project (i, j) (corresponderend met de gerichte tak van knoop i naar knoop j) kan pas worden uitgevoerd als alle projecten eindigend in knoop i zijn afgelopen. De gewichten op de takken zijn de benodigde tijdsduren van de afzonderlijke projecten.

Bereken op een systematische manier, met behulp van een topologische ordening van de graaf, het vroegste tijdstip waarop alle projecten afgerond kunnen zijn, ervanuitgaand dat we op tijdstip 0 met de eerste projecten kunnen beginnen. Laat duidelijk zien hoe je aan je (tussen-)resultaten komt.



4) **a.** Geef van elk van de volgende reguliere expressies de structuur mbv. een boom weer. Geef ook de corresponderende postfix notaties.

i. `(B*B+A*)*` **ii.** `(A+B)*(AA+B+BAA)`

b. Construeer voor de expressie `(B*B+A*)*` op systematische manier de bijbehorende eindige automaat. Nummer de knopen van deze automaat in de volgorde waarin ze tijdens de postorde evaluatie van de boom aan de automaat worden toegevoegd.

Teken nogmaals de boom die bij de expressie hoort, en geef bij elke knoop van de boom de nummers van de corresponderende begin- en eindknoop van de eindige automaat.

c. Laat zien dat de string '`BBAB`' aan de expressie `(B*B+A*)*` voldoet. door het letter-voor-letter uitvoeren van een complete wandeling door de automaat. Geef het zoekproces schematisch in een vertakkende 'boom'structuur weer. Hierbij dienen ook 'doodlopende' paden opgenomen te zijn.

d. Bovenstaand zoekproces wordt uitgevoerd in een graaf waarin een tak niet altijd voorzien is van een letter. Neem nu aan dat in de graaf élke tak een latter draagt, terwijl daarentegen het aantal uitgaande takken van een knoop niet beperkt is tot maximaal twee. Beschrijf nauwkeurig het algoritme dat voor een gegeven string S bepaalt of er een pad is van beginknoop naar eindknoop langs de takken.

Welke datastructuren gebruikt u om deze wandeling uit te voeren ?