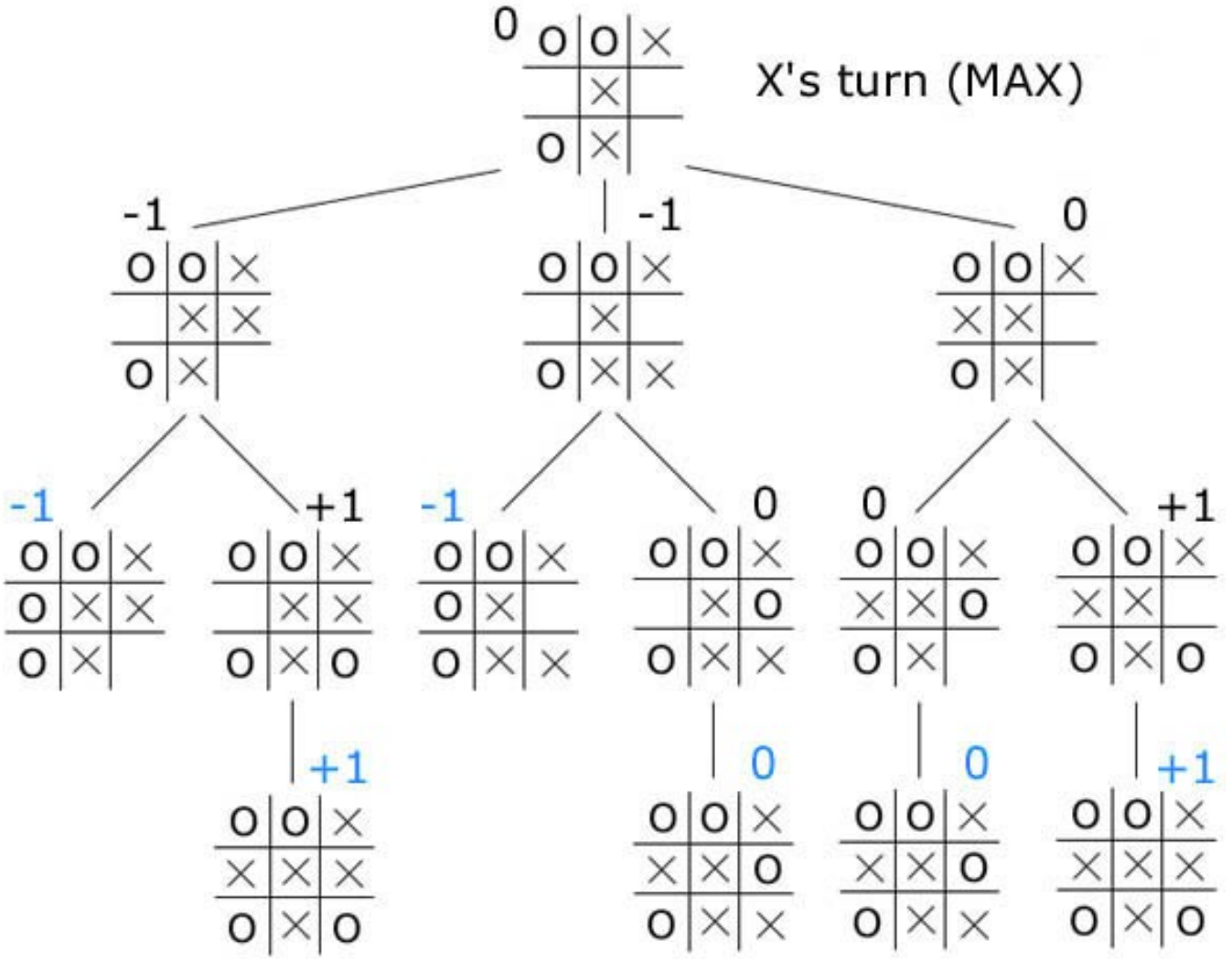
Trær



Et eksempel: Åtterspillet

- To spillere som «trekker» annenhver gang
- I hvert trekk velges et av tallene 1, 2, 3, men *ikke* tallet som motspiller valgte i forrige trekk
- Valgte tall summeres fortløpende
- Hvis en spiller får summen til å bli *lik* 8, vinner han/hun spillet
- Hvis en spiller får summen til å bli *over* 8, taper han/hun spillet

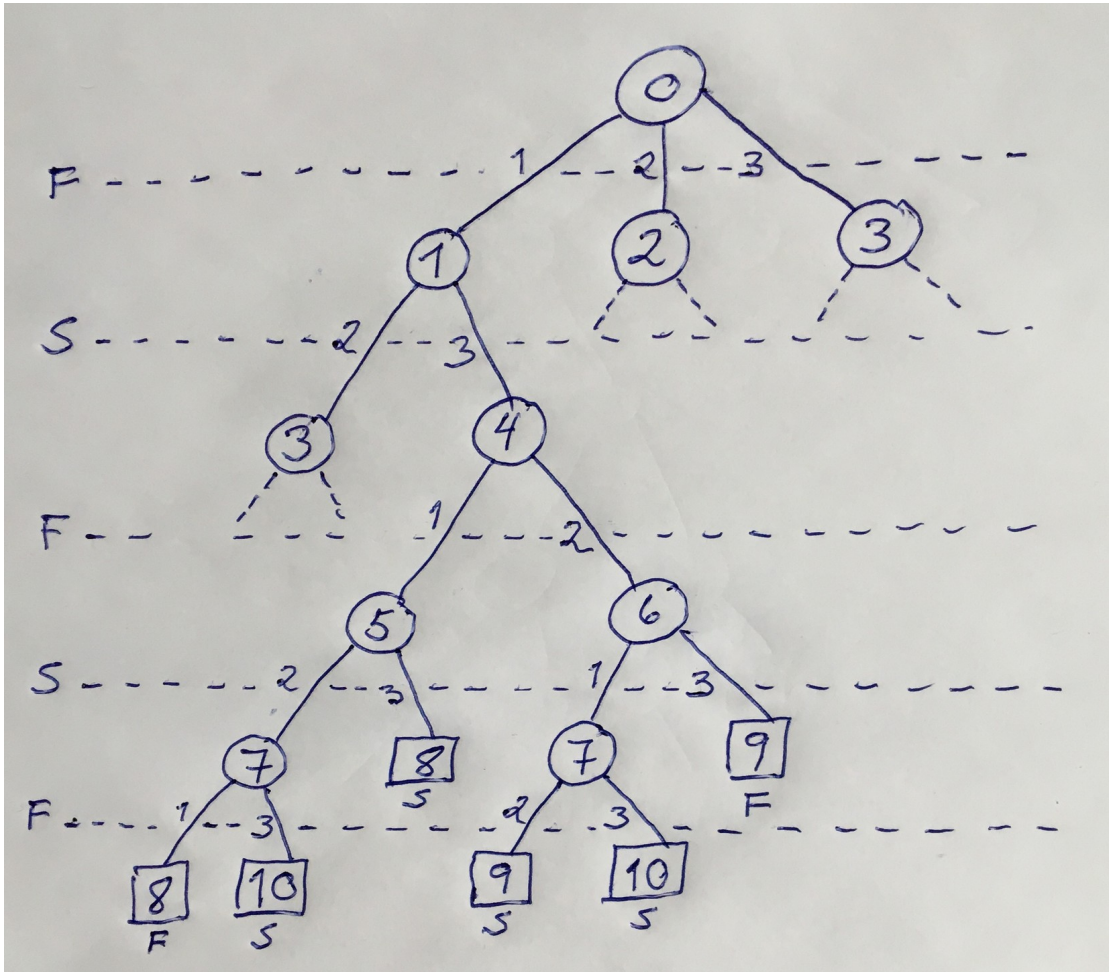
Åtterspillet: Eksempel

F: Spiller 1 (First)
S: Spiller 2 (Second)

F	Sum	S	Sum
1	1	3	4
1	5	2	7
1	8		

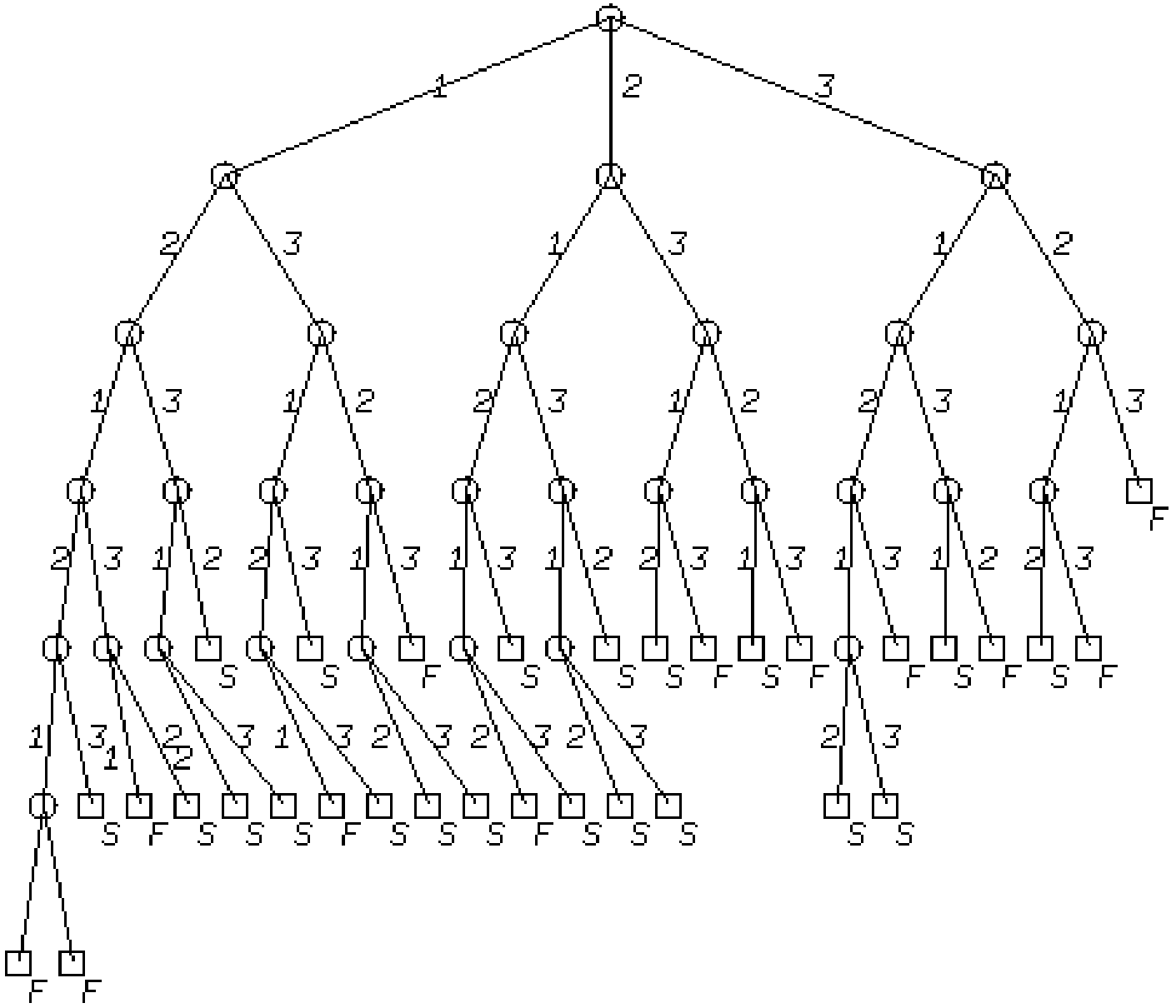
Spiller 1 vinner

Åtterspillet kan tegnes som et *tre*



- Nodene inneholder summen så langt i spillet
- Forbindelser mellom nodene er mulige trekk

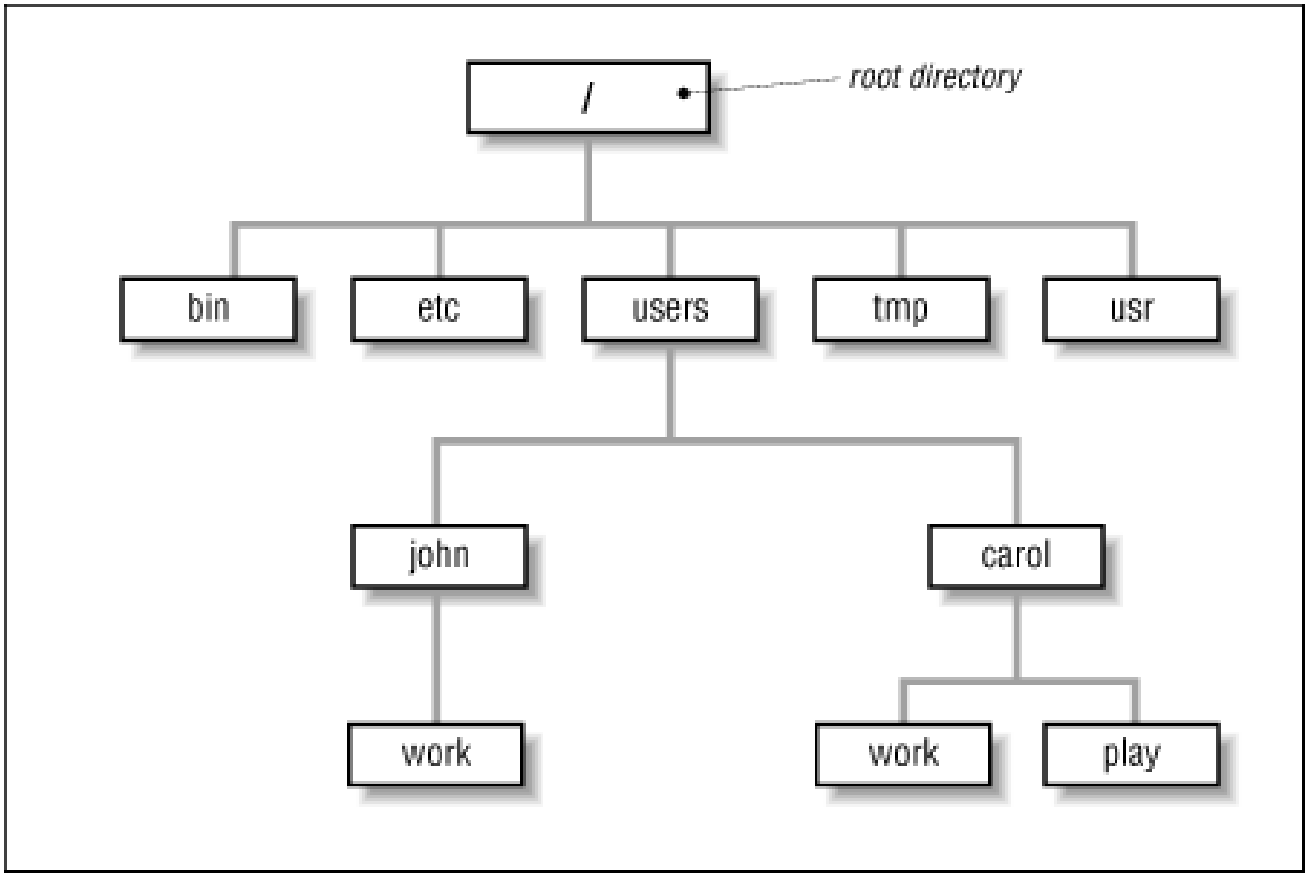
Et spilltre for *hele* åtterspillet



Trær og *hierarkiske* data

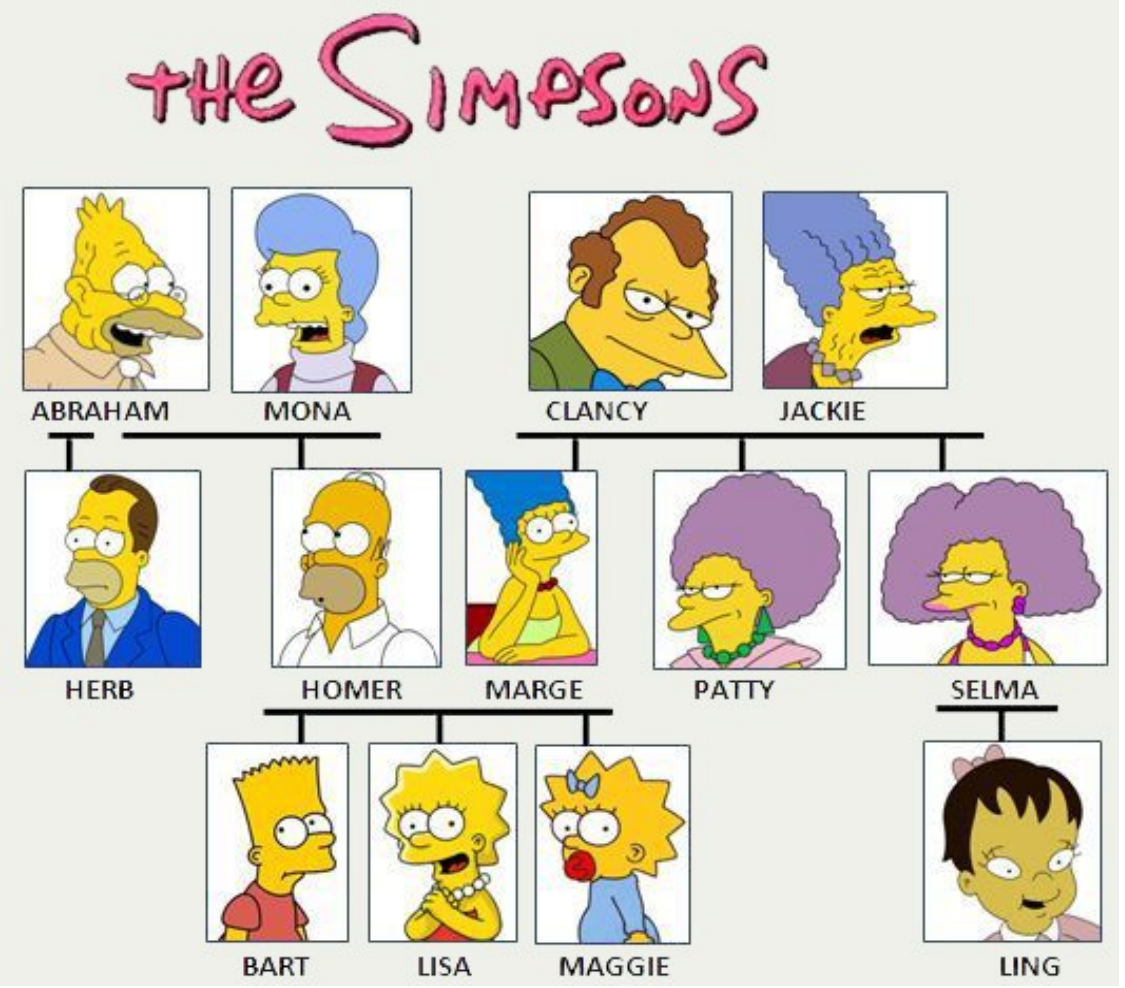
- Trestrukturer er velegnet for å representere data som er ordnet innbyrdes i et nivå- eller verdi-system
- Dataene i treet ligger i lag eller *nivåer,* som tilsvarer en hierarkisk ordning
- Nodene kan bare ha én direkte *forgjenger* på nivået over i treet, men flere direkte *etterfølgere* på nivået under
- På øverste nivå er det bare én node, *roten* i treet, som er den eneste som ikke har noen direkte forgjenger
- Maksimalt antall direkte etterfølgere som hver node kan ha, bestemmer *typen* av tre

Eksempel: Katalogtrær i filsystemer

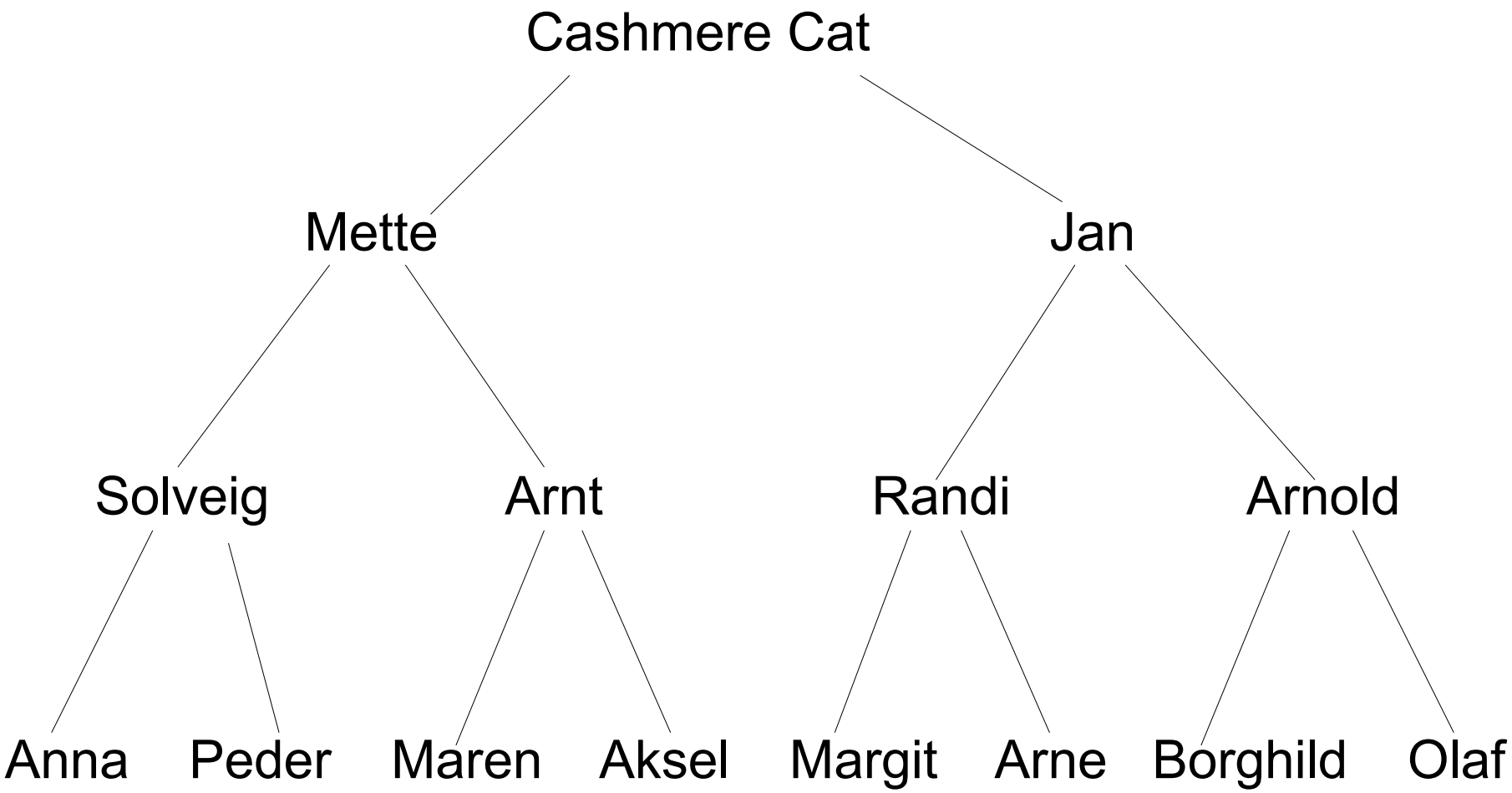


Eksempel: Turneringstrær

Eksempel: Familietre



Eksempel: Foreldretre (binært)

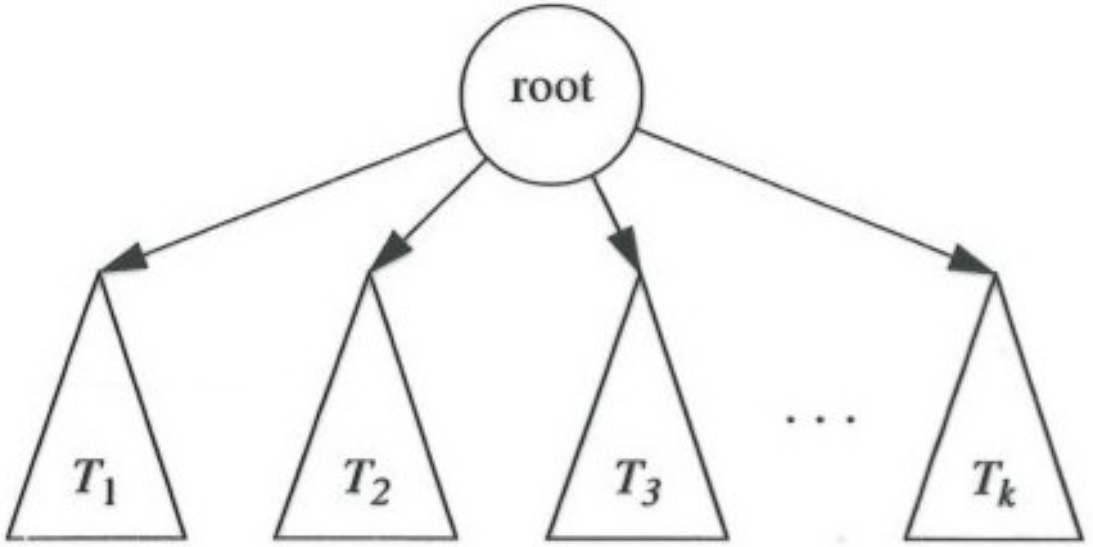


Trær av orden *n*

- Trær klassifiseres etter hvor mange *etterfølgere* en node *maksimalt* kan ha
- Et tre der nodene har maksimalt *n* etterfølgere kalles for et tre av *orden n,* eller et *n*-ært tre
- Trær av orden 2, *binære* trær, er vanlig å bruke i algoritmer for effektiv datahåndtering
- Store databasesystemer bruker ofte trær av svært høy orden
- Trær som *ikke* har et begrenset antall etterfølgere til hver node kalles *generelle* trær

Rekursiv definisjon av tre av orden *n*

- Et tre av orden *n* er en datastruktur som enten er:
  - Tom (inneholder 0 noder), eller
  - Består av en *rot*node med maksimalt *n* etterfølgere som alle er trær av orden *n*

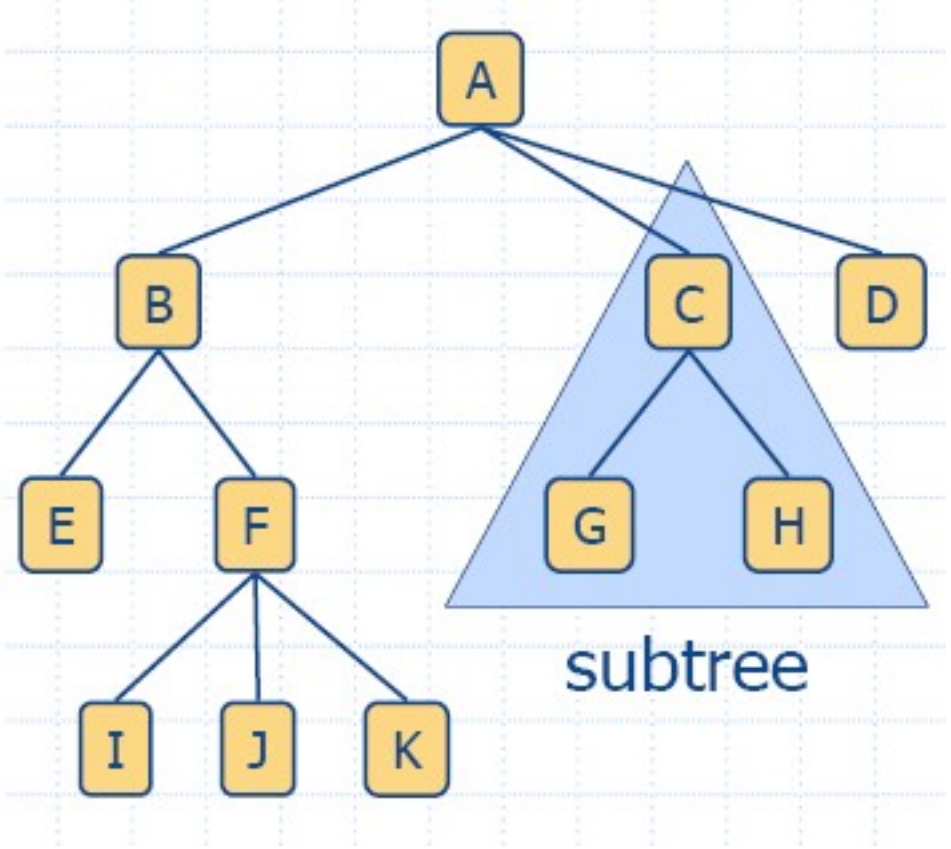


Betegnelser på nodene i et tre

- **Rot** (root):
  - Den ene noden på øverste nivå som ikke har noen forgjenger – inngangen til hele treet
- **Barn** (child/children):
  - Noder som er *direkte* etterfølgere til en node på nivået over, er *barn* av denne noden
- **Forelder** (parent):
  - Noden på nivået over, som er direkte forgjenger til en node (barnet), er forelder til denne noden
- **Søsken** (siblings)
  - Noder som har samme forelder er søsken

Betegnelser på nodene i et tre (forts.)

- **Blad** (leaf /leaves):
  - Node som ikke har noen barn, terminalnoder
- **Indre node** (internal node):
  - Alle noder, unntatt rotnoden, som har *minst* ett barn, er indre noder
- **Kant** (edge):
  - Et node*par* som angir en *forbindelse* mellom to noder, fra forelder til barn



- Rot: A
- Blader: E, I, J, K, G, H, D
- Indre noder: B, F, C
- Barn/forelder/søsken:

  A → B, C, D

  B → E, F

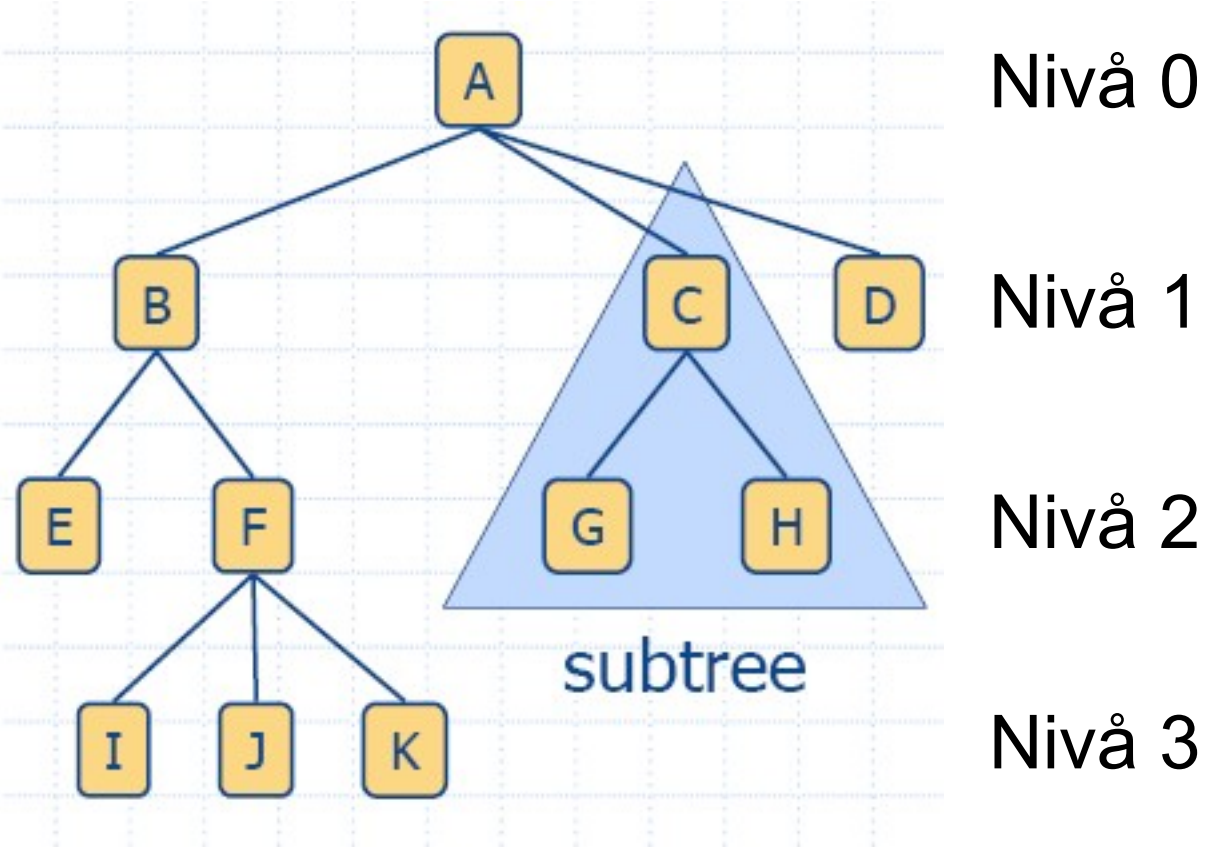
  F → I, J, K

  C → G, H
- Alle kanter i treet:

  (A,B), (A,C), (A,D),
  (B,E), (B,F), (C,G),
  (C,H), (F,I), (F,J), (F,K)

Vei, nivå, lengde og høyde

- Trær aksesseres fra roten og *nedover* i treet, ved hele tiden å gå fra foreldre til barn
- Rotnoden er på nivå 0, barna til roten på nivå 1, etc.
- En **vei** i treet er listen av noder som vi går gjennom for å komme fra en bestemt node til annen node lenger ned i treet, nodene på veien er **etterfølgere**
- **Lengden** av en vei er antall kanter på veien
- **Høyden** av et tre er lengden av den *lengste* veien som finnes i hele treet fra roten til et blad



- Vei: A, D, lengde 1
- Lengste vei: A, B, F, I, lengde 3
- Høyde: 3