Distribuované systémy a výpočty

X36DSV

Jan Janeček
(dnes Peter Macejko)

Ukončení výpočtu (Termination Detection)

Terminal configuration

- terminal state

Termination

- implicit (message) x explicit (process)

Process

- active x passive
  - active → passive – only in an internal event
  - ? → active – when message is received
  - Terminal configuration = ???

Ukončení výpočtu

Dijkstra - Scholten

Proměnné - legenda
- Defin – deficit na vstupu
- Defout – deficit na výstupu
- Others – seznam dalších žádajících procesů

Zprávy - legenda
- MESSAGE = žádost, SIGNAL = odpověď

```
begin                                          { inicializace }
   Defin:=0; Defout:=0; Others:=Ø
end

receiving MESSAGE from j do                    { příjem žádosti aplikace }
  begin
    if DefIn=0
    then Parent := j
    else Others := Others+j;
    DefIn := DefIn+1
  end
```

Ukončení výpočtu

Dijkstra - Scholten

```
receiving SIGNAL from j do                    { příjem odpovědi aplikace }
  DefOut:=DefOut-1;

sending MESSAGE to j do                       { odeslání žádosti aplikace }
    { possible if DefIn>0 }
    DefOut := DefOut+1;

sending SIGNAL to (Oth=any of Others) do      { odeslání odpovědi aplikace }
    { possible if (DefIn>1 }
    begin
        Others := Others-Oth;
        DefIn := DefIn-1
    end

sending SIGNAL to Parent do                   { odeslání odpovědi aplikace }
    { possible if (DefIn=1 and DefOut=0) }
    DefIn := DefIn-1
```

Ukončení výpočtu

Dijkstra – Feijen – Van Gasteren

Proměnné - legenda
State – stav procesu
Color – „barva“ procesu
TPresent – informace o vlastnictví tokenu
TColor – barva přijatého tokenu

```
begin                                   { inicializace }
   TPresent := F; Color := WHITE
end

receiving MESSAGE do                    { příjem zprávy aplikace }
   State := ACTIVE

waiting MESSAGE or State=TERMINATED do
   State := PASSIVE                     { čekání na zprávu aplikace }

sending MESSAGE to j begin              { odeslání zprávy procesu s indexem j>i }
   if i<j then Color := BLACK
```

Ukončení výpočtu

Dijkstra – Feijen – Van Gasteren

```
when received TOKEN(ct) from i+1 do     { příjem zprávy TOKEN }
   begin
      TPresent := T;
      TColor := ct;
      if i=0 then
         if Color=WHITE and TColor=WHITE
         then { TERMINATION DETECTED }
         else TColor := WHITE
   end

when TPresent and State=PASSIVE do
   begin                                { předání zprávy TOKEN následníkovi }
      if Color=BLACK then TColor := BLACK;
      TPresent := F;
      send TOKEN(TColor) to i-1;
      Color := WHITE
   end
```

Ukončení výpočtu

Misra

```
begin                                              { inicializace }
   Color := BLACK; TPresent := F; nb := 0
end

when waiting MESSAGE do                  { čekání na zprávu aplikace }
   State := PASSIVE

when received TOKEN(j) do                { příjem zprávy TOKEN }
   begin
      nb := j;
      TPresent := T;
      if nb=Size(C) and Color=WHITE then
         { TERMINATION DETECTED }
   end
```

Ukončení výpočtu

Misra

```
when received MESSAGE do          { příjem zprávy aplikace }
   begin
      State := ACTIVE;
      Color := BLACK
   end

when TPresent and State=PASSIVE   { odeslání zprávy TOKEN }
   begin
      if Color=BLACK then nb := 0
      else nb := nb+1;
      send TOKEN(nb) to Succesor(C,i);
      Color := WHITE;
      TPresent := F
   end
```

Ukončení výpočtu

Misra

```
when TPresent and State=PASSIVE        { odeslání zprávy TOKEN }
   begin
      if Color=BLACK then nb := 0
      else nb := nb+1;
      send TOKEN(nb) to Succesor(C,i);
      Color := WHITE;
      TPresent := F
   end

begin                                  { inicializace }
   Color := BLACK; TPresent := F; nb := 0
end
```

Ukončení výpočtu

Rana

Proměnné - legenda

state – stav procesu
LC – lokální logický čas
unack – počet nepotvrzených zpráv
QT – čas posledního přechodu do *quiet*

begin { inicializace }
state:=active; LC:=0; unack:=0; QT:=0;
end

{state = active} { poslání zprávy }
LC:=LC+1; send MSG(LC); unack:=unack+1;

{receive MSG(FC) from q to p} { příjem zprávy }
LC:=max(FC,LC)+1;
send ACK(LC) to q;
state:=active;

Ukončení výpočtu

Rana

```
{state = active}                          { poslání zprávy }
    LC:=LC+1; state:=passive;
    If (unack = 0) then {p becomes quiet}
        begin
            QT:=LC;
            send TKN(LC,QT,p) to next(p);
        end

{receive ACK(FC) to p}                    { příjem potvrzení }
    LC:=max(FC,LC)+1; unack:=unack-1;
    If (unack = 0) and (state = passive) {p becomes quiet}
        begin
            QT:=LC;
            send TKN(LC,QT,p) to next(p);
        end
```

Ukončení výpočtu

Rana

```
{receive TKN(FLC,FQT,q) to p}        { příjem tokenu }
    LC:=max(FLC,LC)+1;
    if (p is quiet) then
        if (p = q) then Announce
        else if FQT >= QT then
            send TKN(LC,FQT,q) to next(p);
```

Replikace

Performance enhancement

- concurrent read-only accesses
- sequential updates

Enhanced availability

$1 – p = 1 - p^n$

Fault tolerance

$t < n/2$ - fail stops

$b < n/3$ - Byzantine failures

Replikace - DB

Database

- no concurrent read or write on data
- quorum-based replica control
- each copy of data has a vote – total V

V_r – read quorum V_w – write quorum

$V_r + V_w > V$ $V_w > V/2$

Quora - definitions

set of nodes/sites

$S = \{ s_1, s_2, \ldots, s_n \}$

coterie

$C = \{ Q_1, Q_2, \ldots, Q_n \}, \; Q_i \neq \emptyset \wedge Q_i \subseteq S$

quorum

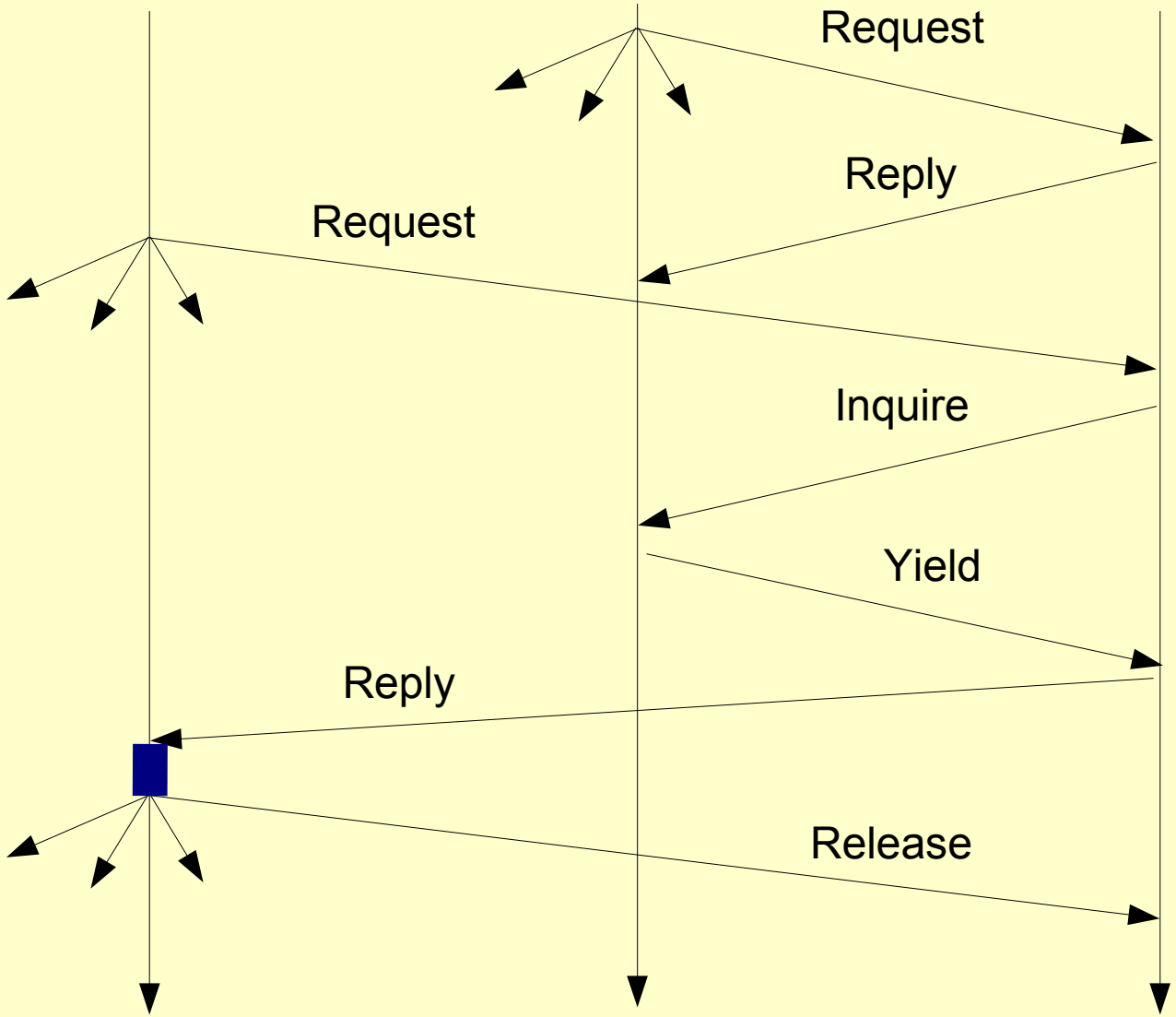
$Q_i \cap Q_j \neq \emptyset, \; Q_i, Q_j \in C, \; i \neq j$ - intersection property

$Q_i \not\subset Q_j, \quad Q_i, Q_j \in C, \; i \neq j$ - minimality property

Quora - algorithm

Maekawa algorithm

(mutual exclusion)



Quora - measurement

Load

- *L(Q)* – the probability that the busiest server is in use under an optimal strategy of accessing the servers

Resilience

- R*(Q)* – maximal number of failing nodes such that, there is a Quorum with no failing node

Failure probability

- $F_p(Q)$ – the probability that at least one server of every quorum fails

Quora

Singleton quorum

quorum size = 1

Majority quorum

quorum size = (n+1)/2

Grid quorum

quorum size = $2(\sqrt{n}) - 1$

Maekawa's quorum

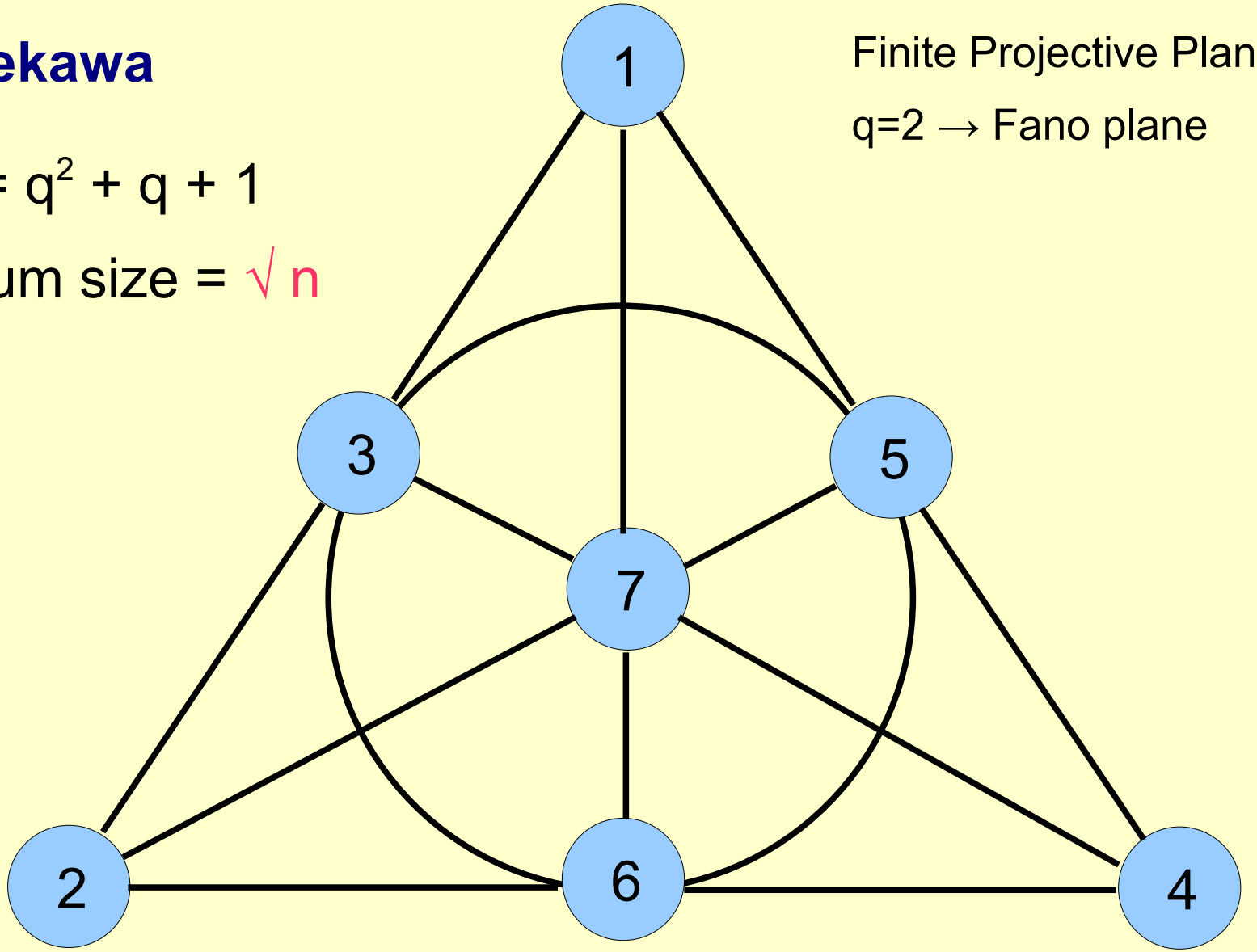
Tree quorum (Agrawal, El Abbadi)

Quora

Maekawa

$n = q^2 + q + 1$

quorum size = $\sqrt{n}$

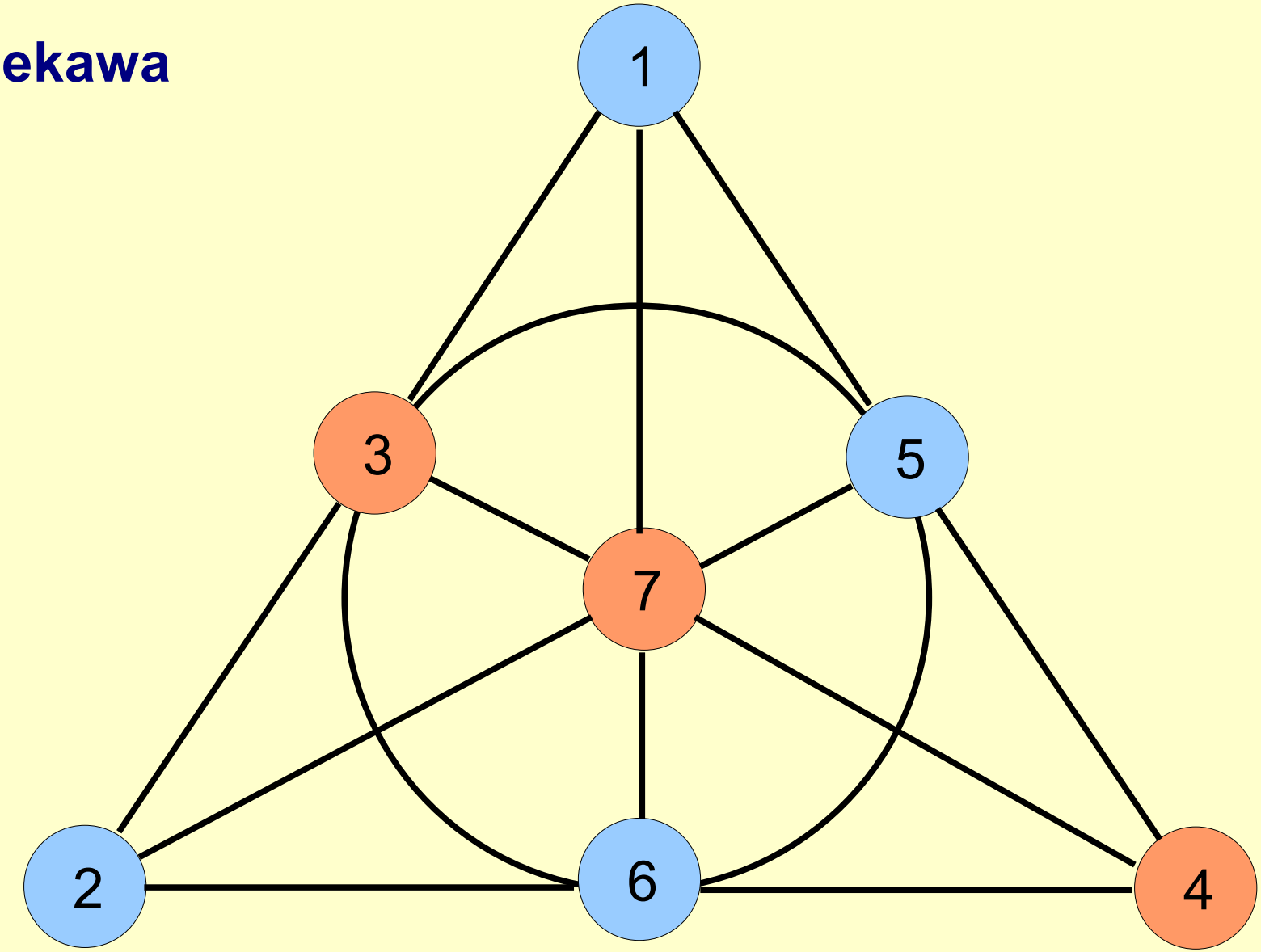


Finite Projective Planes

q=2 → Fano plane

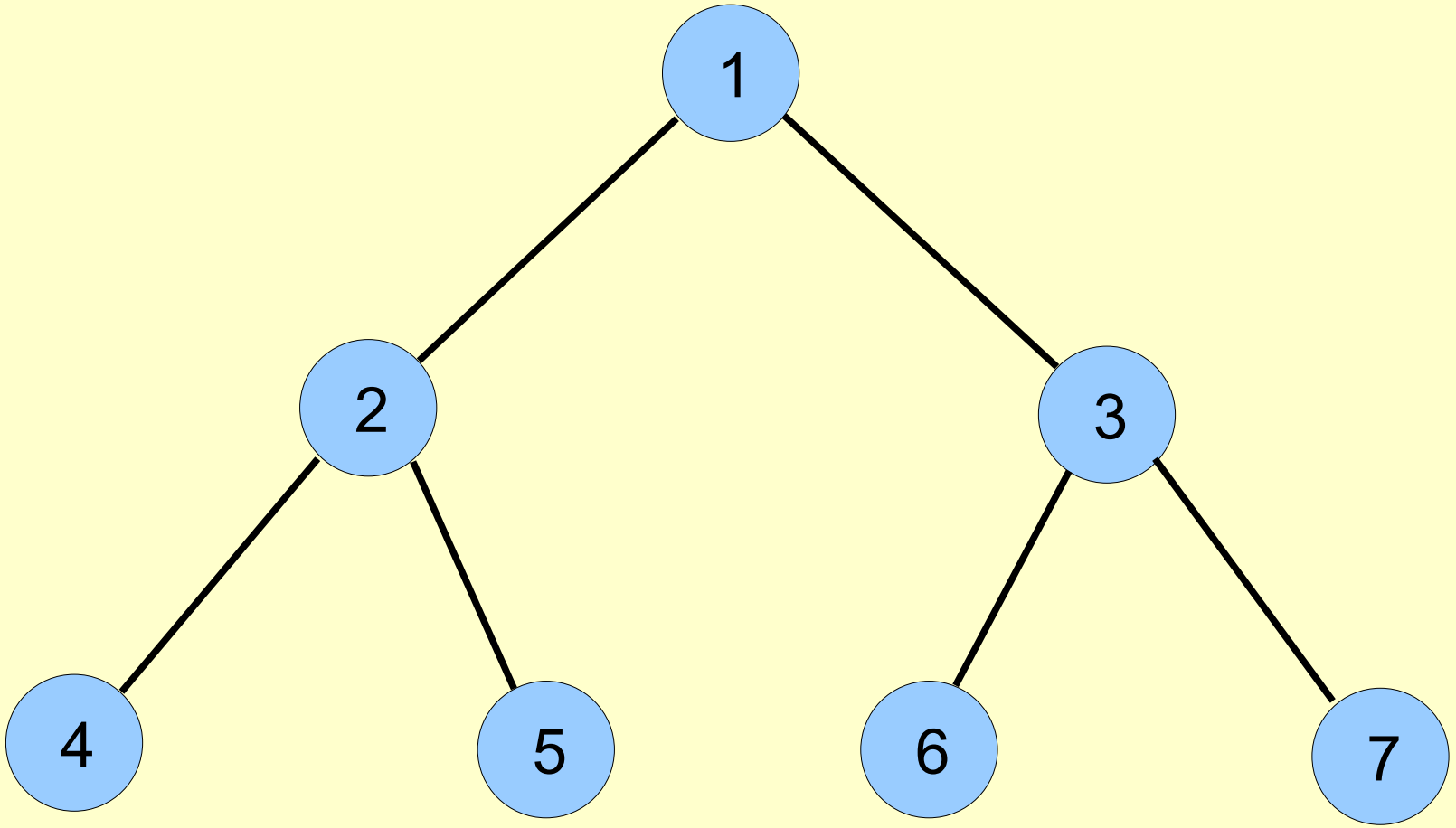
Quora

Maekawa



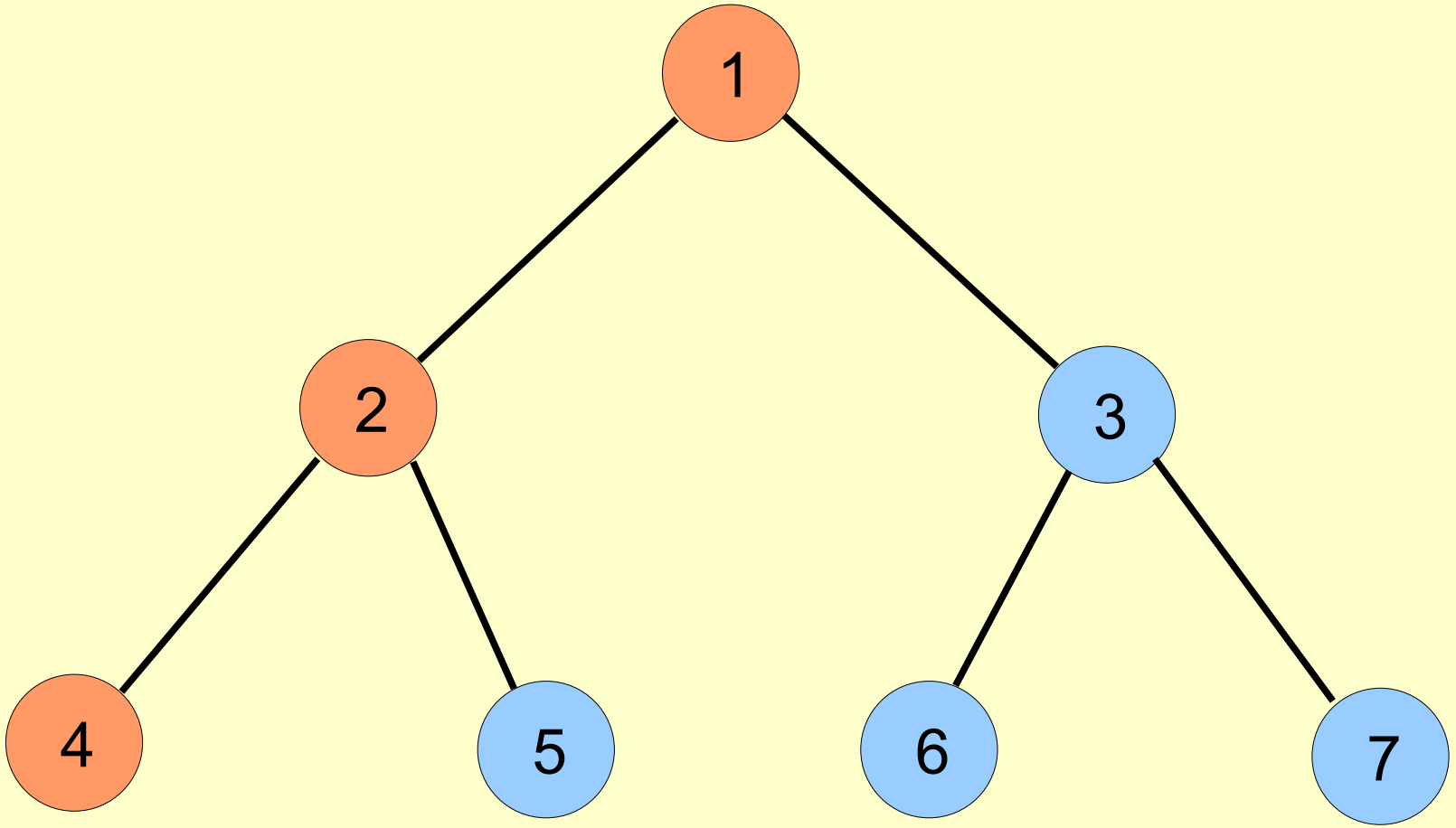
Quora

Tree



Quora

Tree

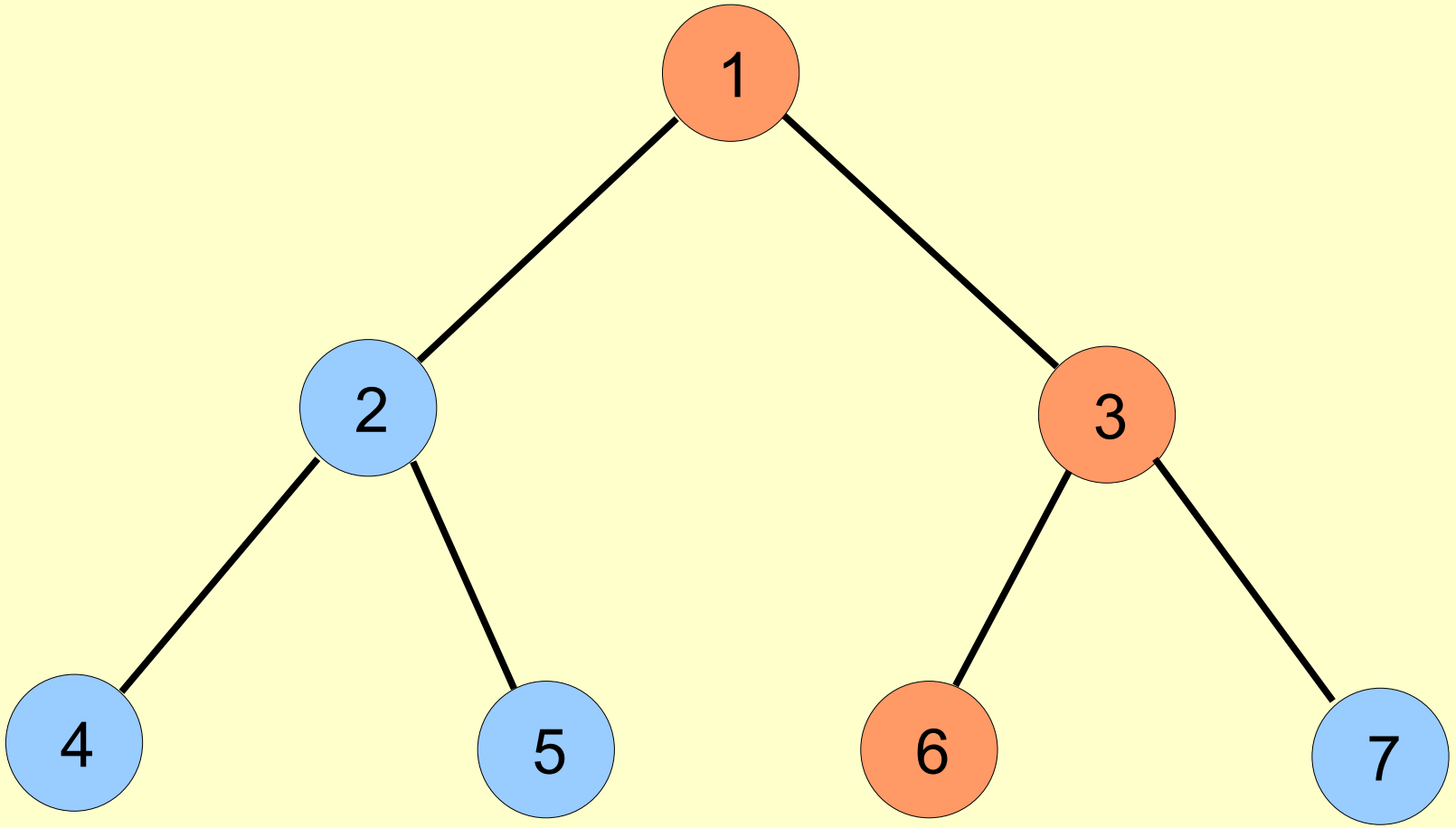


Quora

Tree – Agrawal, El Abbadi

```
function GetQorum (Tree:tree) : quorumset;
var left, right : quorumset;
begin
   if Empty(Tree) then
      return ({});
   else
      if GrantsPermission(Tree^.Node) then
            return ({Tree^.Node} U GetQuorum(Tree^.LeftChild))
         or
            return ({Tree^.Node} U GetQuorum(Tree^.RightChild))
      else
         left = GetQuorum(Tree^.left);
         right = GetQuorum(Tree^.right);
         if (left = Ø and right = Ø ) then
               (* Unsuccessful in establishing a quorum *); exit(-1);
         else
               return(left U right};
         end; end;
end.
```

Quora



quorum size = $\log_2 n$ . . . $(n+1)/2$

Quora

Tree