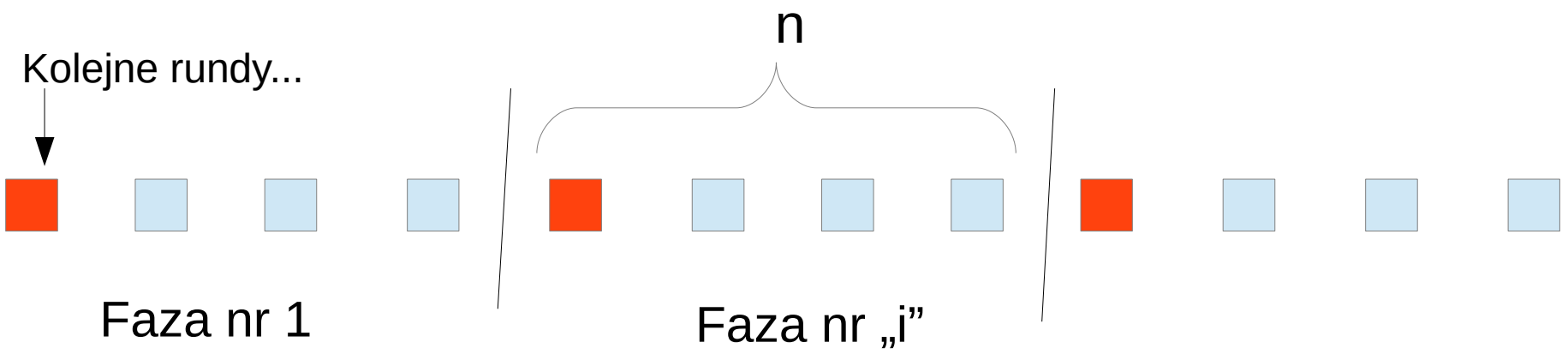
LE, synch, nie-jednorodny, O(n)-kom



Nie-jednorodny <=> wierz znają „n”

Algorytm:

W 1 rundzie każdej fazy wierz,

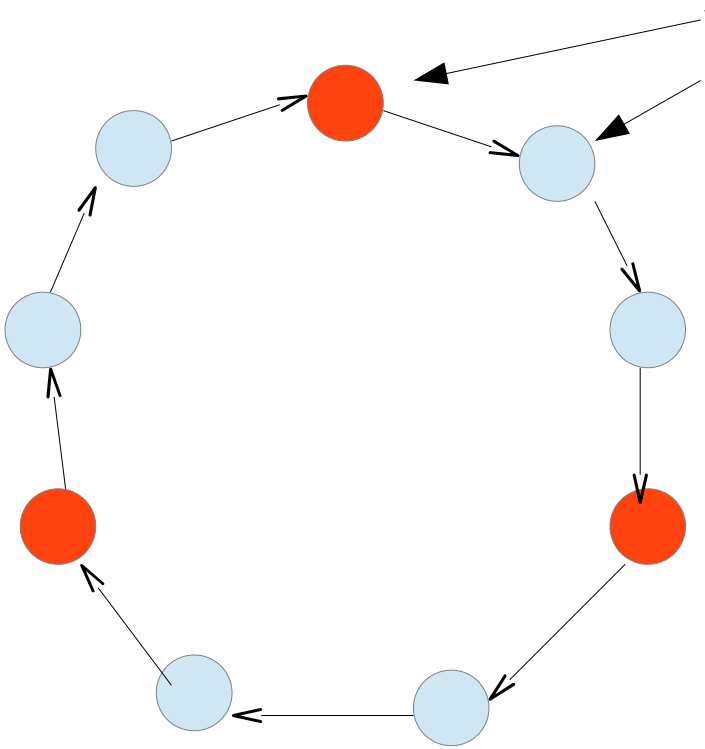
każdy wierz sprawdza, czy jego ID = nr fazy;

jeśli tak jest, to ogłasza się *liderem*

oraz zawiadamia wsz inne wierz że NIE są liderami

Liczba kom = n, czas działania = n * min ID

LE, synch, jednorodny, O(n)-kom



Prędkość komunikatu = 1/k
<=> 1 skok na k rund...

Wierz uczestniczące wysyłają swój ID
z predkością = 1,
aż do napotkania drugiego uczestn wierz;
potem wierz porusza się
z prędkoścIa = 1/ (2^ID)

Zasada przekazywania komunikatów:
Kom jest „połykany” gdy jego ID jest większy
niż min ID który dany wierz widział do tej pory,
w przeciwnym wypadku jest przekazywany dalej
Różnica między uczest a przekaźnikami:
uczest: widzą swój ID
przekaźniki: nie widzą...

Tylko min ID wierz uczestn wykona pełne koło,
wróci do źródłowego wierz i ogłosi się liderem...
(ID przekaźników nie mają znaczenia)

LE, synch, jednorodny, O(n)-kom

Oszacujemy liczbę komunikatów
w „fazie I” (prędkość=1) oraz w „fazie II” (prędkość=1/2^ID)

Przez ile rund wysyła się jakieś komunikat ?
odp: n + 2^i * n, gdzie i = min ID wierz uczest

Szacujemy liczbę komunikatów w fazie II:
jest to suma po „j” ϵ „zbioru ID wierz uczestn” ...

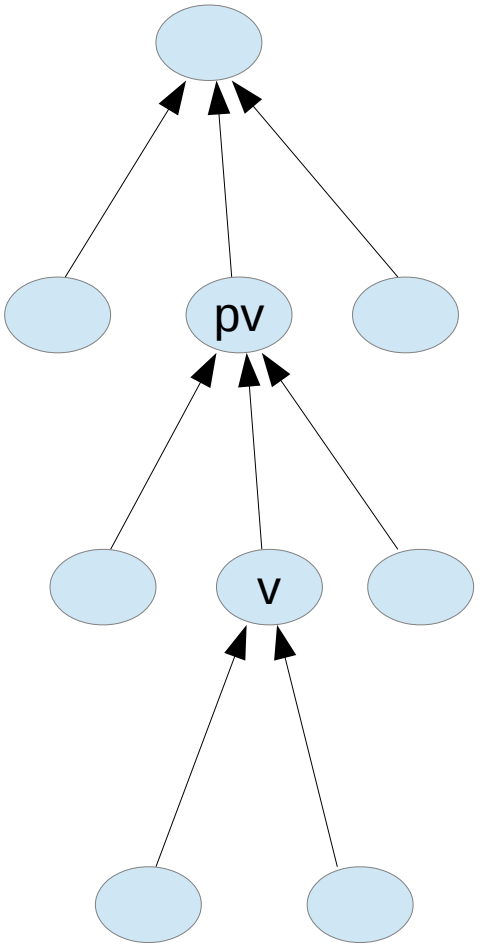
$$\Sigma_j \frac{\left(2^i+1\right)*n}{2^j}$$

Ta suma jest min gdy ID wierz uczestn ϵ {0, 1, 2, … }

$$\Sigma_j \frac{\left(2^i+1\right)*n}{2^j} \leq \Sigma_{j=0}^{n-1} \frac{\left(2^0+1\right)*n}{2^j} = 4*n$$

W fazie I wysyła się „n” kom,
zatem łącznie, w obu fazach, wysyła się „5*n” kom...

8-kolorowanie wierz drzewa ukorzenionego Cole & Vishkin, synch, czas= O(log^*n)



log^*n = „ile razy trzeba zlogarytmować n aby zejść poniżej 1”
chodzi o **prawidłowe** kolorowanie wierz,
czyli wierz połaczone kraw mają RÓŹNE kolory...

Algorytm:

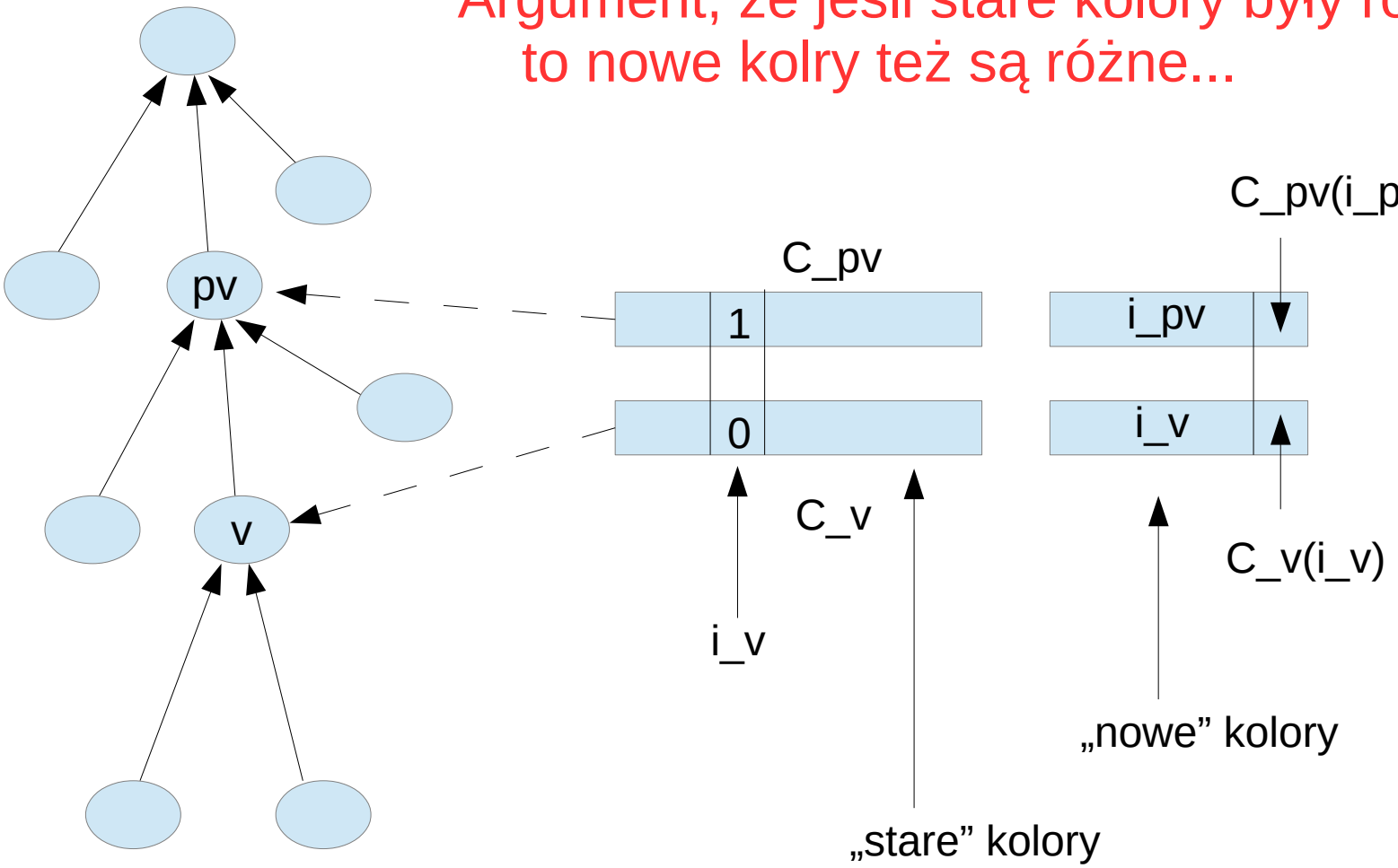
1. C_v : = ID_v
2. powtórz O(log^*n) razy:
   i_v:= pozycja bitu, na którym różnia się C_v i C_pv
   # gdzie C_pv to kolor parenta wierz v
   C_v:= (bity(i_v), C_v(i_v))
   # bity(x) = reprezentacja dwójkowa liczby x
   # C_v(k) = k-ty bit C_v

L_k = liczba bitów C_v w k-tej iteracji
L_0 = O(logn)
L_{k+1} = sufit(log_2 L_k) + 1
Jest to ciąg malejący zbieżny do 3.
Jeśli kolor jest zpisany na 3 bitach, to liczba kolorów=8

Dlaczego po zakończeniu algrytmu mamy
prawidłowe kolorowanie wierz ? (8-ma kolorami)

8-kolorowanie wierz drzewa ukorzenionego Cole & Vishkin, synch, czas= O(log^*n)

Argument, że jeśli stare kolory były różne,
to nowe kolry też są różne...



8-kolorowanie wierz drzewa ukorzenionego Cole & Vishkin, synch, czas= O(log^*n)

Można w prosty sposób
zmniejszyć liczbę kolorów
z 8 do 3 ...

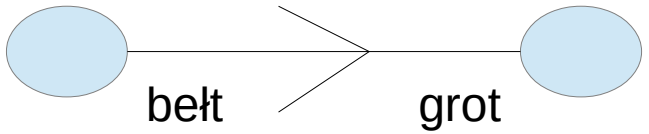
W jaki sposób uogólnic
algorytm C&V
na inne grafy ???
(np. dla cyklu niezorient ??)

(Delta+1)-kolorowanie wierz, w grafie dowolnym, synch, czas= O(Delta^2 + log*n)

Def „forest decomposition”:
Mamy graf G=(V,E)
1. orientujemy kraw wg ID
2. każdy wierz numeruje sąsiadów
3. każdej kraw przypisujemy nr
   na „bełcie strzały”

Otrzymujemy podział:
E=E_1 u … u E_Delta
E_i = zb kraw którym przyp „i”

Fakt: (V,E_i) jest lasem ukorz drzew
Dowód:
1. z wierz nie wychodzi >1 strzał
2. nie ma skierowanych cykli



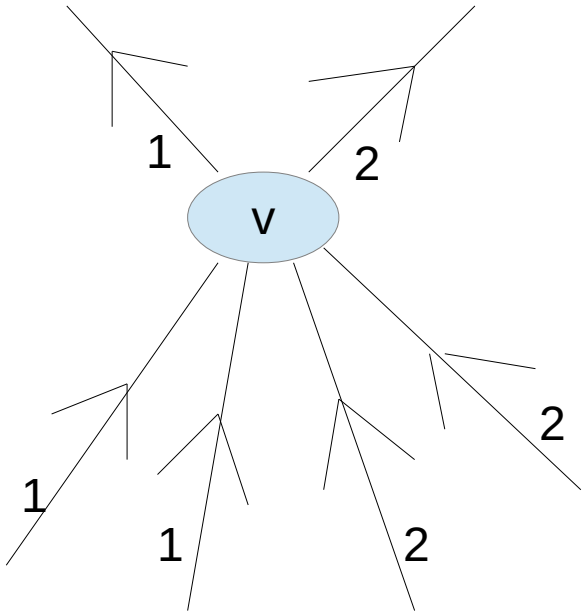
Algorytm:
1. oblicz forest decomp
2. w drzewach ukorze wsz E_i
 oblicz 8-kol-wierz metodą C&V
3. A:= puste
 For i=1 to Delta do
  (1) w G[A u E_i] mamy
  8*(Delta+1)-kol-wierz
  pochodzące z połączenia
   (Delta+1)-kol-wierz w A
   z 8-kol-wierz w E_i
  (2) redukujemy je do
  (Delta+1)-kol-wierz
  (3) A:=A u E_i

Po zakonczeniu pkt 3,
 mamy w calym grafie
  (Delta+1)-kol-wierz...

(Delta+1)-kolorowanie wierz, w grafie dowolnym, synch, czas= O(Delta^2 + log*n)

Co jeszcze trzeba wyjaśnić w tym algorytmie ???

Łączenie kolorowania
w 2 podgrafach rozł kraw



W E_1 mamy kolor C^1_v
W E_2 mamy kolor C^2_v
w całym grafie mamy kolor
(C^1_v, C^2_v)

Redukcja liczby kolorów
do Delta+1

W palecie Delta+1 kolorów
zawsze jest 1 kolor wolny...

Można iterować po „starych kolorach”;
Dla wsz wierz bieżącego koloru,
  obliczać nowe kolor,
    z palety Delta+1 kol...