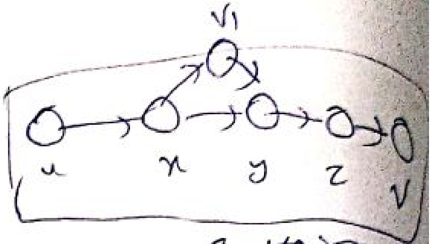
Dt 17/05/2020 Assignment - 2 (PDS) - Module 3

Name - Atul Kumar Agrawal, Branch → CSE - I
Roll No - 1602040031, VSSUT, BURLA.

Lemma 4.1 :- Let u, v be in V, if a path from u to v exists in G, then there exists a simple path that is optimal,

Proof :-

given, $V = \{u, v\}$



now,

Let S_0 = lowest cost path (P) from u to v.

i.e for every simple path P', from u to v,

$C(S_0) \le C(P')$,

Hence, $C(S_0)$ is lower bound for cost of path.

now, $V = \{v_1, v_2 \dots \dots v_N\}$

→ By successfully eliminating cycles from P, there exists a simple path P' from u to v where $C(P') \le C(P)$,

Let $P_0 = P$.

for i = 1 to N, construct P_i path as follows:-
if v_i occurs at most once in P_{i-1},
then $P_i = P_{i-1}$.

else,

$P_{i-1} = \{u_0, \ldots\ldots u_k\}$

where u_{j1} = 1st

& u_{j2} = last occurrence of v_i in P_{i-1}.

& let $P_i = \{u_0, \ldots\ldots u_{j1} (= u_{j2}), u_{j2+1}, \ldots\ldots u_k\}$

→ By construction, P_i : path from u to v & contains all nodes of $\{v_1 \ldots\ldots v_i\}$ at most once, hence P_N is a simple path from u to v.

→ P_{i-1} consists of P_i & cycle $Q = \{u_{j1}, \ldots\ldots u_{j2}\}$

hence, $\boxed{c(P_{i-1}) = c(P_i) + c(Q)}$.

→ As there are no cycles of -ve weight,

⇒ $\boxed{c(P_i) \leq c(P_{i-1})}$.

hence, $c(P_N) \leq c(P)$.

So, by choice of S_0, $\boxed{c(S_0) \leq c(P_N)}$.

⇒ $\boxed{c(S_0) \leq c(P)}$ (hence, proved)

Theorem 4.2 :- for each $d \in V$ $\exists$ a tree $T_d = (V, E_d)$ such that $E_d \subseteq E$ & such that for each node $v \in V$ the path from 'v' to 'd' in T_d is an optimal path from v to d in G.

Proof :- Let $V = \{v_1, \ldots v_N\}$. We shall inductively construct a series of trees $T_i = (V_i, E_i)$ (for $i = 0, \ldots N$) with the following properties:-

(i) each T_i is a subtree of G, i.e. $V_i \subseteq V$, $E_i \subseteq E$ & T_i is a tree.

(ii) Each T_i (for $i < N$) is a subtree of T_{i+1}.

(iii) for all $i > 0$, $v_i \in V_i$ & $d \in V_i$,

(iv) for all $w \in V_i$, the simple path from w to d in T_i is an optimal path from w to d in G.

→ These properties imply that T_N satisfies the requirements for T_d.

→ To construct the sequence of trees, set $V_0 = \{d\}$ & $E_0 = \phi$.

→ The tree T_{i+1} is constructed as follows. choose an optimal simple path $P = (u_0, \ldots u_k)$ from v_{i+1} to d, & let l be the smallest index such that $u_l \in T_i$ (Such an l exists because $u_k = d \in T_i$; possibly $l = 0$).

Now set

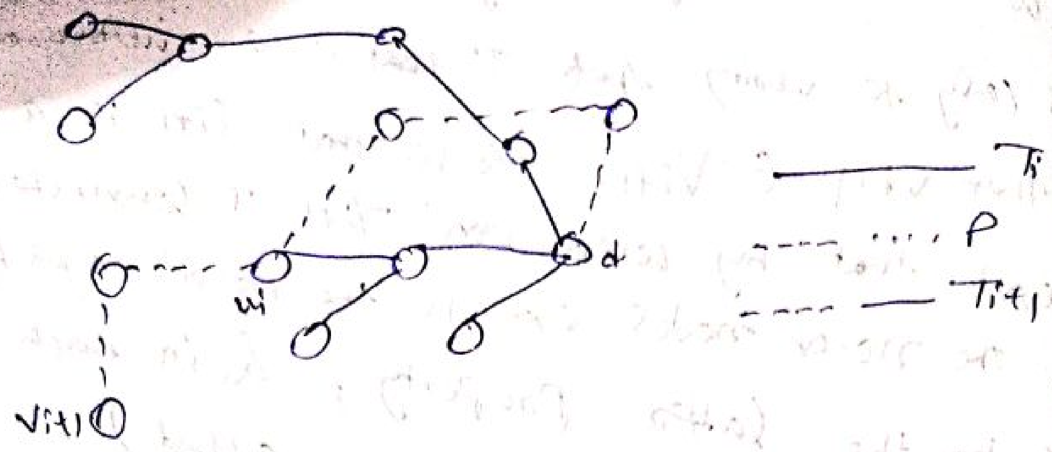
$$V_{i+1} = V_i \cup \{u_j : j < l\} \ \&$$

$$E_{i+1} = E_i \cup \{(u_j, u_{j+1}) : j < l\}.$$

→ (The construction is pictorially represented in fig 4.1)
It is easy to verify that T_i ~~that~~ is a subtree of T_{i+1}
& that $V_{i+1} \in V_{i+1}$. To see that T_{i+1} is a tree,
observe that by construction T_{i+1} is connected,
& the no. of nodes exceeds the no. of edges by one.
(T_0 has the latter property, & in each stage
as many nodes as edges are added).

→ It remains to show that for all $w \in V_{i+1}$,
the (unique) path from w to d in T_{i+1} is an optimal
path from w to d in G; for the nodes $w \in V_i \subset V_{i+1}$
this follows because T_i is a subtree of T_{i+1}; the path
from w to d in T_{i+1} is the same as the path in T_i
which is optimal. Now let $w = u_j$, $j < l$ be a
node in V_{i+1}/V_i. Write Q for the path from u_l to
d in T_i, then in T_{i+1} u_j is connected to d
by the path $\langle u_j \text{---} u_l \rangle$ concatenated with Q, & it
remains to show that this path is optimal in G.
First the suffix $P' = \langle u_l, \text{---} u_k \rangle$ of P is an optimal
path from u_l to d, i.e $C(P') = C(Q)$; the
optimality of Q implies $C(P') \geq C(Q)$; & $C(Q) < C(P')$
implies (by additivity of path costs) that path
$(u_0, \text{---} u_l)$ concatenated with Q has lower
cost than P, contradicting the optimality of P.
Now assume that a path R from u_j to d has
lower cost that path $(u_j, \text{---} u_l)$ concatenated
with Q. Then, by the prev. observation, R has a
lower cost than the suffix $(u_j \text{---} u_l)$ of P, &
this implies (again by additivity of path costs) that the
path $(u_0 \text{---} u_j)$ concatenated with R has lower cost
contradicting optimality of P.

Scanned with CamScanner



Lemma 4.3 :- The forwarding mechanism delivers every packet at its destination iff the routing tables are cycle-free,

Proof :- If the tables contain a cycle for some destination d, a packet for d is never delivered if its source is a node in the cycle.

→ Assume the tables are cycle-free, & let a packet with destination d (& source u_0) be forwarded via $u_0, u_1, u_2, ---$ if the same node occurs twice in this sequence, say $u_i = u_j$, then the tables contain a cycle, namely $(u_i, --- u_j)$ contradicting the assumption that the tables are cycle-free. Thus, each node occurs at most once, which implies that this sequence is finite, ending, say in node u_k $(k<N)$.

→ According to the forward procedure, the sequence can only ~~lead to~~ end in d, i.e. $u_k = d$ & the packet has reached its destination in at most N-1 hops.

~~Theorem 4.6~~

Algorithm 4.4: The FLOYD-WARSHALL ALGORITHM

begin (* Initialize S to ϕ & D to d-distance *)

$S := \phi$;

forall u, v do
  if $u = v$ then $D[u,v] := 0$
  else if $uv \in E$ then $D[u,v] := w_{uv}$.
    else $D[u,v] := \infty$;

(* Expand S by pivoting *)

while $S \neq V$ do
  (* loop invariant : $\forall u, v$: $D[u,v] = d^S(u,v)$ *)
  begin pick w from $V \setminus S$;
    (* Execute a global w-pivot *)

    forall $u \in V$ do
      (* Execute a local w-pivot at u *)
      for all $v \in V$ do
        $D[u,v] := \min(D[u,v], D[u,w] + D[w,v])$;

    $S := S \cup \{w\}$
  end (* $\forall u, v$; $D[u,v] = d(u,v)$ *)

Theorem 4.6 :- Algorithm 4.4 computes the distance b/w each pair of nodes in $O(n^3)$ steps.

Proof :- The algorithm starts with $D[u,v] = 0$ if $u = v$, $D[u,v] = w_{uv}$ if $uv \in E$ & $D[u,v] = \infty$ otherwise, & $S = \phi$. Hence by proposition 4.5, $\forall u, v$: $D[u,v] = d^S(u,v)$ holds parts (1) & (2) For a pivot round with pivot-node w the set S_i T expanded with w, & the assignment to $D[u,v]$ ensures (by parts (3) 4 (4) of the Proposition) that the assertion $\forall u,v: D[u,v] = d^S(u,v)$ is preserved as a loop invariant. The program terminates when $S = V$, i.e. (by parts (5) 4 (6) of the Proposition & the loop invariant) the S-distances equal the distances.

→ The main loop is executed N times, & contains N^2 operations (which can be executed in parallel or serially), which implies the time bound stated in the theorem.

Proposition 4.5 :- For all u & S, $d^S(u,u) = 0$, Further S-paths satisfy the following rules for $u \neq v$:-

(1) There exists an ϕ-path from u to v iff $uv \in E$.
(2) If $uv \in E$ then $d^\phi(u,v) = w_{uv}$, otherwise $d^\phi(u,v) = \infty$.

(3) If $S' = S \cup \{w\}$ then a simple S'-path from u to v is an S-path from u to v or an S-path from u to w concatenated with an S-path from w to v.

(4) If $S' = S \cup \{w\}$ then $d^{S'}(u,v) = \min(d^S(u,v), d^S(u,w) + d^S(w,v))$

(5) A path from u to v exists iff a V-path from u to v exists.

(6) $d(u,v) = d^V(u,v)$.

Lemma 4.7 :- Let S & w be given & suppose that

(1) for all u $D_u[w] = d^S(u,w)$ &

(2) if $d^S(u,w) < \infty$ & $u \neq w$, then $Nb_u[w]$ is the 1st channel of a shortest-s-paths to w.

then the directed ~~path~~ graph $T_w = (V_w, E_w)$, where

$(u \in V_w \iff D_u[w] < \infty)$ & $(un \in E_w \iff (u \neq w \wedge Nb_u[w] = n))$ is a tree rooted towards w.

Proof :- First observe that if $D_u[w] < \infty$ for $u \neq w$ then $Nb_u[w] \neq udef$ & $D_{Nb_u[w]}[w] < \infty$. So for each node $u \in V_w$, $u \neq w$ there is a node n for which $Nb_u[w] = n$, & this node satisfies $n \in V_w$.

→ for each node $u \neq w$ in V_w there is one edge in E_w, so the no. of nodes ~~in~~ of T_w exceeds the no. of edges by one & it suffices to show that T_w contains no cycle. As $un \in E_w$ implies that $d^S(u,w) = w_{un} + d^S(n,w)$, the existence of a cycle $(u_0, u_1 --- u_k)$ in T_w implies that

$$d^S(u_0,w) = w_{u_0u_1} + w_{u_1u_2} + --- w_{u_{k-1}u_0} + d^S(u_0,w)$$

i.e. $0 = w_{u_0u_1} + w_{u_1u_2} + --- w_{u_{k-1}u_0}$, which contradicts the assumption that each cycle has a +ve weight

Remembering:

Algorithm 4.5 : The Simple Algorithm (for node u)

var Su : set of nodes ;
    Du : array of weights ;
    Nbu : array of nodes ;

begin Su := φ ;
    forall v ∈ V do
        if v = u
            then begin Du[v] := 0 ; Nbu[v] := udef
        else if v ∈ Neighu
            then begin Du[v] := wuv ; Nbu[v] := v end
            else begin Du[v] := ∞ ; Nbu[v] := udef end ;

    while Su ≠ V do
    begin pick w from V \ Su ;
        (* All nodes must pick the same node w here *)
        if u = w
            then "broadcast the table Dw"
        else "receive the table Dw" ;
        forall v ∈ V do
            if Du[w] + Dw[v] < Du[v] then
                begin Du[v] := Du[w] + Dw[v] ;
                    Nbu[v] := Nbu[w]
                end ;
        Su := Su ∪ {w}
    end
end ;

Theorem 4.8 :- Algorithm 4.5 terminates in each node after N iterations of the main loop. When the algorithm terminates in node u, Du[V] = d(u,v), & if a path from u to v exists then Nbu[V] is the first channel of a shortest path from u to v; otherwise Nbu[V] = udef.

Proof :- The termination & correctness of Du[V] on termination follows from the correctness of the Floyd-Warshall algorithm (theorem 4.6).

→ The statement about the value of Nbu[V] follows because Nbu[V] is updated each time Du[V] is assigned.

Algo 4.6 :- TOUEG's Algo (for node u)

Var Su : set of nodes, Du : array of weights,
    Nbu : array of nodes.

begin Su := ∅;
  forall v ∈ V do
    if v = u
      then begin Du[V] := 0; Nbu[V] := udef end
    else if v ∈ Neighu
      then begin Du[V] := wuv; Nbu[V] := v
        end
    else begin Du[V] := ∞; Nbu[V] := udef
        end;

while Su ≠ V do
  begin Pick w from V \ Su;
    (* Compute the tree Tw *)
    forall x ∈ Neighu do
      if Nbu[w] = x then send (ys, w) to x
        else send (nys, w) to x;
    num_recu := 0; (* u must receive |Neighu| messages *)

while num_rec < |Neigh_u| do
begin receive ⟨ys, w⟩ or ⟨nys, w⟩ message;
num_rec_u := num_rec_u + 1
end;
if $D_u[w] < \infty$ then (* participate in pivot round *)
begin if $u \neq w$
then receive ⟨dtab, w, D⟩ from $Nb_u[w]$;
forall $x \in Neigh_u$ do
if ⟨ys, w⟩ was received from x
then send ⟨dtab, w, D⟩ to x;
forall $v \in V$ do (* local w-pivot *)
if $D_u[w] + D[v] < D_u[v]$ then
begin $D_u[v] := D_u[w] + D[v]$;
$Nb_u[v] := Nb_u[w]$
end.
end;
end.

Th. 4.9 :- Alg. 4.6 computes for each u & v the distance from u to v, & if dist = finite, the 1st channel of a path of this length, the algorithm exchanges $O(N)$ messages per channel, $O(N \cdot |E|)$ messages in total, $O(N^2 W)$ bits per channel, $O(N^3 W)$ bits in total, & requires $O(NW)$ bits of storage per node.

Proof ⇒ Algorithm 4.6 is derived from Algo 4.5, which implies its correctness.

→ Each channel carries 2 $\langle ys, w\rangle$ or $\langle nys, w\rangle$ messages (one in each direc^n) & at most one $\langle dtab, w, D\rangle$ message in the w-pivot round, which totals to atmost 3N messages per channel. A $\langle ys, w\rangle$ or $\langle nys, w\rangle$ message contains $O(W)$ bits & a $\langle dtab, w, D\rangle$ message contains $O(NW)$ bits, which gives the bound on the no. of bits per channel.

At most N^2 $\langle dtab, w, D\rangle$ messages & $2N \cdot |E|$ $\langle ys, w\rangle$ & $\langle nys, w\rangle$ messages are exchanged, which totals to $O(N^2 \cdot NW + 2N \cdot |E| \cdot W)$
$= O(N^3 W)$ bits altogether. The D_u & Nb_u tables maintained in node u require $O(NW)$ bits.

__Lemma 4.10__ :- Let $u_1 \in W$, & u_2 be a descendant of u_1 in T_w at the beginning of the w-pivot round. If u_2 changes its distance to v in the round, then u_1 changes its distance to v in the w-pivot round.

__Proof__ :- As u_2 is a descendant of u_1 in T_w,

$$d^S(u_2, w) = d^S(u_2, u_1) + d^S(u_1, w) \quad -①$$

Because $u_1 \in S$,
$$d^S(u_2, v) \le d^S(u_2, u_1) + d^S(u_1, v) \quad ②$$

If node u_2 changes $D_{u_2}[v]$ in this round. iff
$$d^S(u_2, w) + d^S(w, v) < d^S(u_2, v) \quad ③$$

By applying ② & ③ & then ①, & subtracting $d^S(u_2, u_1)$

we obtain $\boxed{d^S(u_1, w) + d^S(w, v) < d^S(u_1, v)}$ $-④$

⇒ u_1 changes $D_{u_1}[v]$ in this round.

Theorem 4.11 → The algorithm of Merlin & Segall computes the shortest path routing tables by exchanging $O(N^2)$ messages per channel, & $O(N^2 W)$ bits per channel, $O(N^2 \cdot |E|)$ messages in total, & $O(N^2 \cdot |E| W)$ bits in total.

Th 4.12 :- In each computation of Algo. 4.7, a configuration is reached in which, for each node u, $D_u[v_0] \leq d(u, v_0)$.

Th 4.13 → The algo. of Chandy & Misra computes min. hop routing tables by exchanging $O(N^2)$ messages & $O(N^2 W)$ bits per channel, & $O(N^2 \cdot |E|)$ messages & $O(N^2 \cdot |E| \cdot W)$ bits in total.

Lemma 4.14 → for all u_0, w_0, & v_0, $P(u_0, w_0, v_0)$ is an invariant.

Pf → Initially, each node (1) holds by assumption.

If initially we have $\neg$ $ub(u_0, w_0)$, (2) & (3) trivially hold.

→ If initially, we have have $ub(u_0, w_0)$, then if $ndis_{u_0}[w_0, v_0] = N$. If $w_0 = v_0$, then $D_{w_0}[w_0] = 0$ but a message $\langle mydist, v_0, 0 \rangle$ is in $Q_{w_0 u_0}$, so (2) & (3) are true.

If $w_0 \neq v_0$, then $D_{w_0}[v_0] = N$ & no message is in the queue, which also implies that (2) & (3) hold.

Type(1) :- Assume that u receives a <mydist, v, d> message from w. This causes no topological change & no change in the neigh sets, hence (1) remains true. as $v \neq v_0$, this receipt doesn't change anything in $P(u_0, w_0, v_0)$.

Type(2):- Assume that channel uw fails.

Type(3):- Assume that channel uw is added.

Lemma 4.15 → for each u_0 & v_0, $L(u_0, v_0)$ is an invariant.

Proof → Initially, $D_{u_0}[v_0] = 0$ & $Nb_{u_0}[u_0] = local$. for $v_0 \neq u_0$, initially $ndis_{u_0}[w, v_0] = N$ for all $w \in Neigh_u$, & $D_{u_0}[v_0] = N$ & $Nb_{u_0}[v_0] = udef$.

Type(1) :- Assume that u receives a <mydist, v, d> message from w.

Type(2):- Assume that channel uw fails.

Type(3):- Assume that channel uw is added.

Theorem 4.16 :- When a stable configuration is reached, the tables $Nb_u[v]$ satisfy :-

(1) If $u = v$, then $Nb_u[v] = local$.
(2) If a path from u to $v \neq u$ exists, then $Nb_u[v] = w$, where w = 1st neighbours of u on a shortest path from u to v;
(3) If no path from u to v exists then $Nb_u[v] = udef$.

Lemma 4.21 :- there is uniform bound on the ratio b/w $d_T(u,v)$ & $d_G(u,v)$. This holds already in the special case of the hop measure for paths.

Proof :- Choose G to be the ring on N nodes, & observe that a spanning tree of G is obtained by removing one channel, say xy, from G. Now $d_G(x,y) = 1$ & $d_T(x,y) = N-1$, so the ratio is $N-1$. The ratio can be made arbitrarily large by choosing a large ring.

Lemma 4.22 :- T can be choosen in such a way that $\forall$ u & v, $d_T(u,v) \leq 2D_G$.

→ Let T = optimal sink tree for node w_0.

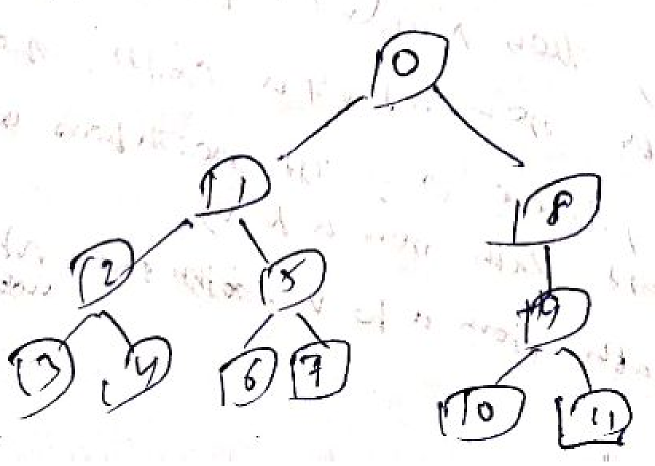
$\Rightarrow d_T(u,v) \leq d_T(u,w_0) + d_T(w_0,v)$

$= d_T(u,w_0) + d_T(v,w_0)$ → by symm

$= d_G(u,w_0) + d_G(v,w_0)$ → by choice of T,

$\leq D_G + D_G$, (proved)

Th 4.23 :- for each connected n/w G, a valid integral labelling scheme exists.



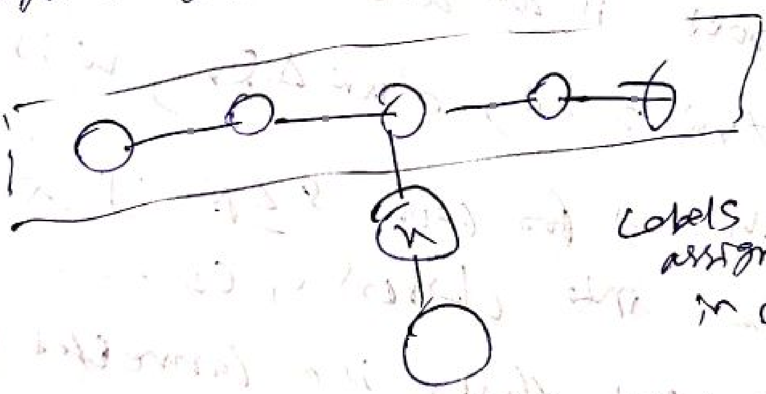
Lemma 4.26 → ∃ a spanning tree ∋ all frond edges are b/w a node & an ancestor of that node.

Pf → Each spanning tree that is obtained by a DFS through the n/w has this property.

Lemma 4.30 → If $l_u < l_v$, then $fr(w) < fr(u)$.

Th. 4.32 → ∃ a N/w G ∋ ∀ valid ILS of G ∃ nodes u & v ∋ a packet from u to v is delivered only after at least $\frac{3}{2} D_u$ hops.

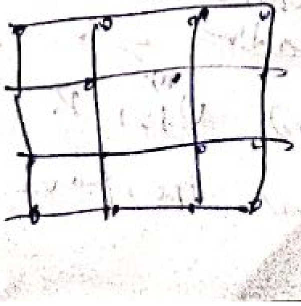
Th. 4.33 → ∃ a N/w for which no valid linear-interval labelling scheme exists.

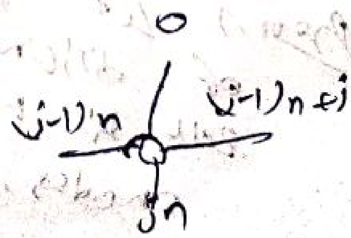


Labels 0 & 6 are assigned to nodes in this box.

Th. 4.37 → ∃ a min-hop ILS for a ring of N nodes.

Th. 4.35 → ∃ a min_hop ILS for n×n grid.





Th. 4.36 → ∃ min. hop linear PLS for hypercube.

Th. 4.37 → ∃ shortest path PLS for outerplanar networks with arbitrary channel weights.

Th. 4.40 → for each connected NW G, a valid PLS exists.

Lemma 4.40 → If $u \notin T[v]$, then w is an ancestor of v or $d_T(w,v) < d_T(u,v)$.

Proof → If $\alpha_{vw} = \epsilon$, then w is father of u or the root; the father of u is closer to v than u, because $u \notin T[v]$, & the root is an ancestor of v.

If $\alpha_{vw} = l_w$, $\alpha_{uw} \le l_v$, w is ancestor of

Lemma 4.47 → for each $s \le N$ ∃ a division of the NW into clusters $C_1, C_2 \text{------} C_m$ ∋
(1) each cluster is a connected subgraph,
(2) each cluster contains at least s nodes &
(3) each cluster has radius at most $2s$.

Proof → Let $D_1 \text{---} D_m$ be a maximal collection of disjoint connected subgraphs such that each D_i has radius $\le s$ & contains at least s nodes. ⇒ $\bigcup_{i=1}^{m} D_i$ is closed & it ⇒ they have radius at most $2s$.

Scanned with CamScanner

Th. 4.48 For every n/w of n nodes, there is a routing method that req. at most $O(\sqrt{n})$ routing decision for each packet & uses 3 colours.

Th. 4.49 → For every n/w of n nodes & every +ve integer $f \le \log n$, there is a routing method that req. at most $O(f \cdot n^{1/f})$ routing decision for each packet, & uses $2f+1$ colours.

Proof :- The argument is similar to the proof of Th. 4.48, but instead of choosing $s \approx \sqrt{n}$, choose $s \approx 2n^{1/f}$.

Hence, no. of routing decisions is bounded by $f \cdot s = O(f \cdot n^{1/f})$. (proved)