

K_n : vplyv orientácie

zmysel pre orientáciu

Porty sú číslované podľa vzdialenosti vo fixnej Hamiltonovej kružnici.

- algoritmus: zajať fixný pattern $\{i[1..k], i[2k], i[3k], \dots, i[n-k]\}$
- prvá fáza $i[1..k]$
 - *level*, *ID*
 - *level* je počet zajatých
 - pripája sa celé územie (planarita!)
- druhá fáza $i[2k], i[3k], \dots, i[n-k]$
 - nastav *owner* vrcholom $i[1..k]$, *ack*
 - pošli *elect* do $i[2k], i[3k], \dots, i[n-k]$
 - *elect*: ak je v prvej fáze, alebo je slabší \Rightarrow accept

počet správ

- prvá fáza: $O(n)$: z planarity
- druhá fáza: max $O(n/k)$ kandidátov, každý pošle n/k správ $\Rightarrow O(n^2/k^2)$ správ

čas

- po zobudení (najsilnejšieho) $O(k)$, v najhoršom $O(n)$
- pridať wakeup fázu (poslať od $i[1], i[k]$) $\Rightarrow O(k + n/k)$
- $k := \sqrt{n}$

vylepšit' čas

- prvá fáza: zajať $i[k], i[2k], \dots, i[n - k]$ sekvenčne
- vzniknú nezávislé "kruhy" R_0, \dots, R_{k-1} , $R_j = \{i[j + k], i[j + 2k], \dots, i[j + n - k]\}$
- druhá fáza: zajať $i[1..k - 1]$ v $\log k$ krokoch
 - level, ID
 - v prvom kroku zajať $i[k/2]$
 - v l -tom kroku zajať $i[k/2^l], i[3k/2^l], \dots, i[(2^l - 1)k/2^l]$ ako v úplnom grafe (za zajatých bojuje šéf)
 - v l -tom kroku je $k/2^l$ kandidátov; je $\log k$ krokov $\Rightarrow O(k \log k)$ správ
- $k := n / \log n$

algoritmy založené na porovnaníach

- ekvivalentné okolia
- c -symetrický reťazec: pre každé $\sqrt{n} \leq l \leq n$ a pre každý segment S je $\lfloor \frac{cn}{l} \rfloor$ ekvivalentných
- bit-reversal je $1/2$ -symetrický
- c -symetrický reťazec, alg. nemôže skončiť po k kolách pre $\lfloor \frac{cn}{2k+1} \rfloor \geq 2$
- počet správ: $k = \lfloor \frac{cn-2}{4} \rfloor$, je aspoň $k + 1$ kôl
- aspoň $\lfloor \frac{cn}{2r-1} \rfloor$ aktívnych v r -tom kole pošle správu

algoritmy využívajúce integer ID

- rôzne rýchlosti
- v i -tom kroku test

V toruse rozmerov $n \times n$ (t.j. zacyklenej mriežke) sú na začiatku zobudené dva vrcholy. Napíšte algoritmus, pomocou ktorého sa každý z nich dozvie identifikátor druhého s použitím $O(n)$ správ.

Nájdite asymptoticky optimálny (čo do počtu správ) algoritmus na voľbu šéfa v úplnom bipartitnom grafe $K_{n,n}$. Dokážte jeho zložitosť a optimalitu.

★

broadcasting a voľba šéfa na (ne?)orientovanej hyperkočke s lineárnym počtom správ

cieľ: doručovať srávy medzi ľubovoľnou dvojicou

modely

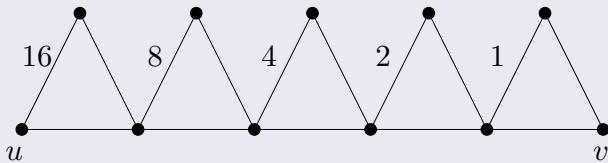
- destination based
- splittable
- connections (wormhole)
- buffering
- selfish

ciele

- statické váhy (najkratšie cesty)
- dynamické váhy (hot potato)
- deadlock

najkratšie cesty

ako DFS



exponenciálne veľa správ

najkratšie cesty

```
begin (* Initialize  $S$  to  $\emptyset$  and  $D$  to  $\emptyset$ -distance *)  
   $S := \emptyset$  ;  
  forall  $u, v$  do  
    if  $u = v$  then  $D[u, v] := 0$   
    else if  $uv \in E$  then  $D[u, v] := \omega_{uv}$   
    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  $u$  *)  
      forall  $v \in V$  do  
         $D[u, v] := \min ( D[u, v], D[u, w] + D[w, v] )$  ;  
       $S := S \cup \{w\}$   
    end (*  $\forall u, v : D[u, v] = d(u, v)$  *)  
end
```

najkratšie cesty

```
var  $S_u$  : set of nodes ;  
     $D_u$  : array of weights ;  
     $Nb_u$  : array of nodes ;  
  
begin  $S_u := \emptyset$  ;  
    forall  $v \in V$  do  
        if  $v = u$   
            then begin  $D_u[v] := 0$  ;  $Nb_u[v] := u$  def end  
            else if  $v \in Neigh_u$   
                then begin  $D_u[v] := \omega_{uv}$  ;  $Nb_u[v] := v$  end  
                else begin  $D_u[v] := \infty$  ;  $Nb_u[v] := u$  def end ;  
    while  $S_u \neq V$  do  
        begin pick  $w$  from  $V \setminus S_u$  ;  
            (* All nodes must pick the same node  $w$  here *)  
            if  $u = w$   
                then “broadcast the table  $D_w$ ”  
                else “receive the table  $D_w$ ”  
            forall  $v \in V$  do  
                if  $D_u[w] + D_w[v] < D_u[v]$  then  
                    begin  $D_u[v] := D_u[w] + D_w[v]$  ;  
                         $Nb_u[v] := Nb_u[w]$   
                    end ;  
                 $S_u := S_u \cup \{w\}$   
            end  
    end  
end
```


najkratšie cestv

```
var  $S_u$  : set of nodes ;
     $D_u$  : array of weights ;
     $Nb_u$  : array of nodes ;

begin  $S_u := \emptyset$  ;
      forall  $v \in V$  do
        if  $v = u$ 
          then begin  $D_u[v] := 0$  ;  $Nb_u[v] := undef$  end
          else if  $v \in Neigh_u$ 
            then begin  $D_u[v] := \omega_{uv}$  ;  $Nb_u[v] := v$  end
            else begin  $D_u[v] := \infty$  ;  $Nb_u[v] := undef$  end ;
        while  $S_u \neq V$  do
          begin pick  $w$  from  $V \setminus S_u$  ;
                (* Construct the tree  $T_w$  *)
                forall  $x \in Neigh_u$  do
                  if  $Nb_u[w] = x$  then send  $\langle \mathbf{ys}, w \rangle$  to  $x$ 
                  else send  $\langle \mathbf{nys}, w \rangle$  to  $x$  ;
                num_rec_u := 0 ; (*  $u$  must receive  $|Neigh_u|$  messages *)
                while num_rec_u <  $|Neigh_u|$  do
                  begin receive  $\langle \mathbf{ys}, w \rangle$  or  $\langle \mathbf{nys}, w \rangle$  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  $\langle \mathbf{dtab}, w, D \rangle$  from this  $Nb_u[w]$  ;
                        forall  $x \in Neigh_u$  do
                          if  $\langle \mathbf{ys}, w \rangle$  was received from  $x$ 
                            then send  $\langle \mathbf{dtab}, w, D \rangle$  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 ;
                end ;
```

Netchange

```
var  $Neigh_u$  : set of nodes ; (* The neighbors of  $u$  *)  
     $D_u$  : array of 0..  $N$  ; (*  $D_u[v]$  estimates  $d(u, v)$  *)  
     $Nb_u$  : array of nodes ; (*  $Nb_u[v]$  is preferred neighbor for  $v$  *)  
     $ndis_u$  : array of 0..  $N$  ; (*  $ndis_u[w, v]$  estimates  $d(w, v)$  *)
```

Initialization:

```
begin forall  $w \in Neigh_u, v \in V$  do  $ndis_u[w, v] := N$  ;  
    forall  $v \in V$  do  
        begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end ;  
         $D_u[u] := 0$  ;  $Nb_u[u] := local$  ;  
    forall  $w \in Neigh_u$  do send  $\langle mydist, u, 0 \rangle$  to  $w$   
end
```

Procedure *Recompute* (v):

```
begin if  $v = u$   
    then begin  $D_u[v] := 0$  ;  $Nb_u[v] := local$  end  
    else begin (* Estimate distance to  $v$  *)  
         $d := 1 + \min\{ndis_u[w, v] : w \in Neigh_u\}$  ;  
        if  $d < N$  then  
            begin  $D_u[v] := d$  ;  
                 $Nb_u[v] := w$  with  $1 + ndis_u[w, v] = d$   
            end  
        else begin  $D_u[v] := N$  ;  $Nb_u[v] := undef$  end  
    end ;  
if  $D_u[v]$  has changed then  
    forall  $x \in Neigh_u$  do send  $\langle mydist, v, D_u[v] \rangle$  to  $x$   
end
```

Processing a $\langle mydist, v, d \rangle$ message from neighbor w :

```
{ A  $\langle mydist, v, d \rangle$  is at the head of  $Q_{wv}$  }  
begin receive  $\langle mydist, v, d \rangle$  from  $w$  ;  
     $ndis_u[w, v] := d$  ; Recompute ( $v$ )  
end
```

Upon failure of channel uw :

```
begin receive  $\langle fail, w \rangle$  ;  $Neigh_u := Neigh_u \setminus \{w\}$  ;  
    forall  $v \in V$  do Recompute ( $v$ )  
end
```

Upon repair of channel uw :

```
begin receive  $\langle repair, w \rangle$  ;  $Neigh_u := Neigh_u \cup \{w\}$  ;  
    forall  $v \in V$  do  
        begin  $ndis_u[w, v] := N$  ;  
            send  $\langle mydist, v, D_u[v] \rangle$  to  $w$   
        end  
end
```

end

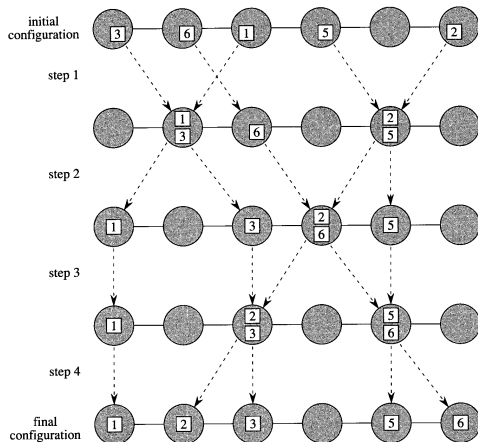
korektnosť

lexikograficky klesá hodnota $[t_0, t_1, \dots, t_N]$

kde t_i je počet správ $\langle mydist, i \rangle$ + počet dvojíc u, v kde $D_u[v] = i$

packet routing

- synchrónny režim
- vrcholy majú pakety (uložené v bufferoch)
- v jednom kroku po jednej linke ide max. jeden paket
- algoritmus = odchádzajúce linky + prioritita bufferov
- celkový čas



packet routing na mriežke $\sqrt{N} \times \sqrt{N}$

vstup

Každý vrchol má 1 paket, do každého smeruje 1 paket (permutation routing)

algoritmus

Najprv riadok, potom stĺpec. Prednosť má ten s najdlhšou cestou.

analýza: stačí $2\sqrt{N} - 2$ krokov

- po $\sqrt{N} - 1$ krokoch je každý v správnom stĺpci (nebrzdia sa)
- routovanie v stĺpci ide v $\sqrt{N} - 1$ krokoch
 - pre každé i platí: po $N - 1$ krokoch sú koncové pakety na koncových miestach
 - dôvod: zdržujú sa iba navzájom