dvojsmerné kruhy

Hirschberg-Sinclair

- level l: dobýjať územie 2^l
- $\log n$ levelov
- $n/2^l$ vrcholov na leveli
- každý vrchol pošle 2^l správ

dvojsmerné kruhy

Hirschberg-Sinclair

- level l: dobýjať územie 2^l
- $\log n$ levelov
- $n/2^l$ vrcholov na leveli
- každý vrchol pošle 2^l správ

Franklin

- level l: poraziť susedov (na rovnakom leveli; "synchronizácia")
- $\log n$ levelov
- n správ na level

dvojsmerné kruhy

Hirschberg-Sinclair

- level l: dobýjať územie 2^l
- $\log n$ levelov
- $n/2^l$ vrcholov na leveli
- každý vrchol pošle 2^l správ

Franklin

- level l: poraziť susedov (na rovnakom leveli; "synchronizácia")
- $\log n$ levelov
- n správ na level

Dolev – simulácia na 1-smernom kruhu

- idea: presunúť identitu

dolný odhad

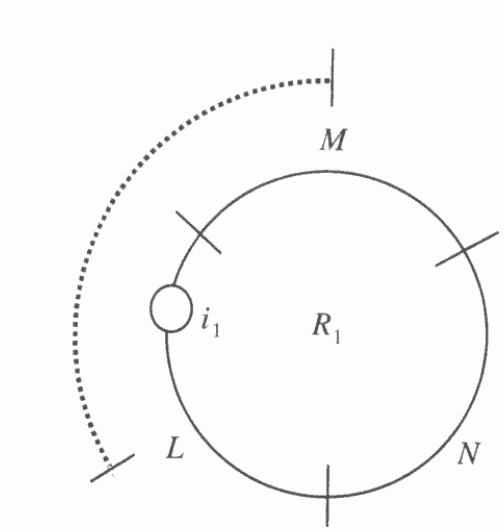
zapojiť do čiary L, vymení sa $C(L)$ správ

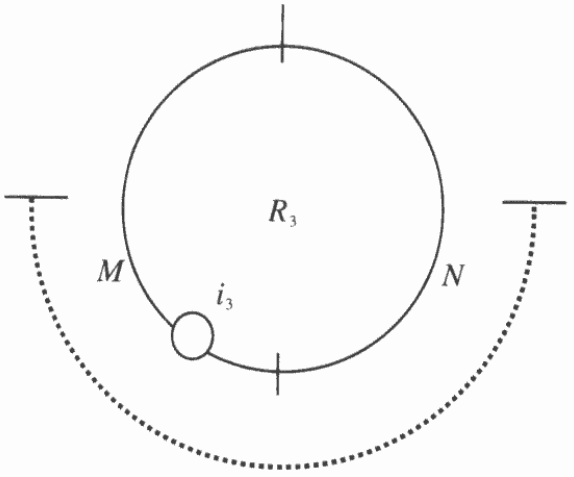
lema

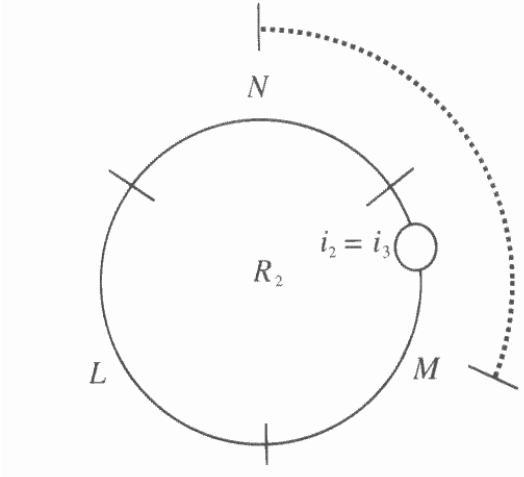
Pre každé r existuje nekonečne veľa čiar dĺžky 2^r, kde $C(L) \geq r2^{r-2}$

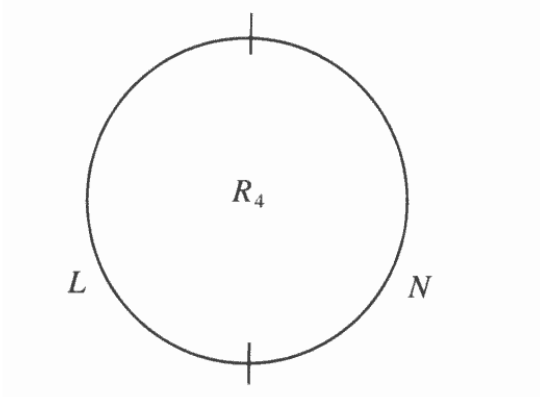
- indukcia
- dve vrecia: vyberám trojice, chcem dve spojiť item pri spojení: dĺžka 2^{r+1}, počet správ $\geq r2^{r-1}$
- ešte treba 2^{r-1} správ; sporom

dolný odhad









GHS

- ľubovoľná topológia
- buduje sa kostra
- "segmenty"
- spájanie po najlacnejšej odchádzajúcej hrane:
  - A menší sa pripojí k väčšiemu
  - B rovnakí sa spoja
  - C väčší čaká
- veľkosť $\Rightarrow$ level

GHS

var $state_p$: $(sleep, find, found)$;
$stach_p[q]$: $(basic, branch, reject)$ for each $q \in Neigh_p$;
$name_p$, $bestwt_p$: real ;
$level_p$: integer ;
$testch_p$, $bestch_p$, $father_p$: $Neigh_p$;
rec_p : integer ;

(1) As the first action of each process, the algorithm must be initialized:
begin let pq be the channel of p with smallest weight ;
$stach_p[q] := branch$; $level_p := 0$;
$state_p := found$; $rec_p := 0$;
send $\langle$ **connect**, 0 $\rangle$ to q
end

(2) Upon receipt of $\langle$ **connect**, L $\rangle$ from q:
begin if $L < level_p$ **then** (* Combine with Rule A *)
begin $stach_p[q] := branch$;
send $\langle$ **initiate**, $level_p$, $name_p$, $state_p$ $\rangle$ to q
end
else if $stach_p[q] = basic$
then (* Rule C *) process the message later
else (* Rule B *) send $\langle$ **initiate**, $level_p + 1, \omega(pq), find$ $\rangle$ to q
end

(3) Upon receipt of $\langle$ **initiate**, L, F, S $\rangle$ from q:
begin $level_p := L$; $name_p := F$; $state_p := S$; $father_p := q$;
$bestch_p := udef$; $bestwt_p := \infty$;
forall $r \in Neigh_p : stach_p[r] = branch \wedge r \neq q$ **do**
send $\langle$ **initiate**, L, F, S $\rangle$ to r ;
if $state_p = find$ **then begin** $rec_p := 0$; $test$ **end**
end

GHS

(4) **procedure** *test*:
begin if $\exists q \in Neigh_p : stach_p[q] = basic$ **then**
begin $testch_p := q$ with $stach_p[q] = basic$ and $\omega(pq)$ minimal ;
send $\langle$ **test**, $level_p$, $name_p$ $\rangle$ to $testch_p$
end
else begin $testch_p := udef$; *report* **end**
end

(5) Upon receipt of $\langle$ **test**, L, F $\rangle$ from q:
begin if $L > level_p$ **then** (* Answer must wait! *)
process the message later
else if $F = name_p$ **then** (* internal edge *)
begin if $stach_p[q] = basic$ **then** $stach_p[q] := reject$;
if $q \neq testch_p$
then send $\langle$ **reject** $\rangle$ to q
else *test*
end
else send $\langle$ **accept** $\rangle$ to q
end

(6) Upon receipt of $\langle$ **accept** $\rangle$ from q:
begin $testch_p := udef$;
if $\omega(pq) < bestwt_p$
then begin $bestwt_p := \omega(pq)$; $bestch_p := q$ **end** ;
report
end

(7) Upon receipt of $\langle$ **reject** $\rangle$ from q:
begin if $stach_p[q] = basic$ **then** $stach_p[q] := reject$;
test
end

GHS

(8) **procedure** *report*:
begin if $rec_p = \#\{q : stach_p[q] = branch \wedge q \neq father_p\}$
and $testch_p = udef$ **then**
begin $state_p := found$; send $\langle$ **report**, $bestwt_p$ $\rangle$ to $father_p$ **end**
end

(9) Upon receipt of $\langle$ **report**, ω $\rangle$ from q:
begin if $q \neq father_p$
then (* reply for **initiate** message *)
begin if $\omega < bestwt_p$ **then**
begin $bestwt_p := \omega$; $bestch_p := q$ **end** ;
$rec_p := rec_p + 1$; *report*
end
else (* pq is the core edge *)
if $state_p = find$
then process this message later
else if $\omega > bestwt_p$
then *changeroot*
else if $\omega = bestwt_p = \infty$ **then stop**
end

(10) **procedure** *changeroot*:
begin if $stach_p[bestch_p] = branch$
then send $\langle$ **changeroot** $\rangle$ to $bestch_p$
else begin send $\langle$ **connect**, $level_p$ $\rangle$ to $bestch_p$;
$stach_p[bestch_p] := branch$
end
end

(11) Upon receipt of $\langle$ **changeroot** $\rangle$:
begin *changeroot* **end**

analýza

správnosť

ukázať, že sa zvolí práve jeden šéf: nenastane deadlock

počet správ

- testovacie správy: jeden test po každej hrane
- kostrové správy: fragment s n_i vrcholmi pri postupe o level $O(n_i)$ správ
- postupy na level l: dizjunktné vrcholy

KKM

- $f(x)$-traverzovanie
- tokeny traverzujú/označujú územia
- levely: keď sa stretnú dva, vznikne nový
- naháňanie

KKM

```
var lev_p          : integer     init −1 ;
    cat_p, wait_p  : P           init udef ;
    last_p         : Neigh_p     init udef ;

begin if p is initiator then
        begin lev_p := lev_p + 1 ; last_p := trav(p, lev_p) ;
              cat_p := p ; send ⟨ annex, p, lev_p ⟩ to last_p
        end ;
      while ... (* Termination condition, see text *) do
        begin receive token (q, l) ;
              if token is annexing then t := A else t := C ;
              if l > lev_p then (* Case I *)
                begin lev_p := l ; cat_p := q ;
                      wait_p := udef ; last_p := trav(q, l) ;
                      send ⟨ annex, q, l ⟩ to last_p
                end
              else if l = lev_p and wait_p ≠ udef then (* Case II *)
                begin wait_p := udef ; lev_p := lev_p + 1 ;
                      last_p := trav(p, lev_p) ; cat_p := p ;
                      send ⟨ annex, p, lev_p ⟩ to last_p
                end
              else if l = lev_p and last_p = udef then (* Case III *)
                wait_p := q
              else if l = lev_p and t = A and q = cat_p then (* Case IV *)
                begin last_p := trav(q, l) ;
                      if last_p = decide
                        then p announces itself leader
                        else send ⟨ annex, q, l ⟩ to last_p
                end
              else if l = lev_p and ((t = A and
                   q > cat_p) or t = C) then (* Case V *)
                begin send ⟨ chase, q, l ⟩ to last_p ; last_p := udef end
              else if l = lev_p then (* Case VI *)
                wait_p := q
```

KKM

počet správ

- naháňacie: 1 na vrchol a level, spolu n za level
- objavovacie: $\sum_i f(n_i)$
- ak f je konvexná, t.j. $f(a) + f(b) \leq f(a + b)$, tak je $O(\log n(n + f(n)))$ správ

K_n: vplyv orientácie

zmysel pre orientáciu

Porty sú číslované podľa vzdialenosti vo fixnej Hamiltonovej kružnici.

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

K_n: vplyv orientácie

počet správ

- prvá fáza: $O(n)$: odpoveď zajatému 1x, dizjunknosť území
- 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 $i[1], i[k]$) $\Rightarrow$ $O(k + n/k)$
- $k := \sqrt{n}$

vplyv synchrónnosti

algoritmy založené na porovnaniach

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

vplyv synchrónnosti

algoritmy využívajúce integer ID

- rôzne rýchlosti
- v i-tom kroku test

cvičenie

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.

$\star$

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