Paralelni algoritmi

Modeli paralelnih računara, osnovne karakteristike paralelnih algoritama.

Osnovne mere složenosti za sekvencijalne algoritme su vreme izvršavanja i veličina korišćene memorije. One su važne i **kod paralelnih algoritama ali se kod njih mora voditi računa i broju procesora, pošto je u praksi njihov broj u velikoj meri ograničen**. Postoje problemi koji su u suštini sekvencijalni i koji se ne mogu paralelizovati, ali se većina problema može paralelizovati u određenoj meri. **Jedno od pravila koje važi jeste to da brzina izvršavanja algoritma raste sa brojem procesora, ali samo do određene granice**. Pošto je broj procesora ograničen, veoma je važno efikasno ih iskoristiti. Važan elemenat je komunikacija između procesora koju je potrebno minimizovati i organizovati na efikasan način, jer je često više vremena potrebno za razmenu podataka nego za izvršavanje nekih jednostavnih operacija nad njima. Kod problema takozvanih distribuiranih algoritama bitan faktor je sinhronizacija.

Prva podela paralelnih računara vrši se na osnovu sposobnosti njegovih procesora da u istom vremenskom trenutku izvršavaju različite instrukcije. Računari koji imaju ovu mogućnost nazivaju se MIMD (Multiple Instruction Multiple Data), a računari čiji procesori u jednom trenutku mogu da izvršavaju samo jednu istu instrukciju nad eventualno različitim podacima nazivaju se SIMD (Single Instruction Multiple Data).

```
for 1 ≤ i ≤ n do in parallel
    if i ≤ 10 then
        A[ i ] := 2 * A[ i ]
    else
        A[ i ] := A[ i ] + 1
endif
end in parallel
```

Primer 1 – Algoritam za MIMD model.

```
for 1 ≤ i ≤ n do in parallel
    A[ i ] := 2 * A[ i ]
end in parallel
for 1≤ i ≤ n do in parallel
    A[ i ] := A[ i ] + 1
end in parallel
```

Primer 1 – Algoritam za SIMD model.

U primeru 1 (MIMD algoritam), 10 procesora je dobilo zadatak izvrsavanja instrukcije A[i] = 2*A[i], dok drugi skup od *n-10* procesora ce izvrsavati instrukciju A[i] = A[i] + 1. Sva izvrsavanja su simultana.

Ali, ovo nije slucaj u primeru 2 (algoritam za SIMD model), u kom se dve razlicite instrukcije izvode simultano. Dakle, zahteva se izmenjena verzija algoritma iz primera 1. Otuda u primeru 2 postoje dve **in parallel** naredbe.

Slicno, ako bi morali izvrsavati case naredbu in parallel, onda bi u SIMD modelu imali niz **in parallel** naredbi, svaku za po jednu klauzu.

Vreme izvršavanja algoritma označavaćemo sa ***T(n,p)***, gde je ***n***, veličina ulaza, ***p*** broj procesora.

Odnos S(p)=T(n,1)/T(n,p) zove se ubrzanje algoritma.

Za paralelni algoritam se kaže da dostiže savršeno ubrzanje ukoliko je S(p)=p.

Za vrednost T(n,1) uzima se složenost najboljeg sekvencijalnog algoritma koji rešava konkretan problem.

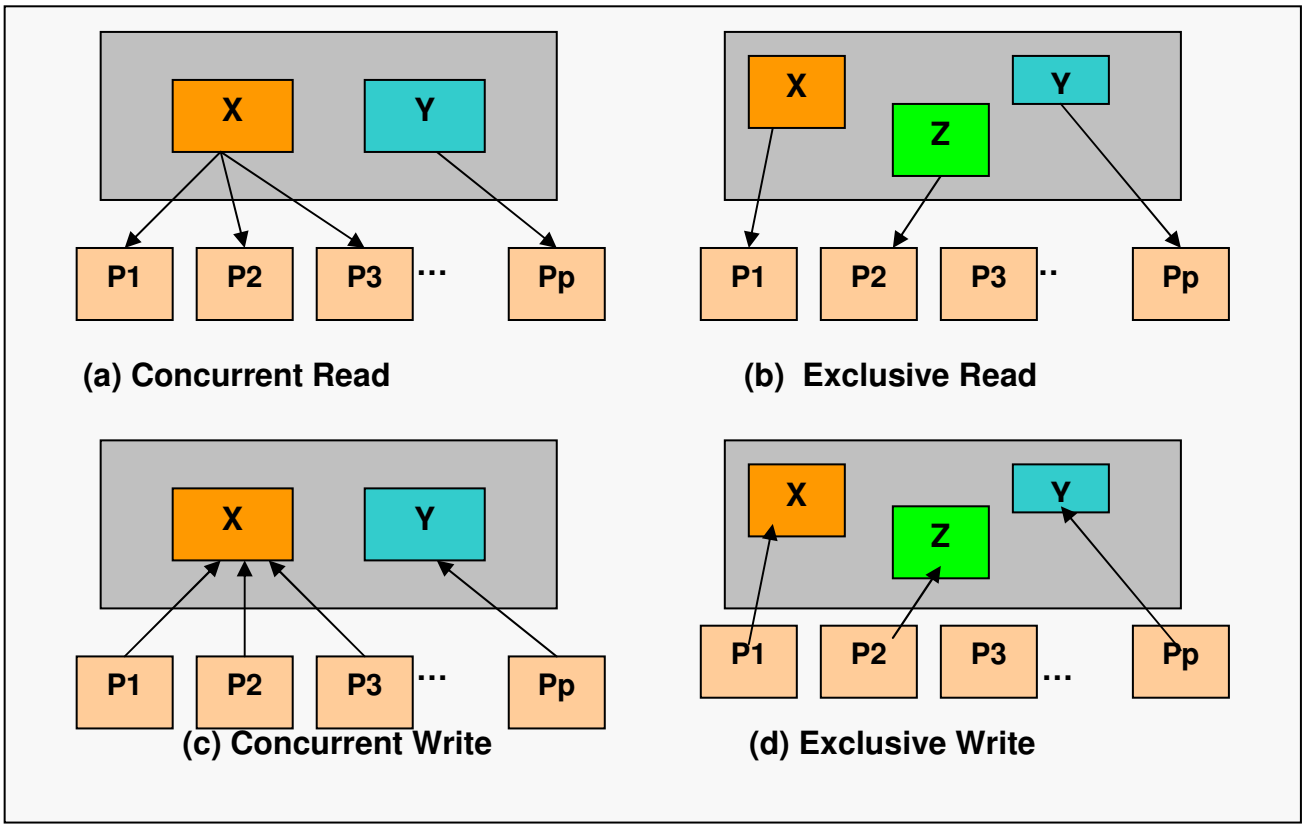
Mera iskorišćenosti procesora naziva se efikasnost i definiše se kao: E(n,p)=S(p)/p=T(n,1)/(pT(n,p))

To je odnos vremena izvršavanja na jednom procesoru i ukupnog vremena izvršavanja na ***p*** procesora.

Ako je E(n,p)=1, onda je količina računanja obavljenog na svim procesorima u toku izvršavanja algoritma, jednaka količini računanja koju zahteva sekvencijalni algoritam. Jedan od osnovnih ciljeva jeste maksimiziranje efikasnosti.

Modeli paralelnog izračunavanja se razlikuju po načinu komunikacije i sinhronizovanosti procesora. Za nas će biti od interesa samo modeli koji su potpuno sinhronizovani. Jedan od takvih modela je model računara sa zajedničkom

memorijom. Ovaj model podrazumeva da postoji zajednička memorija kojoj može pristupiti svaki od procesora za konstantno vreme. Pretpostavićemo da se računanje sastoji iz koraka. U svakom koraku jedan procesor izvršava neku operaciju nad podacima kojima raspolaže, čita iz ili piše u zajedničku memoriju.
Modeli sa zajedničkom memorijom dele se prema tome na koji način obrađuju memorijske konflikte, i razlikujemo modele
EREW(Exclusive Read Exclusive Write),
CREW(Concurrent Read Exclusive Write) i
CRCW (Concurrent Read Concurrent Write).



Model EREW ne dozvoljava da dva procesora istovremeno pristupaju istoj memorijskoj lokaciji (u smislu čitanja i pisanja), dok CREW ne dozvoljava da više procesora istovremeno upisuju podatke na istu memorijsku lokaciju, ali dozvoljava da više procesora istovremeno čitaju sa iste memorijske lokacije. Model CRCW dozvoljava da više procesora istovremeno čitaju podatke sa iste memorijske lokacije, ali i da više procesora istovremeno pišu na istu lokaciju pod uslovom da upisuju istu vrednost.

Modeli računara sa zajedničkom memorijom predstavljaju najjednostavniji način za modeliranje komunikacije ali ih je hardverski najteže realizovati. Druga grupa modela podrazumeva da su računari međusobno povezani u jednu mrežu. Mreža se može predstaviti grafom, pri čemu čvorovi odgovaraju procesorima, i povezani su tamo gde postoji direktna veza između dva procesora. Svaki procesor ima svoju (lokalnu) memoriju, a komunikacija se vrši porukama, koje ponekad moraju da prolaze i više direktnih veza da bi došle do odgovarajućeg procesora. Brzina komunikacije najviše zavisi od rastojanja među procesorima.
Treća grupa modela su modeli sistoličkog računanja. Oni podsećaju na pokretnu traku u fabrikama. Podaci se kreću ravnomerno kroz procesore. Procesori podatke dobijaju od svojih suseda, obrađuju ih i prosleđuju dalje. Osnovni teorijski model je kolo, koje predstavlja usmereni aciklični graf u kome čvorovi odgovaraju jednostavnim operacijama, a grane predstavljaju kretanje operanada. Posebno su izdvojeni ulazni čvorovi (sa ulaznim stepenom 0) i izlazni čvorovi (sa izlaznim stepenom 0). Jedna od karakteristika kola je dubina koja predstavlja dužinu najdužeg puta od nekog ulaznog do nekog izlaznog čvora, i ona odgovara vremenu izvršavanja algoritma.

1. Konstruisati EREW algoritam složenosti T(n,n)=O(log n) koji u vektoru A dimenzije n će vrednost A[1] smestiti na preostale lokacije (tzv. broadcast).
REŠENJE:

Koristiti se EREW model, te iz lokacije A[1] može u jednom trenutku da čita samo jedan procesor.

Pretpostavimo bez smanjenja opstosti da je $n=2^k$.
Koristi se metod "dupliranja":

u prvom koraku jedan procesor kopira A[1] u A[2];
u 2. koraku učestvuje 2 procesora i oni (pojedinačno) kopiraju elemente niza A[1] A[2] u lokacije A[3] A[4].
. . .
u i-tom koraku učestvuje 2^{i-1} procesora i oni (pojedinačno) kopiraju elemente niza $A[1..2^{i-1}]$ u lokacije $A[2^{i-1}+1..2^i]$.

Takvih koraka ima ukupno k, te vreme izvršavanja algoritma je O(log n).

2. Konstruisati EREW algoritam efikasnosti O(1) čije vreme izvršavanja je O(log n) koji u vektoru A dimenzije n će vrednost A[1] smestiti na preostale lokacije.
REŠENJE:
Da bi se poravila efikasnost iz prethodnog zadatka korisiti se Brentova lema.
Algoritam je staticki ako se unapred zna pridruzivanje procesora operacijama.

Brentova lema: Ako postoji statički EREW algoritam složenosti T(n,p)=O(t(n)), takav da je ukupan broj koraka (na svim procesorima) s(n), onda postoji statički EREW algoritam složenosti T(n,s(n)/t(n))=O(t(n))

Pretpostavimo bez smanjenja opstosti da je $n=2^k$ i neka je na raspologanju $[n/\log_2 n]$, tj. $[2^k/k]$ procesora. Dalje, neka i je takvo da važi $2^{i-1}<2^k/k<2^i$.

Za i-1 korak se kopira A[1] na lokacije od $1..2^{i-1}$ (predstavljenim algoritmom).
Dalje, svaki od tih 2^{i-1} elemenata se paraleleno kopira uz 2^{i-1} procesora $2k/2^{i-1}-1$ puta (Procesor Pj kopira element sa lokacije j na lokacije $j+m*2^{i-1}$, m=1, 2, ..., $2^k/2^{i-1}-1$).
Kako je i-1 <k i
kako $2^k/2^{i-1}-1=2^{k-i+1}-1 < 2k-1 < 2\log_2 n$, to je ukupno vreme izvršavanja
i-1 korak + $2^k/2^{i-1}-1$ koraka
sto je prema vec navedenom manje od $\log_2 n + 2\log_2 n = 3\log_2 n$, tj. vreme izvršavanja je O(log n).

3.

Dat je n-dimenzioni niz A $(n = 2^k)$ i sistem sa n procesora koji funcionišu po modelu EREW. Opisati algoritam reda $O(\log n)$ koji u i-ti element niza $(1 < i \leq n)$ smešta vrednost $A[1] + i$.

REŠENJE:

U prvom koraku jedan procesor upisuje vrednost $A[1] + 2$ u lokaciju $A[2]$.
U drugom koraku učestvuju dva procesora i prvi upisuje vrednost $A[1] + 3$ u lokaciju $A[3]$, a drugi vrednost $A[2] + 2$ u $A[4]$.
...
U k-tom koraku učestvuje 2^{k-1} procesora i prvi procesor upisuje vrednost $A[1] + 2^{k-1} + 1$ u lokaciju $A[1 + 2^{k-1}]$, a j-ti procesor $(2 \leq j \leq 2^{k-1})$ upisuje vrednost $A[j] + 2^{k-1}$ u lokaciju $A[j + 2^{k-1}]$.
U opisanim koracima nikoja dva procesora ne pristupaju istoj memorijskoj lokaciji i algoritam se izvršava u modeli $EREW$.
Za niz od $n = 2^k$ elemenata ovakvih koraka ima $k = \log_2 n$, pa je složenost ovog algoritma $O(\log n)$.

4.

Dat je niz od n memorijskih lokacija i n procesora $(n = 2^k)$ koji funkcionišu po $EREW$ modelu. Lokacija $A[i]$ sadrži vrednost a_i, $1 \leq i \leq n$. Opisati algoritam sa vremenom izvršavanja $O(\log n)$ nakon čijeg izvršavanja lokacija $A[i]$ sadrži vrednost $\prod_{k=1}^{i} a_k$, za $1 \leq i \leq n$.

Resenje:

Algoritam kreiramo korišćenjem indukcije. Problem trivijalno rešavamo za $n = 1$. Neka je $n = 2^k$ i pretpostavimo da znamo da rešimo problem za $\frac{n}{2}$ elemenata. Najpre za svako i takvo da $1 \leq i \leq \frac{n}{2}$ izračunamo vrednost $a_{2i-1} * a_{2i}$ i rezultat upišemo u $A[2i]$. Sada imamo $\frac{n}{2}$ elemenata, pa po pretpostavci možemo rešiti dobijeni potproblem čime dobijemo tražene vrednosti za svako $A[2i]$. Kada to znamo, lako pronalazimo vrednosti $A[2i+1]$ $(1 \leq i < \frac{n}{2})$ na sledeći način: vrednost $A[2i] \cdot a_{2i+1}$ upišemo u $A[2i+1]$.

Rekurzivni poziv primenjujemo na $\frac{n}{2}$ elemenata, pa je složenost ovog algoritma

$$T(n) = T(\frac{n}{2}) + c\,,$$

odakle sledi

$$T(n) = O(\log n).$$

```
Algoritam PARALLEL_PREFIX (a,n);
input: a, n;
output: a;

begin
  PP(1);
end.

procedure PP(inc);
begin
  if (inc= n/2) then
    a[n]:=a[n/2]*a[n];
  else
    for i:=1 to n/(2*inc) do in parallel
      a[2*i*inc]:=a[2*i*inc-inc]*a[2*i*inc];
      PP(2*inc);
    for i:=1 to n/(2*inc)-1 do in parallel
      a[2*i*inc+inc]:=a[2*i*inc]*a[2*i*inc+inc];
end.
```

5.

Naci paralelni algoritam za izracunavanje vrednosti polinoma $a_0 + a_1x + \ldots + a_nx^n$ sa vremenom izvrsavanja O(log n) na racunaru sa n procesora koji funkcionise po EREW modelu ($n = 2^k$). Koeficijenti su smesteni u vektor duzine n + 1 u zajednickoj memoriji.

Uputstvo: Neka su procesorima P_i $(1 \leq i \leq n)$ pridružene memorijske lokacije v_i $(1 \leq i \leq n)$ respektivno. Algoritam koji izračunava vrednost $a_0 + a_1x + \ldots + a_nx^n$ u vremenu $O(\log n)$ može da ima sledeći oblik:

- u prvom koraku procesor P_1 čita vrednost argumenta x i upisuje da u lokaciju v_1; procesori P_i, $2 \leq i \leq n$ respektivno upisuju u memorijske lokacije v_i, $2 \leq i \leq n$ vrednost 1; potrebno vreme: $O(1)$;
- primenom algoritma za paralelni prefiks za množenje (videti zadatak 4) ili *broadcast* algoritma (videti zadatak 3) na lokacije v_i $(1 \leq i \leq n)$ njihov sadržaj postaje $x, x, x, \ldots, x$; potrebno vreme: $O(\log n)$;
- primenom algoritma za paralelni prefiks za množenje na lokacije v_i $(1 \leq i \leq n)$ njihov sadržaj postaje $x, x^2, x^3, \ldots, x^n$; potrebno vreme: $O(\log n)$;
- procesor P_i $(1 \leq i \leq n)$ čita vrednost a_i, množi je sadržajem lokacije v_i i rezultat upisuje u istu tu lokaciju; time sadržaj lokacija v_i $(1 \leq i \leq n)$ postaje $a_1x, a_2x^2, a_3x^3, \ldots, a_nx^n$; potrebno vreme: $O(1)$;
- primenom algoritma za paralelni prefiks za sabiranje na lokacije v_i $(1 \leq i \leq n)$ sadržaj lokacije v_n postaje $a_1x + \ldots + a_nx^n$; potrebno vreme: $O(\log n)$;
- procesor P_n čita vrednost a_0, sabira je sa sadržajem lokacije v_n i rezultat upisuje u istu tu lokaciju; time sadržaj lokacija v_n postaje $a_0 + a_1x + \ldots + a_nx^n$, što je vrednost koju je trebalo izračunati; potrebno vreme: $O(1)$.

6. Konstruisati algoritam za odredjivanje maksimalnog od datih n brojeva (ne nuzno razlicitih) na modelu CRCW, tako da vreme izvrsavanja algoritma bude O(1) i da koristi najvise n^2 procesora .

REŠENJE: Koristi se $n(n-1)/2$ procesora. Svakom od datih brojeva x_i pridružuje se po jedna (deljena) memorijska lokacija v_i. Svakom paru $\{i, j\}$ zadatih brojeva se pridružuje procesor P_{ij}, a najpre se u sve lokacije v_i upisuje vrednost 1. U sledećem koraku algoritama, procesor P_{ij} poredi elemente x_i i x_j; ako je jedan od njih manji onda u njegovu odgovarajuću lokaciju upisuje se vrednost 0; ako su x_i i x_j jednaki onda ako je $i < j$ onda se 0 upisuje u v_i, a inače se 0 upisuje u v_j. Nakon toga samo u jednoj od lokacija v_i ostaje upisana vrednost 1. Detektuje se kojem je zadatom elementu pridružena ta lokacija. Taj broj je maksimum zadatih vrednosti.

Nezavisno od n, algoritam zahteva dva koraka, dok je broj procesora koji se koristi $n(n-1)/2$. Otuda, vreme izvršavanja algoritma je $O(1)$ i on zahteva $O(n^2)$ procesora.

7. Konstruisati CRCW algoritam za izračunavanje disjunkcije n Bulovih promenljivih za vreme O(1).
REŠENJE:
Vrednost operacije disjunkcije nad n Bulovih promenljivih jednaka je maksimumu datih n vrednosti. Time se problem svodi na problem odredjivanje maksimalnog od datih n brojeva.

Koristi se n(n-1)/2 procesora.
Svakom od datih brojeva x[i] pridruzujemo se po jedna (deljena) memorijska lokacija v[i]. Svakom paru {i,j} zadatih brojeva pridruzuje se procesor P[ij].
Najpre se u sve lokacije v[i] upisuje vrednost 1.
U sledećem koraku algoritma, procesor P[ij] poredi elemente x[i] i x[j];
Ako je jedan od njih manji onda u njegovu odgovarajuću lokaciju upisuje se vrednost 0;
ako su x[i] i x[j] jednaki onda ako je i<j onda se 0 upisuje u v[i], a inace se 0 upisuje u v[j].
Nakon toga samo u jednoj od lokacija v[i] ostaje upisana vrednost 1.
Otkrije se kom zadatom elementu je pridružena ta lokacija i taj broj je maksimum zadatih vrednosti.

Vreme izvršavanja algoritma je O(1) i on zahteva $O(n^2)$ procesora.

8. Konstruisati CREW algoritam za sortiranje niza različitih brojeva $x_1, x_2, \ldots, x_n$. Vremenska složenost treba da bude O(log n) na paralelnom CREW računaru sa zajedničkom memorijom i dovoljnim brojem procesora.

Resenje:
Posto je na raspolaganju dovoljan broj procesora, a dozvoljeno je istovremeno citanje sa iste lokacije, moze se najpre formirati matrica *A* sa elementima
A[i][j]=1, ako $x_i<x_j$
A[i][j]=0, ako $x_i>x_j$

za sta je potrebno $O(1)$ paralelnih koraka. Rang i_r elementa x_r (pozicija x_r u sortiranom redosledu) jednak je broju elemenata x_j manjih od x_r uvecanom za jedan, odnosno $s_r + 1$, gde je s_r zbir elemenata r-te vrste matrice A.
Pomocu n^2 procesora zbirovi vrsta u A mogu se paralelno izracunati za $O(\log n)$ koraka (posebnim turnirom za svaku vrstu). Zatim se u jednom paralelnom koraku x_r kopira na poziciju i_r izlaznog vektora, $1 <= r <= n$.

9. Konstruisati CRCW algoritam za objedinjavanje dva niza brojeva A i B u jedan sortiran niz. Poredak sortiranja je istovetan kod sva tri niza. Vremenska složenost treba da bude O(1). Na raspolaganju je neograničen i memorijski prostor i broj procesora.
Resenje:

Ako imamo dovoljno procesora i memorijskog prostora treba da napravimo *merge* od dva sortirana niza $a_1, \ldots, a_n$, $b_1, \ldots, b_m$ u vremenu $O(1)$.

Za ovo rešenje trebaće nam $2 \cdot n \cdot m + n + m$ procesora pri čemu ćemo za svaki element a_i niza a imati $m+1$ procesora $P_{i,j}$ $(j = 0, ..., m)$, a svaki element b_j ce obradjivati $n+1$ procesora $Q_{i,j}$ $(i = 0, ..., n)$. Od nizova $a_1, ..., a_n$ i $b_1, ..., b_m$ napravićemo sortirani niz $c_1, ..., c_{n+m}$.

Procesori koji obradjuju (upisuju) elemente niza a radiće sledeće :

* procesori $P_{i,j}$ za $1 \leq i \leq n$, $1 \leq j < m$ rade :
  if $(b_j < a_i \leq b_{j+1})$ then $c_{i+j} := a_i$ (jer se u nizu c pre a_i nalazi $i-1$ elemenata niza a i j elemenata niza b)

* procesori $P_{i,0}$ za $i = 1, ..., n$ rade :
  if $(a_i \leq b_1)$ then $c_i := a_i$

* procesori $P_{i,m}$ za $i = 1, ..., n$ rade :
  if $(b_m < a_i)$ then $c_{m+i} := a_i$

Procesori koji obradjuju (upisuju) elemente niza b radiće sledeće :

* procesori $Q_{i,j}$ za $1 \leq i < n$, $1 \leq j \leq m$ rade :
  if $(a_i \leq b_j < a_{i+1})$ then $c_{i+j} := b_j$

* procesori $Q_{0,j}$ za $j = 1, ..., m$ rade :
  if $(b_j < a_1)$ then $c_j := b_j$

* procesori $Q_{n,j}$ za $j := 1, ..., m$ rade :
  if $(a_n \leq b_j)$ then $c_{n+j} := b_j$

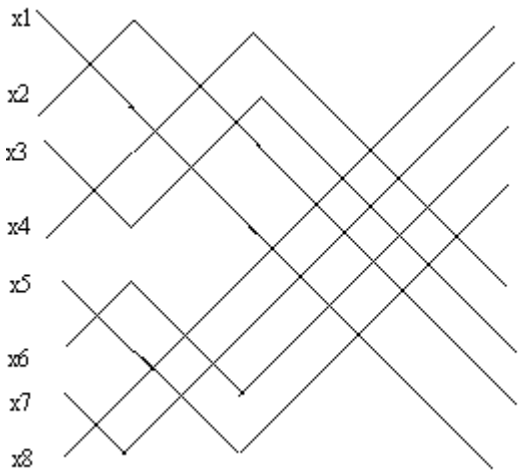
Resenje zadatka se zasniva na činjenici da svaki realan broj pripada tacno jednom od intervala

$$(-\infty, a_1), [a_1, a_2), [a_2, a_3) ... [a_{n-1}, a_n), [a_n, +\infty)$$

Pošto svaki procesor radi samo jedan korak vreme izvršavanja ovog CRCW algoritma je zaista O(1).

Primetimo sledeće: primenom algoritma u jednom trenutku više procesora čita sadržaj iste lokacije, ali samo jedan procesor upisuje u jednu lokaciju u jednom trenutku, pa bi opisani algoritam bio primenljiv i u CREW modelu. Takodje primetimo da algoritam radi ispravno i ako ne važi uslov da u objedinjenom nizu nema jednakih elemenata. U slučaju pojavljivanja istih elemenata u nizovima a i b, u niz c se na niže pozicije upisuju elementi niza a.

10. Pod pretpostavkom da $n=2^k$ osoba znaju neke informacije koje žele da razmene među sobom i pod pretpostavkom da u svakom koraku svaka osoba može da komunicira sa tačno jednom osobom i razmeni sve informacije koje poseduje, konstruisati algoritam koji kreira redosled komunikacije tao da nakon $\log_2 n$ koraka svaka osoba poseduje sve informacije.
Resenje:
Za n=8 sledeća shema ilustruje traženu komunikaciju:



(Ova shema naziva se *leptir* i jedna od primena joj je i u paralelnom izračunavanju FFT.)

Nije teško pokazati da opisana komunikacija zahteva $\log_2 n$ koraka i da nakon nje sve osobe raspolažu svim informacijama.

Redosled komuiciranja moze se konstruisati indukcijom.
Za k = 1 resnje je trivijalno.
Ako se zna resenje za $n = 2^k$ osoba, onda se $2n = 2^{k+1}$ osoba dele u dve grupe {1, 2,..., n} i {n + 1, n + 2,..., 2n}, u okviru kojih se po induktivnoj hipotezi za k koraka mogu razmeniti svi tracevi.
Zatim u (k +1)-om koraku razmenjuju traceve osoba i sa osobom i + n, 1<=i<=n,
posle cega svih 2n osoba znaju sve traceve.