nedeljainformatike v9.0

Paralelni algoritmi

Mihajlo Marković

Matematička gimnazija

13. 05. 2024.

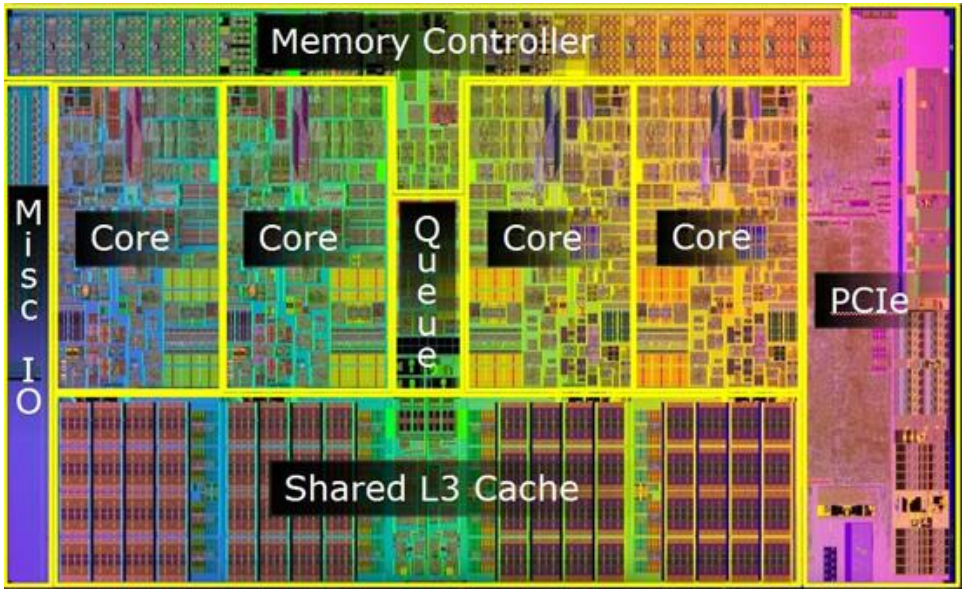
Agenda

1. Teorijski uvod

2. Paralelni algoritmi

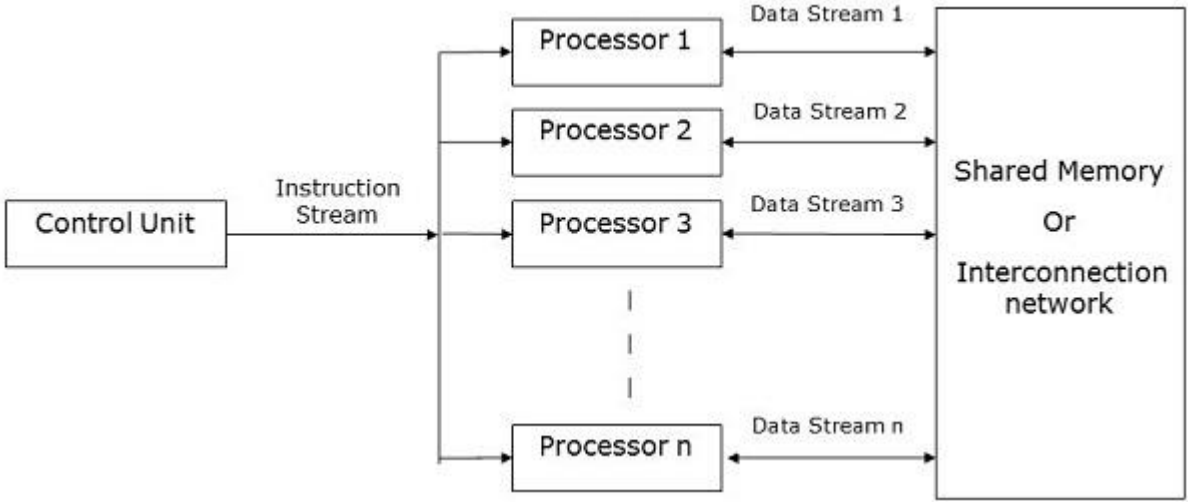
Motivacija

- Murov zakon: Broj tranzistora u integrisanom kolu koje smo u stanju da proizvedemo će se udvostručiti na svakih 18-24 meseci.
- Dodatni tranzistori su korišćeni da ubrzaju rad procesora (složenije operacije, komunikacija između delova memorije...)
- Od 2004. godine procesori se više nisu mogli ubrzati novodobijenim tranzistorima
- Višejezgarni procesori
- Multiprocesorski sistemi



Šta su paralelni algoritmi?

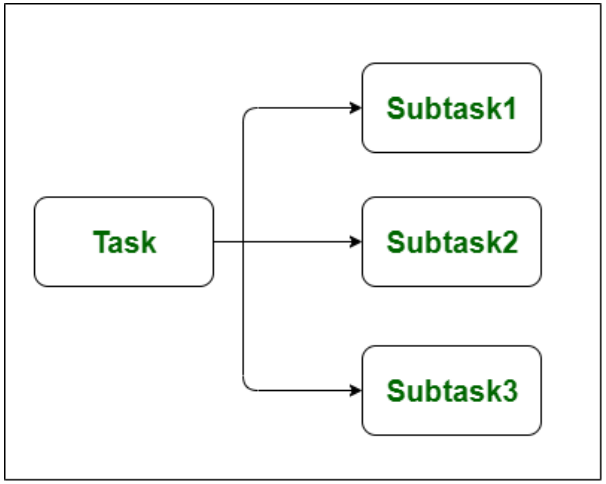
- Paralelni algoritmi su algoritmi koji se mogu izvršavati deo po deo na više procesora istovremeno.
- Dobijeni međurezultati delova algoritma se koriste u narednim koracima paralelnog algoritma.
- Po završetku izvršavanja, dobijeni rezultati za svaki procesor se spajaju u cilju dobijanja konačnog rezultata.



Paralelno ≠ konkurentno programiranje

- Paralelno programiranje podrazumeva da se jedan ili više zadataka obavljaju istovremeno na više procesorskih jedinica, po delovima.
- Koristi se u cilju popravljanja vremenske složenosti.

- Konkurentno programiranje podrazumeva da se više zadataka zajedno obavlja na jednoj (ili više) procesorskoj jedinici.
- Omogućava povećanje količine posla koja se može obaviti sa raspoloživim resursima u nekom vremenskom periodu.



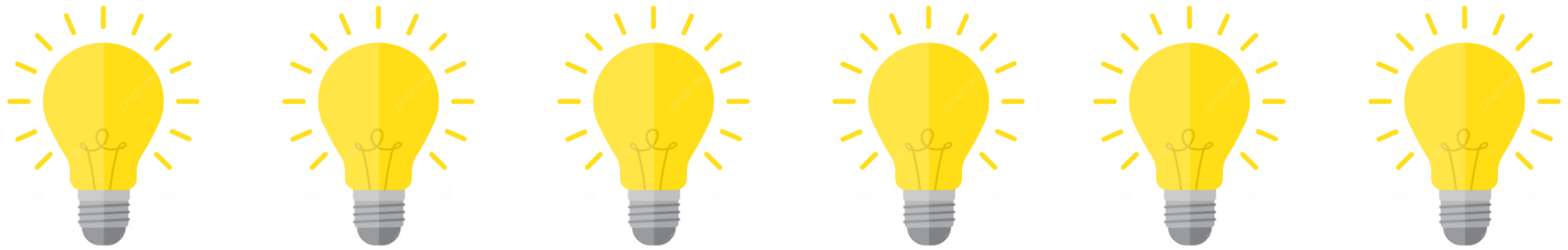
Pitalica – potraga za skrivenom mačkom

- N kutija je poređano u niz. U jednoj od kutija se krije mačka.
- Svake noći mačka se premešta u kutiju koja je za jedno mesto levo ili u kutiju koja je za jedno mesto desno u odnosu na trenutnu (ako postoji).
- Ne može nikad ostati u istoj kutiji!
- Svakog dana imate jedan pokušaj da pogodite u kojoj kutiji se nalazi mačka.
- Pronađite optimalnu strategiju kojom sigurno pronalazite mačku posle konačnog (poželjno što manjeg) broja dana (ili utvrdite da je to nemoguće).

Pitalica – sijalice i prekidači

- Dato je 1000 sijalica i 1000 prekidača (numerisani brojevima 1, 2, ..., 1000).
- i-ti prekidač je povezan sa sijalicama i, 2i, 3i...
- Kada se prekidač pritisne menja stanje svih sijalica sa kojima je povezan.
- Na početku su sve sijalice ugašene.
- Pritiskamo redom prekidače 1, 2, 3, ..., 1000.
- Koliko će sijalica biti upaljeno nakon što svaki prekidač pritisnemo po jednom?

- Šta se dešava ako sve prekidače pritisnemo istovremeno?



Osnovni pojmovi

- Task (zadatak) – logička celina koja predstavlja deo programa
- Balansiranje opterećenja – raspoređivanje posla po procesorskim jedinicama tako da se maksimizuje efikasnost
- Sinhronizacija – postupak kojim se obezbeđuje optimalan redosled izvršavanja taskova
- Tipovi paralelizacije:
  - Paralelizam na nivou bita
  - Paralelizam na nivou instrukcije
  - Paralelizam na nivou zadatka
- Flinova taksonomija (podela računarskih sistema):
  - SISD (Single Instruction, Single Data)
  - SIMD (Single Instruction, Multiple Data)
  - MISD (Multiple Instruction, Single Data)
  - MIMD (Multiple Instruction, Multiple Data)

Tehnike paralelizacije

- Paralelizacija ciklusa
- Paralelizacija višestrukih ciklusa
- Paralelizacija funkcija

- Zavisnost podataka
- Zavisnost resursa
- Zavisnost redosleda
- Zavisnost kontrole

- Tipovi zavisnosti:
  - Prava zavisnost (RAW – Read After Write)

    $A\colon X[i] = 3 \qquad B\colon Y[i] = X[i] + 5$
  - Anti-zavisnost (WAR – Write After Read)

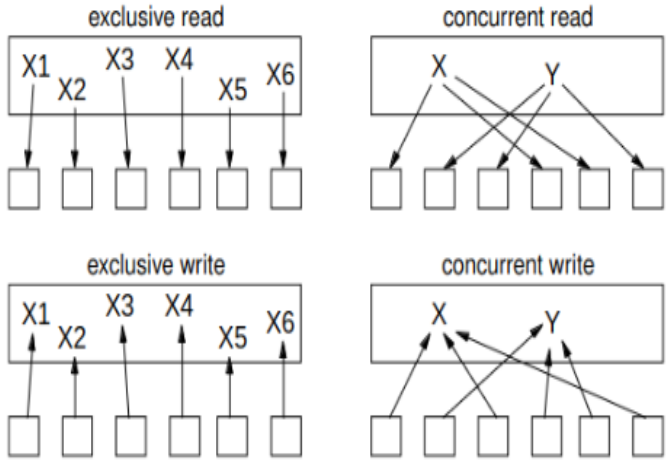
    $A\colon Y[i] = X[i] + 3 \qquad B\colon X[i] = 5$
  - Izlazna zavisnost (WAW – Write After Write)

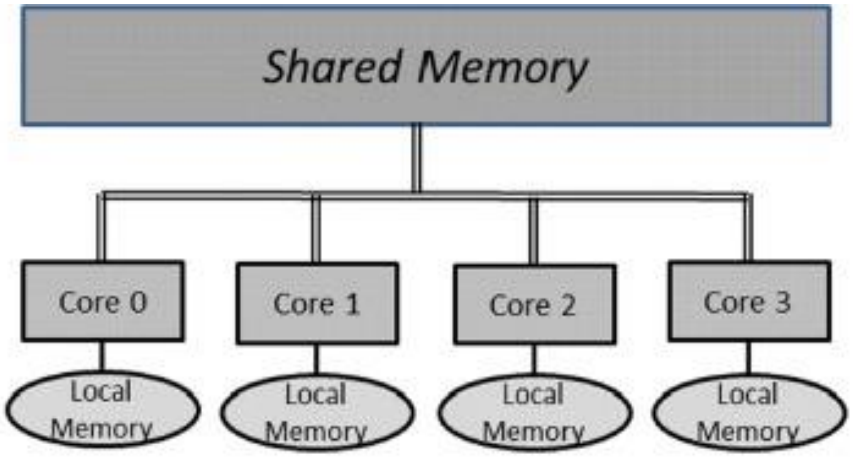
    $A\colon X[i] = 3 \qquad B\colon X[i] = 5$

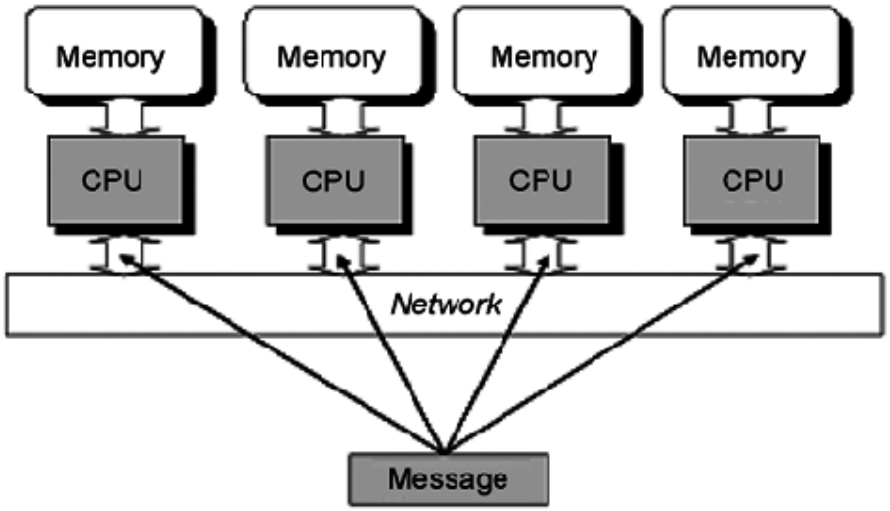
Različiti modeli paralelnih računara

- Paralelni računari sa zajedničkom memorijom
- Mreža računara sa sistemom za prenos poruka

- Discipline pristupa zajedničkoj memoriji:
  - EREW (exclusive–read, exclusive–write)
  - CREW (concurrent–read, exclusive–write)
  - CRCW (concurrent–read, concurrent–write)







Performanse paralelnih algoritama

- **Vreme izvršavanja algoritma na jednom procesoru:**
$$T(1) = T_{priprema} + T_{obrada} + T_{finalizacija} + T_{np}$$

- **Vreme izvršavanja paralelnog algoritma na P procesora:**
$$T(P) = T_{priprema} + \frac{T_{obrada}}{P} + T_{finalizacija} + T_{np}$$

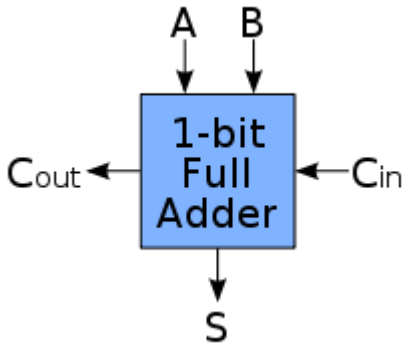
- **Ubrzanje paralelnog algoritma:** $\mathrm{S(P)} = \frac{T(1)}{T(P)}$
- **Efikasnost paralelnog algoritma:** $E(P) = \frac{S(P)}{P} = \frac{T(1)}{P \cdot T(P)}$

- **Sekvencijalni deo programa:** $\gamma = \frac{T_{priprema} + T_{finalizacija} + T_{np}}{T(1)}$
- **Amdalov zakon: Za konstantnu veličinu problema važi** $S = \frac{1}{\gamma + \frac{1-\gamma}{P}}$
- **Gustafsonov zakon: Koliko možemo povećavati veličinu problema sa povećanjem broja procesora?** $S = P + (1 - P)\gamma$

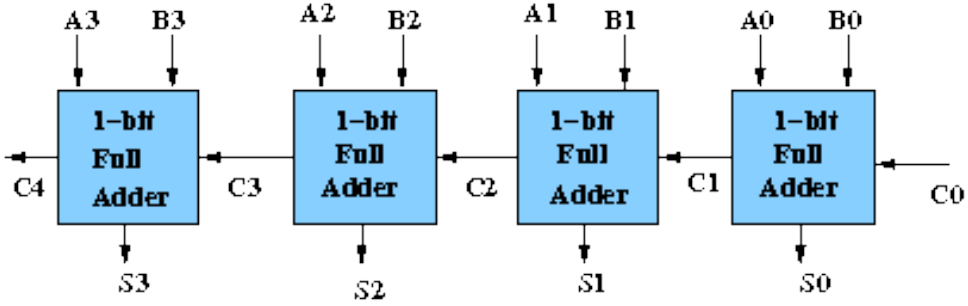
Sabirač

- **Jedan razred sabirača:**
- $F_i = A_i \oplus B_i \oplus C_i$
- $C_{i+1} = A_iB_i + (A_i+B_i)C_i$

A_i	B_i	C_i	F_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



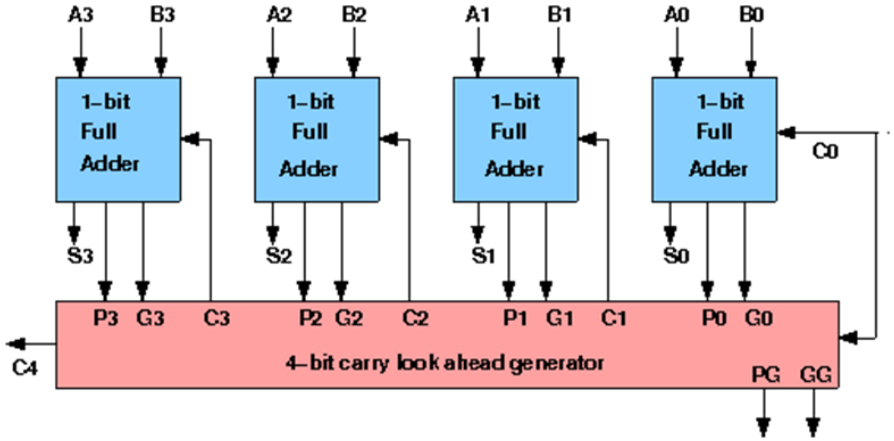
- **Četvorobitni sabirač sa kaskadnim generisanjem prenosa**



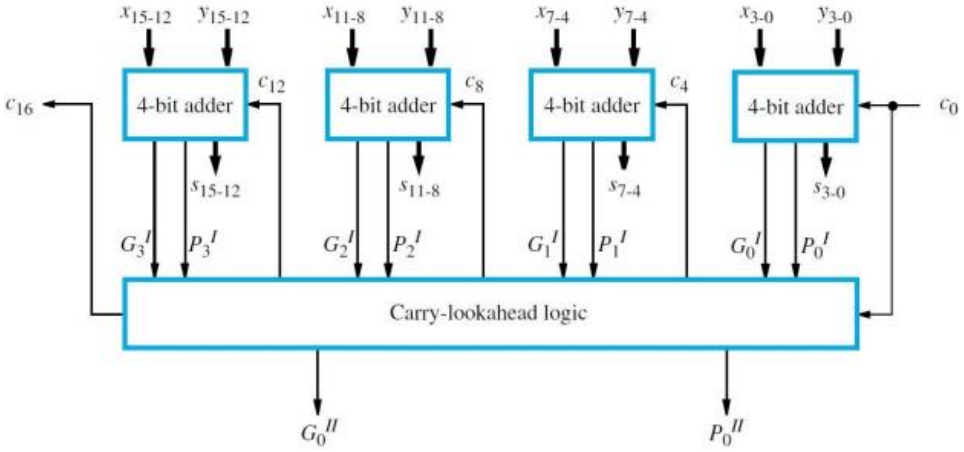
- **Kako paralelno generisati prenose?**
- $C_{i+1} = A_iB_i + (A_i+B_i)C_i$
- $G_i = A_iB_i$
- $P_i = A_i + B_i$
- $C_{i+1} = G_i + P_iC_i$
- $C_1 = G_0 + P_0C_0$
- $C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0)$
  $= G_1 + P_1G_0 + P_1P_0C_0$
- $C_3 = G_2 + P_2C_2$
  $= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

Paralelni sabirači

- Četvorobitni sabirač sa paralelnim generisanjem prenosa



- 16-bitni sabirač sa paralelnim generisanjem prenosa realizovan pomoću četiri četvorobitna sabirača



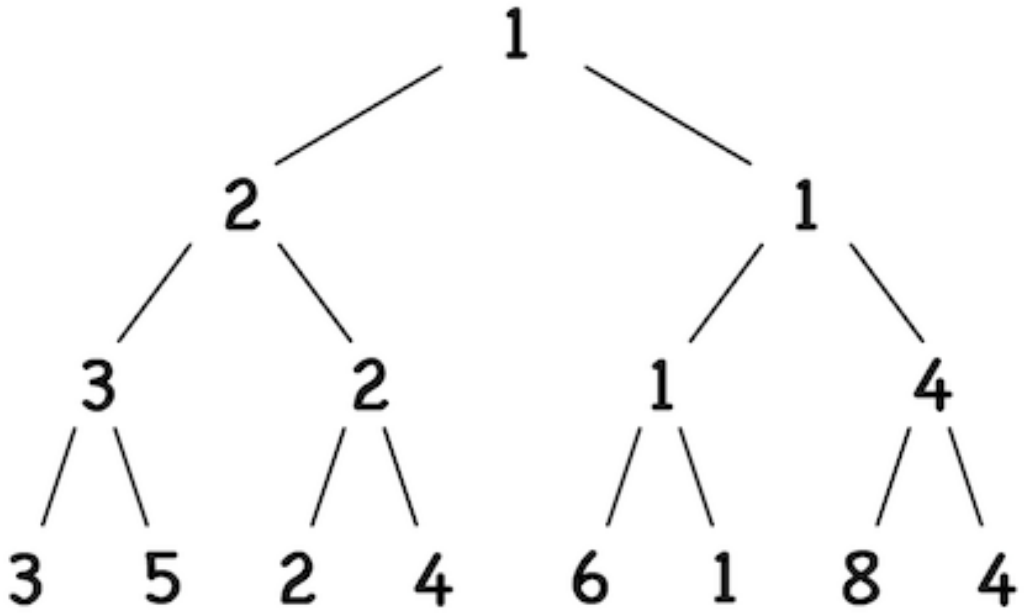
Agenda

1. Teorijski uvod

2. Paralelni algoritmi

Nalaženje minimuma

- **Ideja za EREW model – *tournament tree***
- $T\left(N, \frac{N}{2}\right) = \log_2 N$
- $E\left(N, \frac{N}{2}\right) = \frac{N-1}{\frac{N}{2}\log_2 N} \approx \frac{2}{\log_2 N}$
- **Kako da popravimo efikasnost?**
- **Smanjićemo broj procesora na** $\frac{N}{\log_2 N}$
- $T\left(N, \frac{N}{\log_2 N}\right) = 2\log_2 N$
- $E\left(N, \frac{N}{\log_2 N}\right) = \frac{N-1}{\frac{N}{\log_2 N} 2\log_2 N} \approx \frac{1}{2}$



Nalaženje minimuma

- Ideja za CRCW model – svaki par elemenata (a_i, a_j) dobija svoj procesor $P_{i,j}$
- Svaki element a_i dobija promenljivu v_i koja pripada deljenoj memoriji ($v_i = 0$)
- Procesor $P_{i,j}$ poredi a_i i a_j i većem elementu upisuje 1 u deljenu memoriju
- Na kraju samo jedan element u deljenoj memoriji ima vrednost 0
- Drugi korak: signalizirati gde je minimum
- Na primer, ako je indeks minimuma k svi procesori $P_{k,j}$ i $P_{i,k}$ u drugom koraku upisuju u deljenu promenljivu res vrednost (ili indeks) minimalnog elementa
  - $T\left(N, \frac{N(N-1)}{2}\right) = 2$ $\quad\quad$ $\mathrm{E}\left(N, \frac{N(N-1)}{2}\right) = \frac{N-1}{\frac{N(N-1)}{2}.2} = \frac{1}{N}$

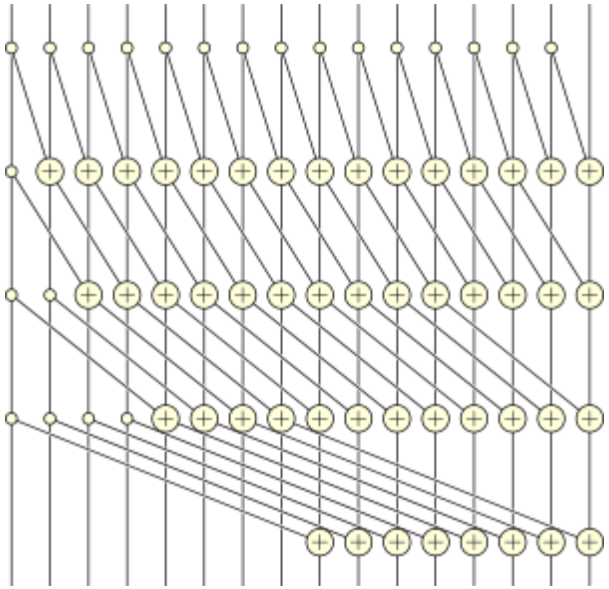
- Popravljanje efikasnosti: koristimo N procesora
- U k-tom koraku delimo niz od preostalih $\frac{N}{2^{2^{k-1}-1}}$elemenata na $\frac{N}{2^{2^{k-1}}}$ grupa od po $2^{2^{k-1}}$ elemenata.
- Tražimo minimum svake grupe prethodnim algoritmom
  - $T(N,N) = O(\log_2 \log_2 N)$ $\quad\quad$ $\mathrm{E}(N,N) = \frac{N-1}{N \log_2 \log_2 N} = O(\frac{1}{\log_2 \log_2 N})$

Prefiksne sume

- **Data je binarna asocijativna operacija $*$ i niz $x_1, x_2 \dots x_n$**
- **Za svako $k \in \{1,2 \dots n\}$ naći $p_k = x_1 * x_2 * \cdots * x_k$**
- **Sekvencijalno rešenje: $p_k = p_{k-1} * x_k$. Vremenska složenost: $O(N)$**

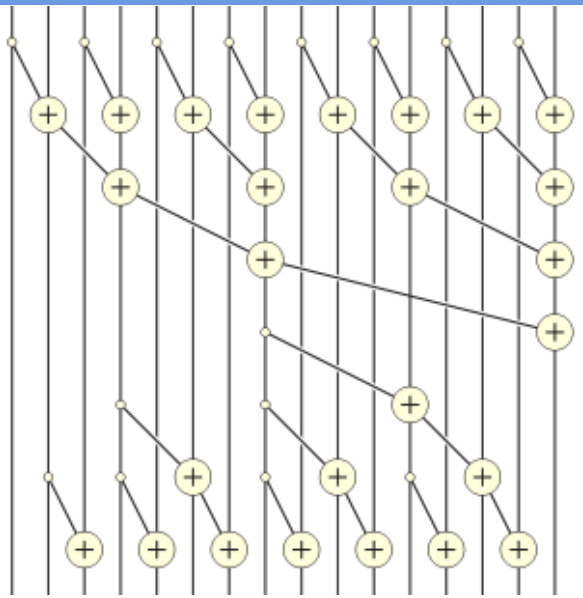
- **Paralelno rešenje 1:**

```
i=1,k=1
While i<n:
    parallel for j in range (1,n):
        if j>i:
            x[j][k]=x[j][k-1]+x[j-i][k-1]
        else:
            x[j][k]=x[j][k-1]
    k+=1,i*=2
```



- **Vremenska složenost sa N raspoloživih procesora: $O(\log_2 N)$**

Prefiksne sume



- Paralelno rešenje 2:
- Formiramo Fenvikovo stablo
- Vremenska složenost sa N procesora: $O(\log_2 N)$

- Šta se dešava ako imamo $\frac{N}{\log_2 N}$ procesora?
- U prvom koraku delimo niz na $\frac{N}{\log_2 N}$ delova dužine $\log_2 N$ (isto stepen dvojke!)
- Za svaki deo je zadužen po jedan procesor (potrebno vreme $O(\log_2 N)$)
- Svi čvorovi su dobili konačne vrednosti, osim $\frac{N}{\log_2 N}$ krajeva intervala
- Paralelnim algoritmom 2 formiramo Fenvikovo stablo nad nizom od $\frac{N}{\log_2 N}$ elemenata (i imamo toliko procesora na raspolaganju, pa je vreme $O(\log_2 N)$)
- U poslednjem koraku za svaki element treba izračunati prefiksnu sumu korišćenjem Fenvikovog stabla ($O((\log_2 N)^2)$ jer nemamo dovoljno procesora)

Paralelno pretraživanje

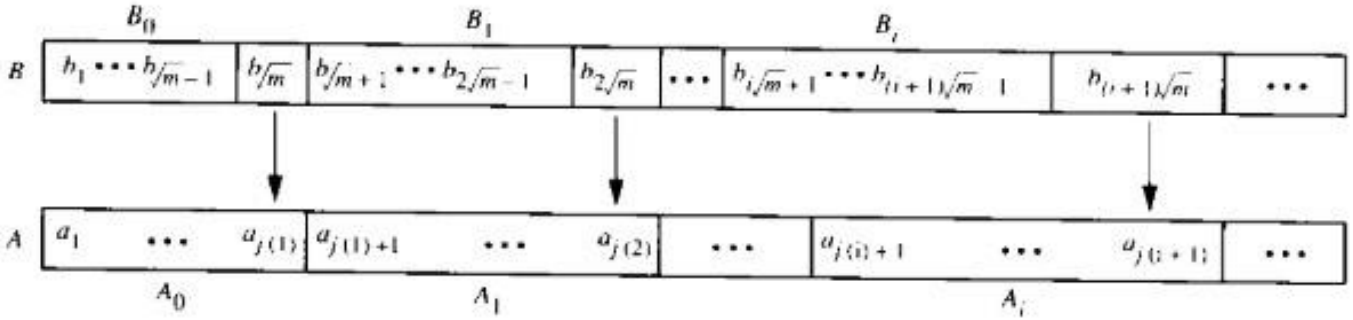
- **Jedan procesor – binarna pretraga** $O(\log_2 N)$
- **Da li može brže sa P procesora?**
- **Trivijalno rešenje: podelimo niz na P segmenata i svaki procesor radi binarnu pretragu nad segmentom za koji je zadužen**
  - **Vremenska složenost:** $O(\log_2 \frac{N}{P} + \log_2 P)$
- **Bolje rešenje: podelimo niz na** $P + 1$ **segmenata, približno jednake dužine.**
- **U prvom koraku i-ti procesor poredi element** $A[i \cdot \frac{N}{P+1}]$ **sa elementom** x**.**
- **Zatim binarna pretraga nad P dobijenih rezultata**
- **Novi segment delimo na** $P + 1$ **delova...**
  - **Vremenska složenost:** $O\left(\log_{(P+1)} N\right) = O\left(\frac{\log_2 N}{\log_2 (P+1)}\right)$

Priprema za merge sort

- **Data su dva rastuće uređena niza:** $a_1, a_2 \dots a_n$ **i** $b_1, b_2 \dots b_m$
- **Za svaki element niza** b **naći na kojoj bi se poziciji našao u nizu** a

- **Ako imamo M·P procesora, koristimo po P procesora za pretragu svakog elementa iz niza** b **u nizu** a
- **Vremenska složenost:** $O\left(\frac{\log_2 N}{\log_2(P+1)}\right)$ **Efikasnost:** $E(N, M, MP) = \frac{N+M}{MP\left(\frac{\log_2 N}{\log_2(P+1)}\right)}$

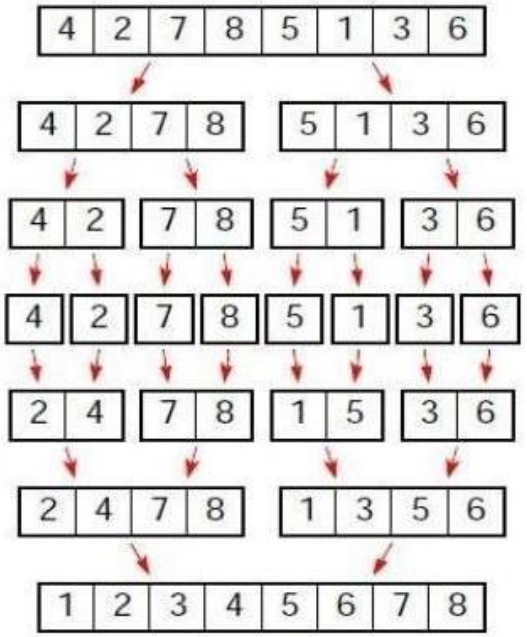
- **Bolje rešenje: koristimo** $\sqrt{NM}$ **procesora**
- **U prvom koraku za svaki od elemenata** $b_{i\sqrt{m}}$ **nalazimo njegov rang u** a **paralelnim pretraživanjem korišćenjem** $\sqrt{N}$ **procesora (složenost** $O(1)$**)**
- **Zatim ovo ponavljamo u svakom bloku dužine** $\sqrt{M}$**...**
- **Primer:** $2^{16} \to 2^8 \to 2^4 \to 2^2$
- **Vremenska složenost:** $O(\log_2 \log_2 M)$ **Ukupno operacija:** $O((\mathrm{N} + \mathrm{M})\log_2 \log_2 M)$



Spajanje dva sortirana niza i merge sort

```
parallelMerge(a,b,n,m):
    u=nadjiRang(a,b,n,m) #niz rangova elemenata niza b u nizu a
    v=nadjiRang(b,a,m,n) #niz rangova elemenata niza a u nizu b
    parallel for i in range (0,n):
        c[i+v[i]]=a[i]
    parallel for i in range(0,m):
        c[i+u[i]]=b[i]
```

```
parallelMergeSort(a,l,r):
    if r-l>0 do in parallel:
        parallelMergeSort(a,l,(l+r)/2)
        parallelMergeSort(a,(l+r)/2+1,r)
    parallelMerge(a[l:(l+r)/2],a[(l+r)/2+1:r])
```



- **Vremenska složenost:** $O((\log_2 N)^2)$

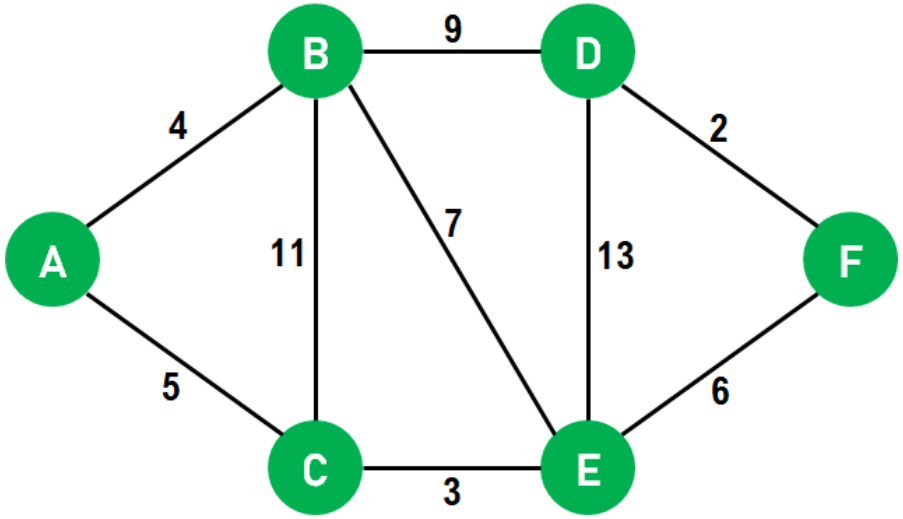
Paralelni quick sort

```
parallelQuickSort(a,l,r):
    p=a[r]
    parallel for i in range(l,r):
        if a[i]<=p:
            m[i]=1, v[i]=0
        else:
            m[i]=0, v[i]=1
    do in parallel:
        parallelPrefixSum(m,pm,l,r)
        parallelPrefixSum(v,pv,l,r)
    parallel for i in range(l,r):
        if a[i]<=p:
            b[l+pm[i]-1]=a[i]
        else:
            b[l+pm[r]+pv[i]-1]=a[i]
    parallel for i in range(l,r):
        a[i]=b[i]
    do in parallel:
        parallelQuickSort(a,l,l+pm[r]-2)
        parallelQuickSort(a,l+pm[r],r)
```

- **Prosečna vremenska složenost:** $O((\log_2 N)^2)$

Paralelni Dajkstrin algoritam

- Na početku postavljamo rastojanja svih čvorova od početnog čvora na beskonačno.
- Svaki put od preostalih neposećenih čvorova uzimamo onaj sa minimalnim rastojanjem od početnog čvora.
- Ažuriramo rastojanja njemu susednih neposećenih čvorova.



- Šta se ovde može paralelizovati?
- Ako imamo N^2 procesora, minimalna rastojanja možemo tražiti u $O(1)$
- Vremenska složenost: $O(N)$

Hvala na pažnji!

Pitanja?