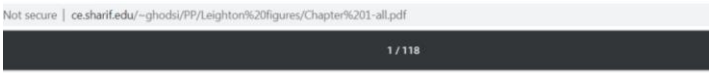
به نام خدا

پردازش موازی جلسه دوم

توپولوژی آرایه خطی

مثال: یک توپولوژی آرایه خطی داریم، n عدد به آن وارد می‌شود.



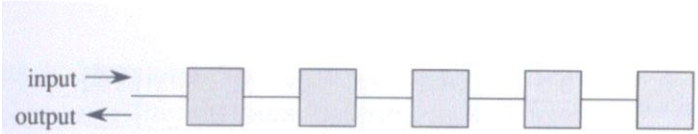


Figure 1-1 *A 5-cell linear array.*

پارامترهای توپولوژی:

- قطر: فاصله‌ی دورترین nodeها در گراف
  - کران پایین زمان است.
  - برای آرایه‌ی خطی n-1 است.
- پهنای باند دوبخشی (bisection width)
  - با b نشان می‌دهند.
  - حداقل تعداد یالهای گراف که باید قطع کنیم که گراف دو قسمت (با تعداد رأس مساوی) شود.
  - کران بالای تعداد پیام‌های مبادله شده در واحد زمان.

در آرایه خطی، b=1 است. در مثال sort، n-1 کران پایین است، یعنی اگر با این زمان حل کنیم بهینه است.

ce.sharif.edu/~ghodsi/PP/Leighton%20figures/Chapter%201-all.pdf

10 / 118

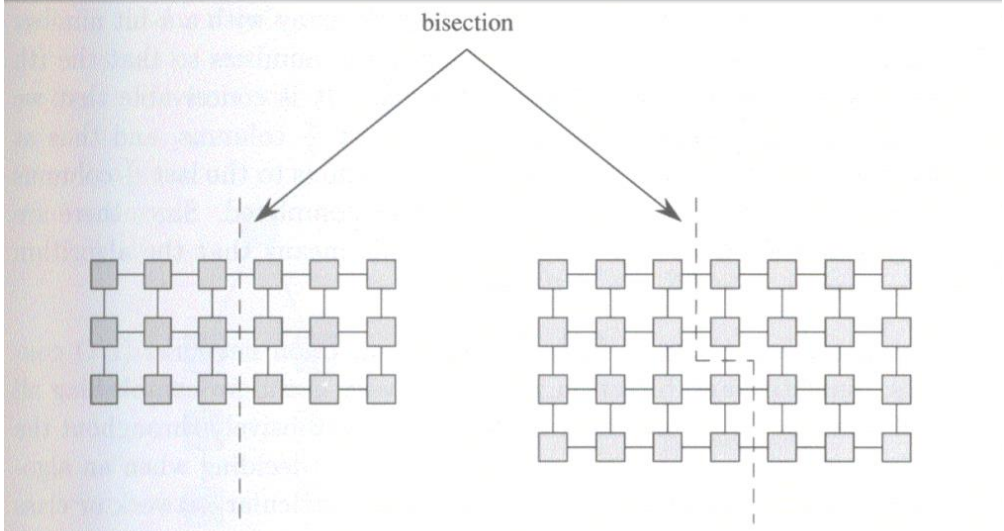


Figure 1-10 *Minimum bisections of a* 3×6 *array and a* 4×7 *array. The* 3×6 *array has bisection width* 3 *and the* 4×7 *array has bisection width* 5.

مسئله را داریم کلمه‌ای حل می‌کنیم یا بیتی؟

- Word: پردازنده‌ای که می‌تواند عملیات پیشرفته مانند جمع و ... در هر کلاک انجام دهد.
- Bit: اعمال بیتی در هر کلاک

کلاک در مدل word کندتر است.

ce.sharif.edu/~ghodsi/PP/Leighton%20figures/Chapter%201-all.pdf

4 / 118

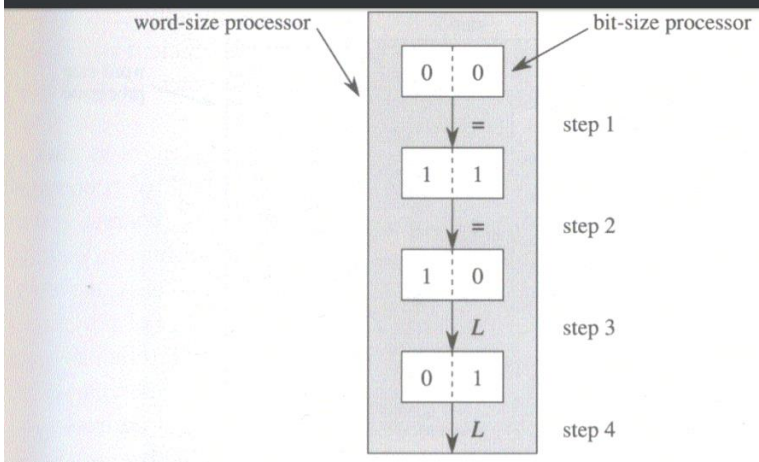


Figure 1-4 *Comparison of* 0110 *and* 0101 *on a* 4*-cell linear array. The notation* L *is used to denote that the number on the left-hand side is bigger.*

از نظر سخت‌افزاری، خروجی عملیاتی که انجام می‌دهد در بافر همسایه می‌نویسد.

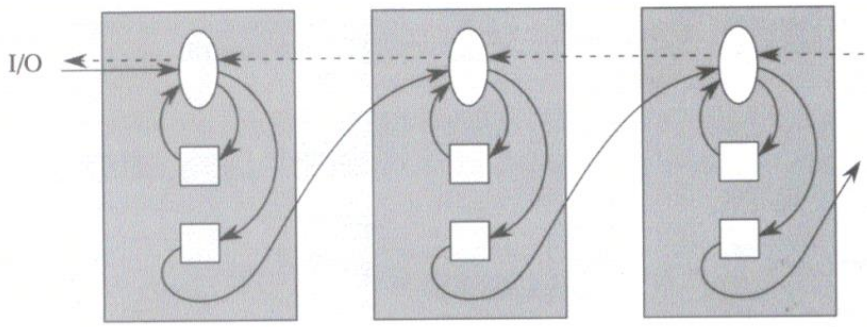


Figure 1-57 *Semisystolic implementation of the algorithm for palindrome recognition. The upper register in each processor is used to store the first value seen. The lower register holds subsequent values before they are passed rightward. Dashed wires are used for the accumulate signal.*

مثال: Sort آرایه خطی با داده خارجی. داده‌ی ورودی را اگر نداشت ذخیره می‌کند، وگرنه با خودش min می‌گیرد و min را در خودش نگه می‌دارد و max را به سمت راستی می‌دهد.

مثالی از یک عمل مختلف روی داده‌های مختلف (SIMD) است.

زمان مرتب‌سازی آرایه خطی:

- بهترین حالت: n-1
- بدترین حالت: 2n-1: عدد سمت چپ‌ترین در سمت‌راست‌ترین پردازنده باشد.

این در مدل کلمه‌ای بود. حالا فرض کنید بیتی باشد (رجیسترها k-بیتی باشند).

اگر کمتر از n پردازنده داشته باشیم، هر کدام به اندازه‌ی n/k کار پردازنده‌ی قبلی را انجام می‌دهند.

ce.sharif.edu/~ghodsi/PP/Leighton%20figures/Chapter%201-all.pdf

3 / 118

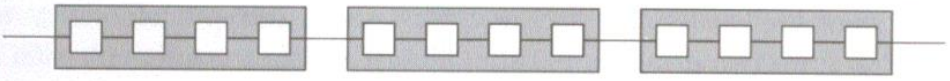


Figure 1-3 *Simulating a 12-processor linear array G_1 on a 3-processor linear array G_2. Each processor in G_2 is responsible for simulating four processors of G_1. Hence, every step of G_1 takes four steps on G_2, and any algorithm that runs in T steps on G_1 can be made to run in $4T$ steps on G_2.*

هر ستون یک عدد است. نتیجه‌ی مقایسه‌ی هر زوج بیت = یا L یا R است (رقم کدام پردازنده بیشتر بوده است).

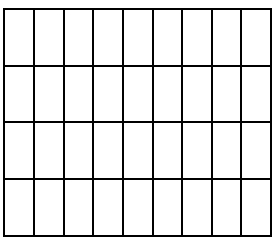
بیت پرارزش را اول مقایسه می‌کنیم. نتیجه بیت پرارزش اگر غیر از = باشد به مرحله بیت بعد می‌رود.

	پردازنده ۱			پردازنده ۲		
بیت پرارزش	0	0	=			
	1	1	=			
	1	0	L			...
بیت کم‌ارزش	0	1	R			
	عدد اول	عدد دوم		عدد سوم	عدد چهارم	

فرض کنید هر پردازنده دو تا عدد داشته باشد.

زمان O(kn) است.

در واقع این معادل یک مدل توری (grid) با k سطر و n ستون است (فرض کنید هر تقاطع یک پردازنده است).



قطر=n+k (کران پایین زمان)

پس (احتمالاً) می‌توانیم الگوریتم را بهتر کنیم.

بهبود: k را به log k مرحله تبدیل کنیم. (مقایسه‌ی بیتی)

هر کسی با سمت راستی مقایسه کند. ساختار درختی به کار ببریم.

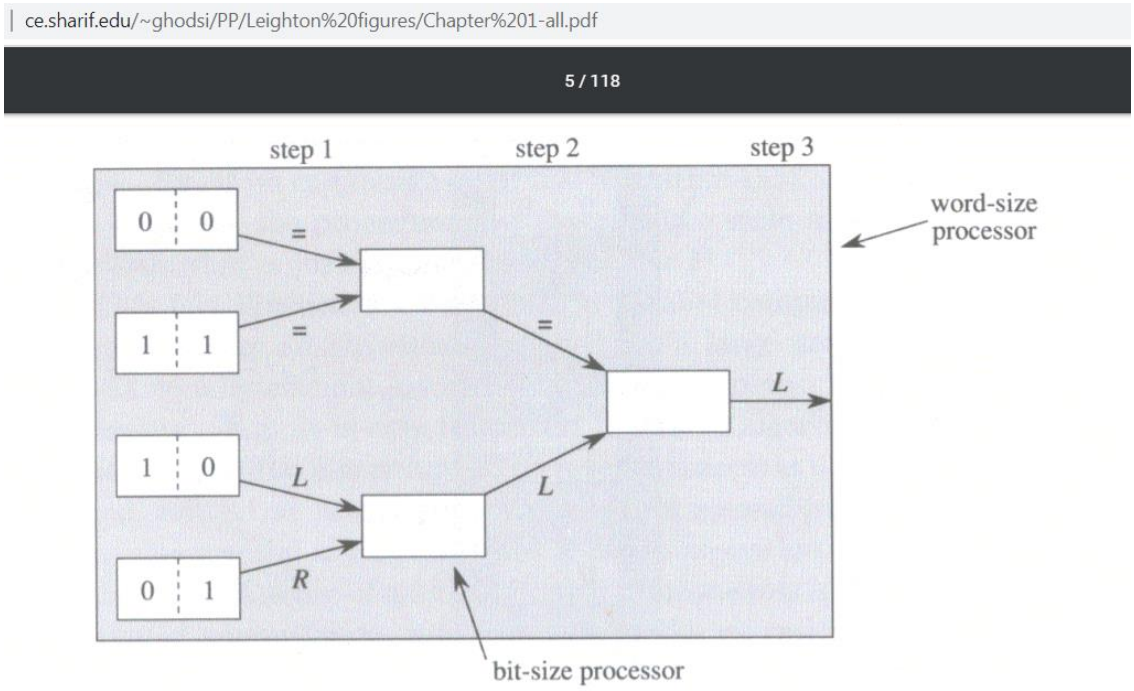


Figure 1-5 *Comparison of* 0110 *and* 0101 *on a 4-leaf binary tree. The notation* L/R *is used to denote which of the numbers covered by a node is bigger.*

2log k زمان نیاز دارد، چون log k مرحله هم برای برگشت جواب لازم داریم:

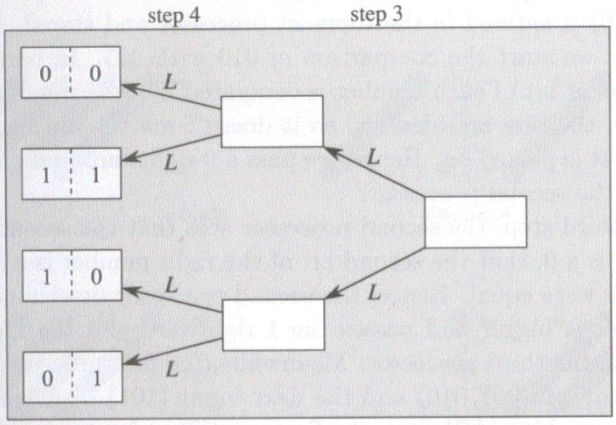


Figure 1-6 *Notification procedure used following comparison of* 0110 *and* 0101.

این روش به k-1 پردازنده بیشتر از روش آرایه برای مقایسه نیاز دارد. برای مرتب‌سازی به «آرایه خطی از درخت‌های دودویی» نیاز داریم:

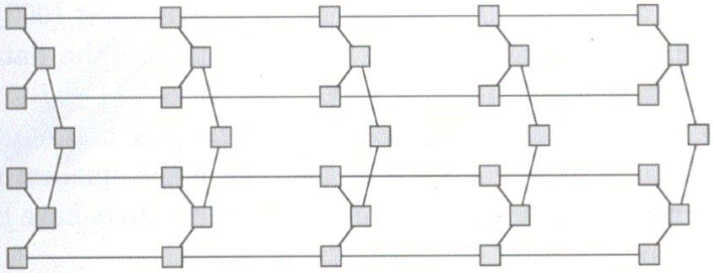


Figure 1-7 *A linear array of complete binary trees.*

زمان O(n log k) می‌شود.

ما هنوز با O(n+k) فاصله داریم. آیا می‌توانیم به آن برسیم؟

در توپولوژی زیر، با شیفت دادن عدد بزرگتر را به سمت راست ارسال می‌کند.

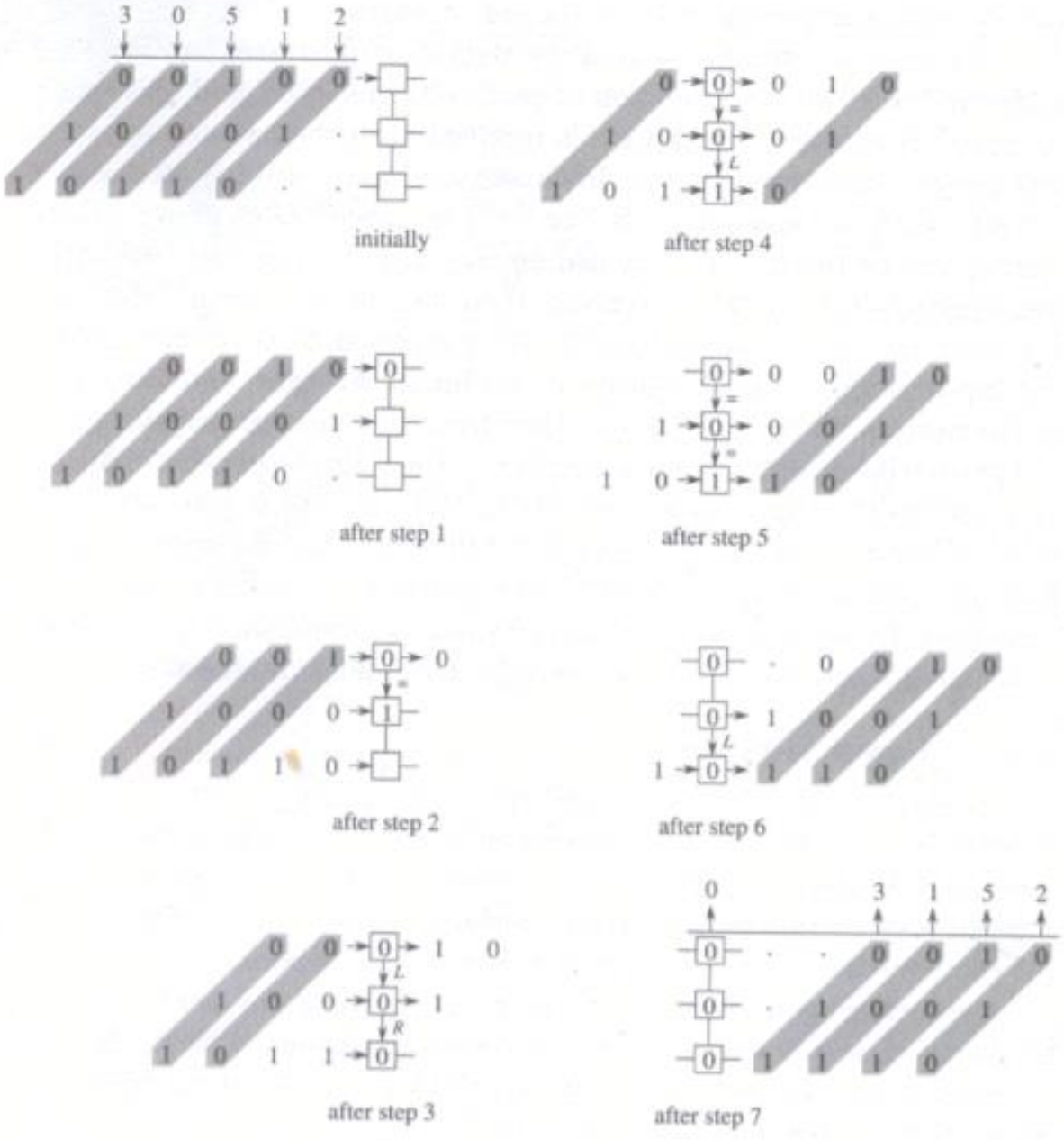


Figure 1-8 *Using a 3-cell linear array to find the smallest of five 3-bit numbers* $\{3, 0, 5, 1, 2\}$.

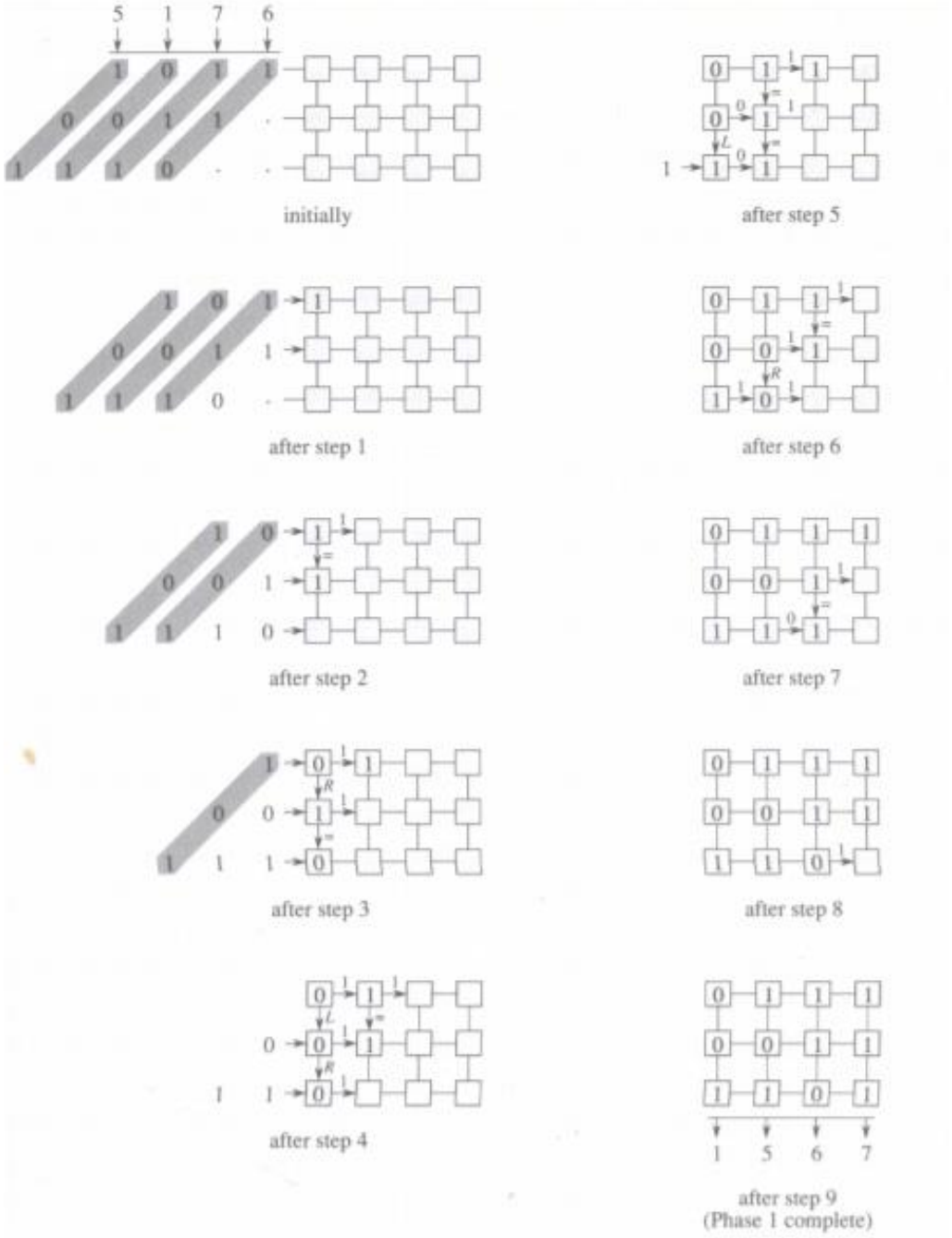


Figure 1-9 *Phase 1 sorting of* $5, 1, 7, 6$ *on a* 3×4 *array in the bit model.*

هر بار کوچکترین عدد ذخیره می‌شود و اعداد بعدی عبور می‌کنند.

زمان: 2n+k-1

(2n-1 گام برای حرکت افقی و k گام برای حرکت عمودی)

برای رسیدن به پایین‌ترین پردازنده‌ی سمت راست

کوچکترین عدد را نگه می‌دارد و بقیه را برای پردازنده بعدی می‌فرستد. وقتی عدد کوچکتری دید اعلام می‌کند بیت‌های بعدی آن نگه داشته شود.

تعداد پردازنده‌ها: nk (پردازنده‌های فقط مقایسه‌کننده دارند)

چون بیتی است کلاک آن بسیار سریع است.

قطر: n+k-1 (کران پایین)

پهنای دوبخشی: k

یعنی در هر کلاک حداکثر k بیت مبادله می‌شود.

تبادل داده لازم کل: nk

کران‌پایین: nk/k=n

هر کدام از این کران‌های پایین (کران پایین با قطر، کران پایین با پهنای دوبخشی) بیشتر بود کران پایین مسئله است.

مثال:

- پهنای دوبخشی درخت: ۱
- Hypercube: n
- گراف کامل: n-1 (درجه هر رأس)

درجه: ملاک قابلیت گسترش مدار

یعنی اگر n را به n+1 بخواهیم تبدیل کنیم، باید همه‌ی قبلی‌ها را تغییر بدهیم (گراف کامل که درجه n دارد)، اما اگر درجه ثابت باشد، این طور نیست.

اکثراً log n است.

مثال: n عدد k بیتی را با ساختار درختی جمع کنید.

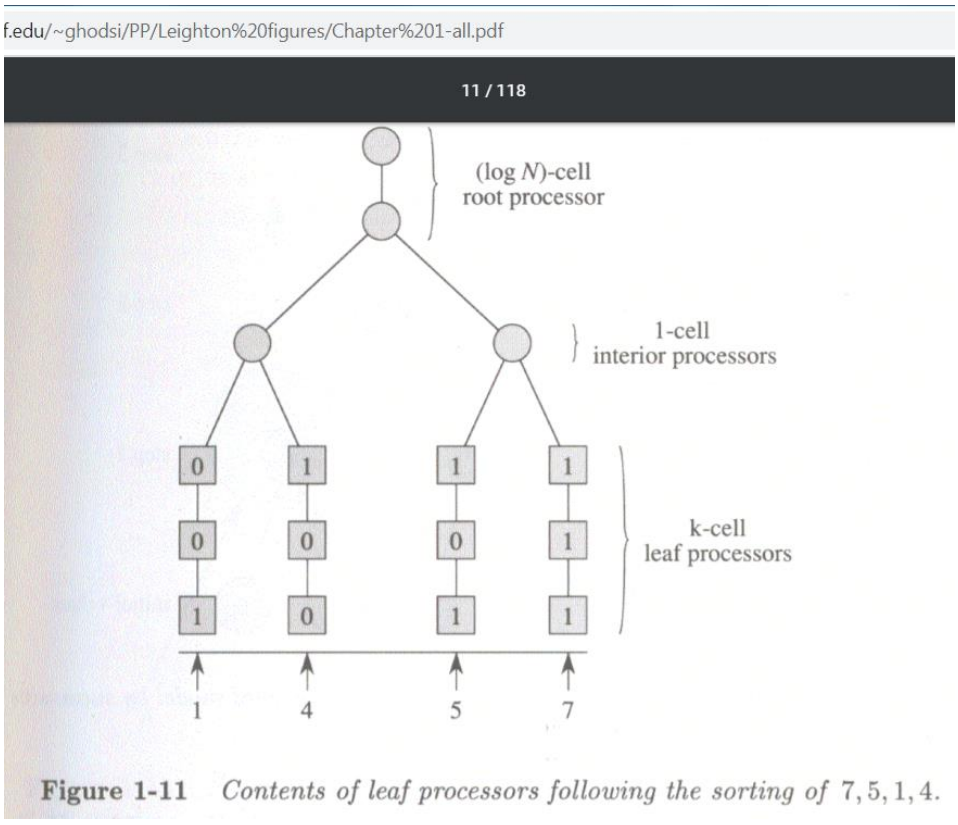


Figure 1-11 *Contents of leaf processors following the sorting of* $7, 5, 1, 4$.

حاصل جمع حداکثر k+log n بیتی است. می‌خواهیم با زمان k+log n حل کنیم.

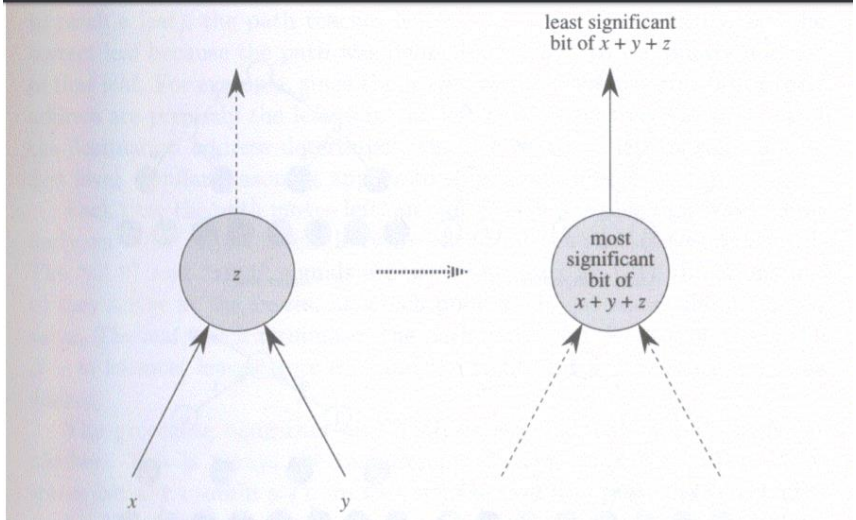


Figure 1-13 *Function of each bit processor in the algorithm for summing N 1-bit numbers.*

از بیت کم ارزش عددها را بالا می‌فرستیم. کم‌ارزش را بالا می‌فرستد، پرارزش را برای دور بعدی نگه می‌دارد.

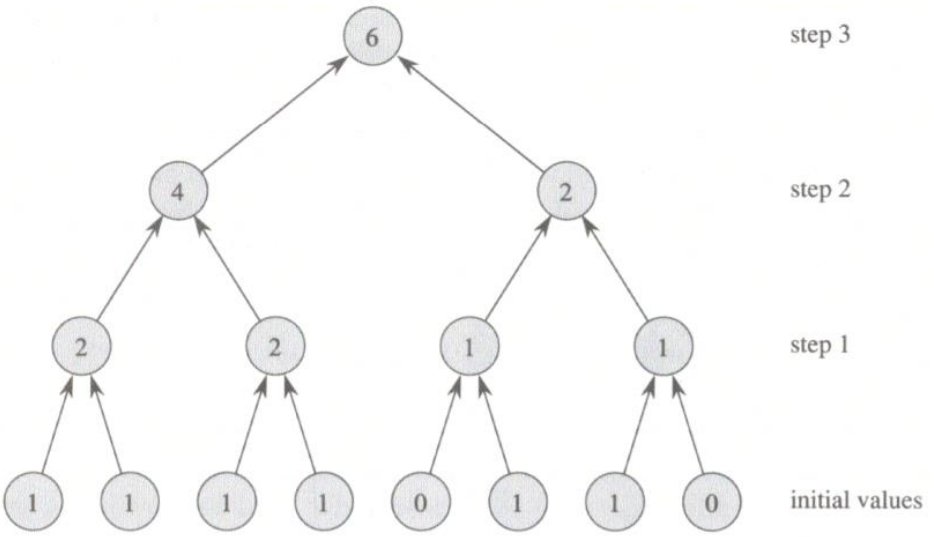


Figure 1-12 *Computing the sum of N numbers in the word model by successive pairwise sums.*

بعد از اولین مرحله سمت راست‌ترین رقم جواب و در کلاک‌های بعدی بیت‌های بعدی ساخته می‌شود.

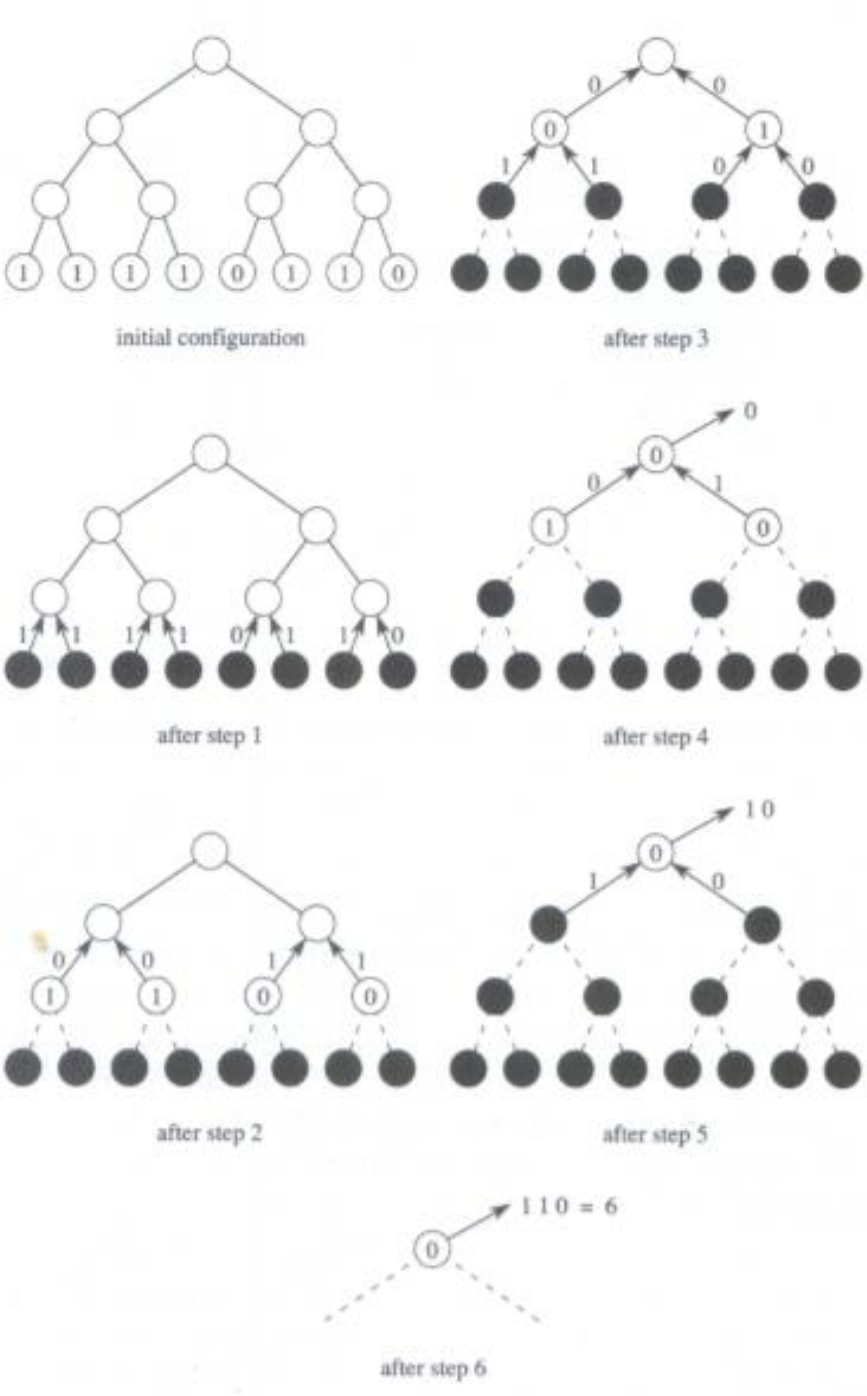


Figure 1-14 *Operation of the counting algorithm on input* $1, 1, 1, 1, 0, 1, 1, 0$. *Processors that have finished their computation are heavily shaded. For simplicity, we have omitted the* $\log N$*-cell linear array at the root.*

زمان = k+log n مرحله

پهنای دوبخشی = log n (چون بیتی است)

چطور n بیت را در برگ‌ها مرتب کنیم؟

کران پایین (کلمه‌ای): n

قطر $\theta(n)$

پهنای دوبخشی ۱

کران پایین (بیتی): log n

قطر log n

پهنای دوبخشی log n

ایده: تعداد ۱ها را بشماریم. به تعداد ۱ها باید در برگ‌های سمت چپ ۱ بگذاریم (فرض کنید m) و به تعداد n- m در برگ‌های سمت راست ۰ بگذاریم.

الگوریتم: با الگوریتم قبلی تعداد ۱ها را حساب کنیم.

در ریشه مرز ۰ و ۱ ها را حساب می‌کنیم.

به سمت راست مرز all0 و به سمت چپ آن all1 می‌فرستیم.

محاسبه مرز در ریشه: تعداد ۰ها را با مکمل گرفتن از بیت‌های نمایش دودویی m به دست می‌آوریم. جایگاه سمت راست‌ترین ۱ در برگ‌های مرتب‌شده یکی بیشتر است. چون پردازنده ریشه log n بیتی است می‌توانیم این محاسبات را در آن انجام بدهیم.

حرکت برای رسیدن به مرز: اگر ما اندیس دودویی یک برگ را داشته باشیم می‌توانیم آن را پیدا کنیم. کافی است که هر بار به ۰ رسیدیم به چپ و هر بار به ۱ رسیدیم به راست حرکت کنیم.

(من اندیس برگ‌ها را از راست به چپ گذاشته‌ام ولی شکل از چپ به راست است).

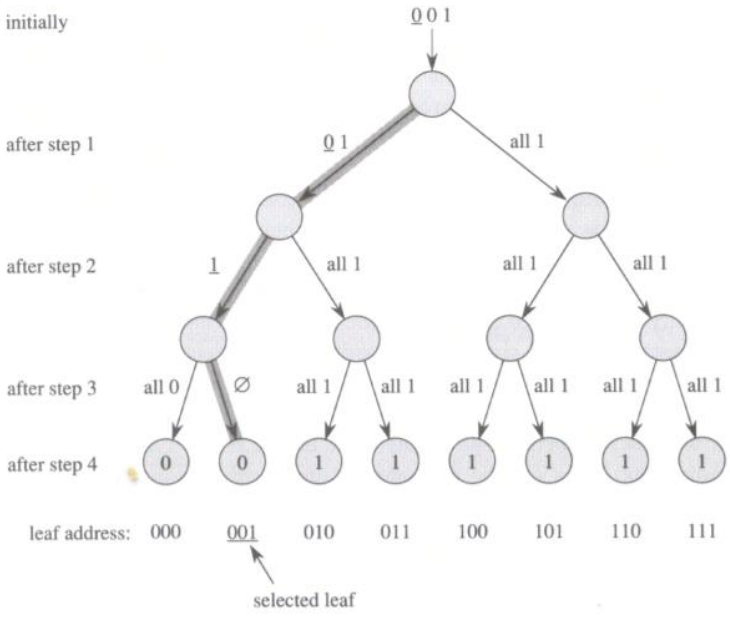


Figure 1-15 *Selection of the path (shown with shaded edges) to leaf* 001. *The underlined bit at each step governs the choice of left* (0) *or right* (1) *made by the path as it proceeds downward. Hence, the path from the root to leaf* 001 *goes left, left, right.*