ΑΝΑΖΗΤΗΣΗ

Η αναζήτηση είναι μία από τις πλέον βασικές πράξεις στην περιοχή των υπολογισμών. Χρησιμοποιείται σε οποιαδήποτε εφαρμογή απαιτείται να βρεθεί αν ένα στοιχείο ανήκει σε μία ήδη ή πιο γενικά ανάκτηση πληροφοριών από ένα αρχείο που σχετίζονται με αυτό το στοιχείο. Το πρόβλημα της αναζήτησης διατυπώνεται ως εξής:

Δίνεται μία ακολουθία $S=\{s_1, s_2, \ldots, s_n\}$ ακέραιων αριθμών και ένας ακέραιος αριθμός x. Ζητείται να βρεθεί αν $x=s_k$ για κάποιο s_k στην S.

Στους ακολουθιακούς υπολογισμούς το πρόβλημα λύνεται με τη σύγκριση της ακολουθίας S και τη σύγκριση του x με τα διαδοχικά στοιχεία της μέχρις ότου είτε ένας ακέραιος είναι ίσος με x ή τελειώσουν τα στοιχεία της ακολουθίας. Η μέθοδος αυτή είναι η σειριακή ή ακολουθιακή αναζήτηση και δίνεται από την procedure SEQUENTIAL SEARCH. Στη χειρότερη περίπτωση ο υπολογιστικός χρόνος του υπολογισμάτος είναι $O(n)$, το οποίο είναι βέλτιστο αφού

Procedure SEQUENTIAL SEARCH (S, x, k)

Βήμα 1: (1.1) $i \leftarrow 1$
(1.2) $k \leftarrow 0$.

Βήμα 2: **όσο** ($i \leq n$ **και** $k = 0$) **κάνε**
αν $s_i = x$ **τότε** $k \leftarrow i$ **τέλος αν**
$i \leftarrow i + 1$
τέλος όσο.

$O(n)$ βέλτιστος χρόνος

Κάθε στοιχείο της S πρέπει να ελεχθεί (όταν το x δε βρίσκεται στην S). Στην περίπτωση που η S είναι ταξινομημένη σε μη φθίνουσα διάταξη, τότε η BINARY SEARCH επιστρέφει τη θέση ενός στοιχείου της S που είναι ίσο με το x (ή 0 αν δεν υπάρχει τέτοιο στοιχείο σε $O(\log n)$ χρόνο.

Από την άλλη πλευρά ο CREW αλγόριθμος βρίσκει ένα κάτω όριο του αριθμού των παράλληλων βημάτων που απαιτούνται για την αναζήτηση ταξινομημένων ακολουθιών, υποθέτοντας ότι όλα τα στοιχεία είναι διακριτά. Αν αυτή η υπόθεση δεν ισχύει, τότε χρειάζεται ένας CRCW αλγόριθμος για να επιτευχθεί η καλύτερη δυνατή επιτάχυνση.

Αναζήτηση σε ταξινομημένη ακολουθία

Υποθέτουμε ότι η ακολουθία $S = \{s_1, s_2, \ldots, s_n\}$ είναι ταξινομημένη σε μη φθίνουσα σειρά, δηλ. $s_1 \leq s_2 \leq \ldots \leq s_n$. Συνήθως, ένα αρχείο με n εγγραφές, οι οποίες είναι ταξινομημένες με βάση το κλειδί s (πεδίο) είναι αυτό το πρόβλημα που συναντάται στις εφαρμογές. Χάριν

δηλαδή, δοθέντος ενός ακεραίου x, αναζητάμε την εγγραφή της οποίας το πεδίο s ισούται με x. Αν βρεθεί τέτοια εγγραφή τότε αυτή αποθηκεύεται σε όλα τα άλλα πεδία και μπορεί να βρεθεί.

απλότητας υποθέτουμε ότι τα s_i είναι διαφορετικά.

EREW Αναζήτηση

Υποθέτουμε ότι έχουμε N επεξεργαστές σε ένα EREW SM SIMD υπολογιστή για την αναζήτηση ενός δεδομένου στοιχείου x.

Καταρχήν η τιμή του x θα πρέπει να γίνει γνωστή σε όλους τους επεξεργαστές. Αυτό μπορεί να γίνει με την BROADCAST σε $O(\log N)$ χρόνο. Στη συνέχεια η S χωρίζεται σε N υποακολουθίες με n/N στοιχεία η κάθε μία και στον P_i επεξεργαστή καταχωρούνται τα στοιχεία $\{s_{(i-1)(n/N)+1}, s_{(i-1)(n/N)+2}, \ldots, s_{i(n/N)}\}$. Όλοι οι επεξεργαστές εκτελούν την BINARY SEARCH στις υποακολουθίες τους, που βρίσκονται $O(\log(n/N))$ χρόνο στην χειρότερη περίπτωση. Επειδή τα στοιχεία της S είναι όλα διαφορετικά, το πολύ ένας επεξεργαστής θα εντοπίσει ένα s_k ίσο με το x και επιστρέφει το k. Ο συνολικός χρόνος που απαιτείται από αυτόν τον EREW αλγόριθμο αναζήτησης είναι $O(\log N) + O(\log(n/N))$, το οποίο είναι $O(\log n)$. Επειδή ο χρόνος αυτός είναι ίδιος με αυτόν της ακολουθιακής BINARY SEARCH δεν επιτυγχάνεται αύξηση της ταχύτητας με τη μέθοδο αυτή!

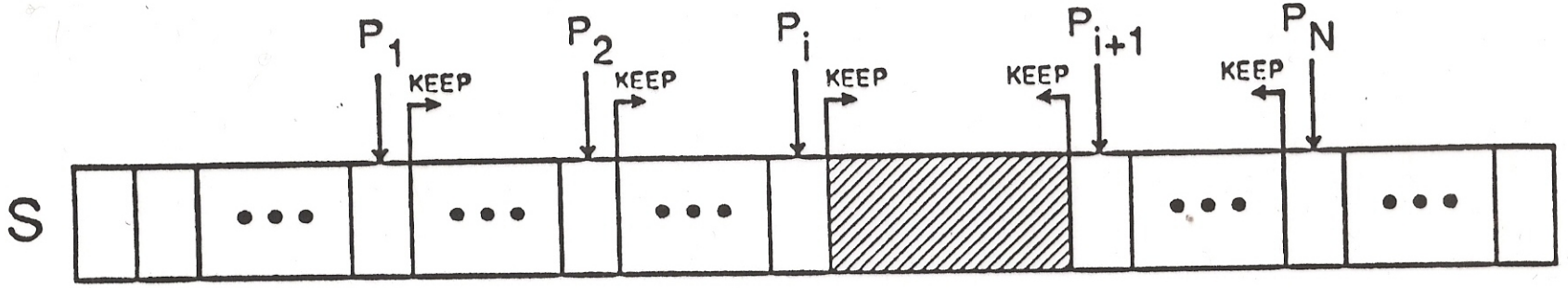
CREW Αναζήτηση

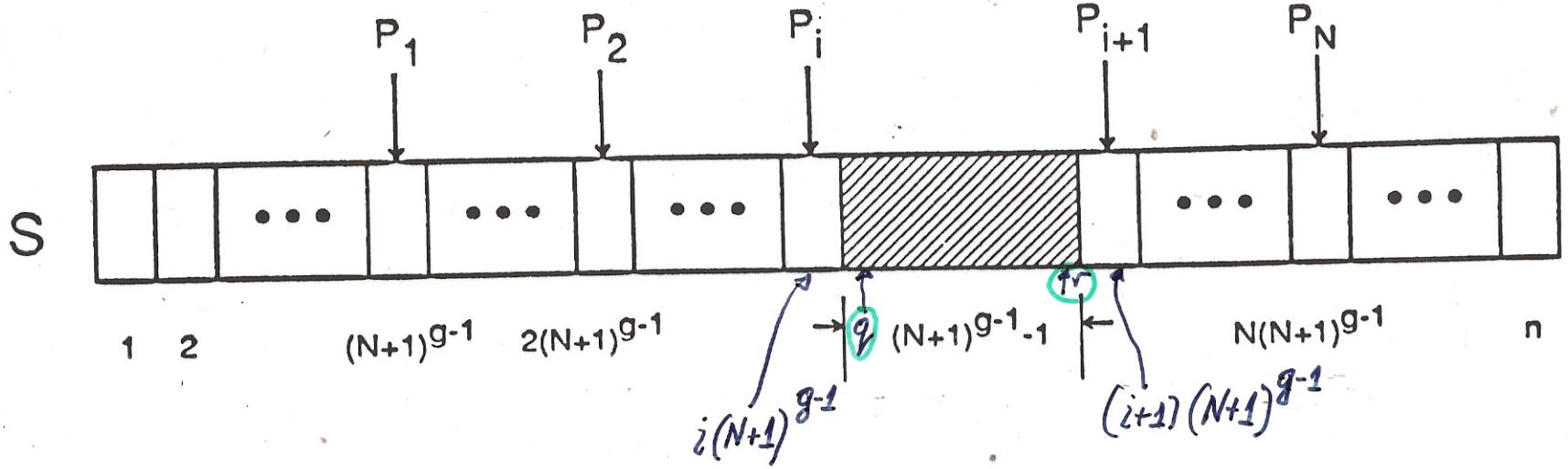
Ο αλγόριθμος για τον CREW υπολογιστή μπορεί να χρησιμοποιηθεί και στην παρούσα περίπτωση με τη διαφορά ότι τώρα όχι οι επεξεργαστές μπορούν να διαβάσουν το x ταυτόχρονα σε σταθερό χρόνο και στη συνέχεια προχωρούν στην εκτέλεση της BINARY SEARCH για τις υποακολουθίες τους. Ο απαιτούμενος χρόνος τώρα είναι $O(\log(n/N))$ στη χειρότερη περίπτωση και είναι μικρότερος από τον ακολουθιακό χρόνο της BINARY SEARCH. Είναι δυνατόν όμως να επιτύχουμε καλύτερο χρόνο με την παραλληλοποίηση της δυαδικής αναζήτησης. Στην παράλληλη αναζήτηση υπάρχουν διαθέσιμοι N επεξεργαστές και συνεπώς χρησιμοποιείται N+1-αδική αναζήτηση. Σε κάθε φάση, η ακολουθία χωρίζεται σε N+1 υποακολουθίες με ίσο μήκος στοιχείων και οι N επεξεργαστές εξέχουν ταυτόχρονα τα στοιχεία που βρίσκονται στο σύνορο μεταξύ διαδοχικών υποακολουθιών (βλ. Σχήμα). Κάθε επεξεργαστής συγκρίνει το στοιχείο s του S με το x. και:

1. Αν $s > x$, τότε απορρίπτονται όλα τα στοιχεία μεγαλύτερα του s.

2. Διαφορετικά, αν $s \leq x$, τότε απορρίπτονται όλα τα στοιχεία μικρότερα του s.

Συνεπώς κάθε επεξεργαστής κυρίως την ακολουθία σε δύο τμήματα. Το ένα τμήμα περιέχει όλα τα στοιχεία τα οποία δημοσιεύονται καθώς σίγουρα δεν περιέχουν ένα στοιχείο ίδιο με x και το άλλο τμήμα που περιέχει να εκείνα τα στοιχεία τα οποία πιθανά να περιέχουν ένα στοιχείο ίδιο με x. Το πρώτο τμήμα δημοσιεύεται ενώ το δεύτερο διατηρείται. Η διαδικασία αυτή έχει δύο αποτέλεσμα να συρρικνώνεται η αρχική ακολουθία σε μια υποακολουθία που είναι η ένωση των τμημάτων που διατηρούνται, δηλαδή η υποακολουθία που βρίσκεται μεταξύ δύο στοιχείων που εξετάζονται σε αυτή τη φάση. Αυτή η υποακολουθία (φαίνεται χαρακτηριστικά στο Σχήμα) είναι εκείνη στην οποία εφαρμόζεται η ίδια διαδικασία στην επόμενη φάση. Η ανταλλαγή συνεχίζεται μέχρις ότου είτε ένα


P1
P2
Pi
Pi+1
PN
KEEP
KEEP
KEEP
KEEP
KEEP
S


P1
P2
Pi
Pi+1
PN
S
1
2
(N+1)g-1
2(N+1)g-1
q
(N+1)g-1-1
r
N(N+1)g-1
n
i(N+1)g-1
(i+1)(N+1)g-1

στοιχείο βρεθεί ίδιο με x ή αποριφθούν όλα τα στοιχεία της S. Επειδή κάθε φάση χρησιμοποιεί μία ακολουθία της οποίας το μήκος είναι $\frac{\text{μήκος προηγούμενης}}{N+1} - 1$, δηλαδή τώρα $O(\log_{N+1}(n+1))$ φάσεις. Στη συνέχεια αναπτύσσουμε πιο αναλυτικά τη μέθοδο.

Έστω g ο μικρότερος ακέραιος τέτοιος ώστε $n \le (N+1)^g - 1$ δηλαδή $g = \lceil \log(n+1)/\log(N+1) \rceil$. Στη συνέχεια θα δείξουμε με επαγωγή ότι g φάσεις είναι ικανές για την αναζήτηση μιας ακολουθίας μήκους n. Προφανώς, η συνέχεια πρότασή είναι σωστή για $g=0$. Υποθέτουμε ότι η πρόταση είναι σωστή για την αναζήτηση μιας ακολουθίας μήκους $(N+1)^{g-1} - 1$. Για την αναζήτηση μιας ακολουθίας μήκους $(N+1)^g - 1$ ο επεξεργαστής P_i, $i = 1, 2, \ldots, N$ συγκρίνει το x με το S_j, όπου $j = i(N+1)^{g-1}$ (βλ. Σχήμα). Μετά τη σύγκριση, η ακολουθία για αναζήτηση έχει μήκος $(N+1)^{g-1} - 1$ και είναι η χρησιμοποιούμενη, πράγμα που αποδεικνύει την πρόταση. Η υποακολουθία αυτή προσδιορίζεται ως εξής. Κάθε επεξεργαστής P_i χρησιμοποιεί μία μεταβλητή c_i, η οποία παίρνει την τιμή left ή right ανάλογα με το

τμήμα της ακολουθίας που απομένει ο P_i να διατρήσει βρίσκεται στα αριστερά ή δεξιά του στοιχείου που συμπίπτει με το x και τη διάσταση αυτής της φάσης. Αρχικά, η τιμή κάθε c_i είναι απροσδιόριστη, με εξαιρέσεις τις τιμές των $c_0 = right$ και $c_{N+1} = left$. Μετά τη σύγκριση μεταξύ του x και ενός στοιχείου s_{j_i} της S, ο P_i καταχωρεί μία τιμή στο $c_i^{j_i}$ (εκτός εάν $s_{j_i} = x$). Αν $c_i \neq c_{i-1}$ για κάποιο i, $1 \leq i \leq N$, τότε η ακολουθία που μπόρεσαν να δημιουργηθεί θα αρχίζει από το s_q και θα τελειώνει στο s_r, όπου

$$q = (i-1)(N+1)^{q-1} - 1 \quad \text{και} \quad r = i(N+1)^{q-1} - 1$$

Ένας μόνον επεξεργαστής εντοπίζει τα q και r στην κοινή μνήμη και όσοι οι υπόλοιποι επεξεργαστές μπορούν ταυτόχρονα να διαβάσουν τις εντοπισμένες τιμές σε σταθερό χρόνο.

Procedure CREW SEARCH (S, x, k)

Βήμα 1: {Αρχικές τιμές στους δείκτες των ακολουθιών που θα σαρωθούν}

(1.1) $q \leftarrow 1$
(1.2) $r \leftarrow n$.

(1.3) c_0=right
(1.4) c_{N+1}=left

Βήμα 2: {Αρχικές τιμές για τα αποτελέσματα και το μέγιστο αριθμό σταδίων}

(2.1) $\kappa \leftarrow 0$
(2.2) $g \leftarrow \lceil \log(n+1)/\log(N+1) \rceil$.

Βήμα 3: **όσο** ($q \le r$ **και** $k = 0$) **κάνε**
(3.1) $j_0 \leftarrow q-1$
(3.2) **για** $i = 1$ **μέχρι** N **κάνε παράλληλα**
(i) $j_i \leftarrow (q-1)+i(N+1)^{g-1}$
{Ο επεξεργαστής P_i συγκρίνει το x με το s_j και προσδιορίζει το μέρος της ακολουθίας που θα διατηρηθεί}
(ii) **αν** $j_i \le r$
τότε αν $s_{j_i} = x$
τότε $k \leftarrow j_i$
αλλιώς αν $s_{j_i} > x$
τότε $c_i \leftarrow$ left
αλλιώς $c_i \leftarrow$ right
τέλος αν
τέλος αν
αλλιώς (a) $j_i \leftarrow r+1$ σημαίνει ότι j ξεφεύγει εκτός της ακολουθίας και δίνεται διορθωτική ενέργεια.
(b) $c_i \leftarrow$ left
τέλος αν
{υπολογίζονται οι δείκτες της ακολουθίας που θα σαρωθεί στην επόμενη επανάληψη}
(iii) **αν** $c_i \ne c_{i-1}$ **τότε** (a) $q \leftarrow j_{i-1} + 1$
(b) $r \leftarrow j_i - 1$
τέλος αν
(iv) **αν** ($i = N$ **και** $c_i \ne c_{i+1}$) **τότε** $q \leftarrow j_i + 1$
τέλος αν
τέλος για
(3.3) $g \leftarrow g - 1$.
τέλος όσο.

Ανάλυση

Τα βήματα 1, 2, 3.1 και 3.3 εκτελούνται από ένα επεξεργαστή, έστω, τον P_1 σε σταθερό χρόνο. Το βήμα 3.2 εκτελείται επίσης σε σταθερό χρόνο. Όπως αποδείχθηκε το βήμα 3 εκτελείται το πολύ σε g επαναλήψεις. Συνεπώς η CREW SEARCH εκτελείται σε

$O(\log(n+1)/\log(N+1))$ χρόνο, δηλαδή

$t(n) = O(\log_{N+1}(n+1))$ συνεπώς $c(n) =$

$= O(N \log_{N+1}(n+1))$ το οποίο δεν είναι βέλτιστο.

Παράδειγμα

Έστω $S = \{1, 4, 6, 9, 10, 11, 13, 14, 15, 18, 20, 23, 32, 45, 51\}$

η ακολουθία, η οποία πρόκειται να αναζητηθεί χρησιμοποιώντας ένα CREW SM SIMD υπολογιστή.

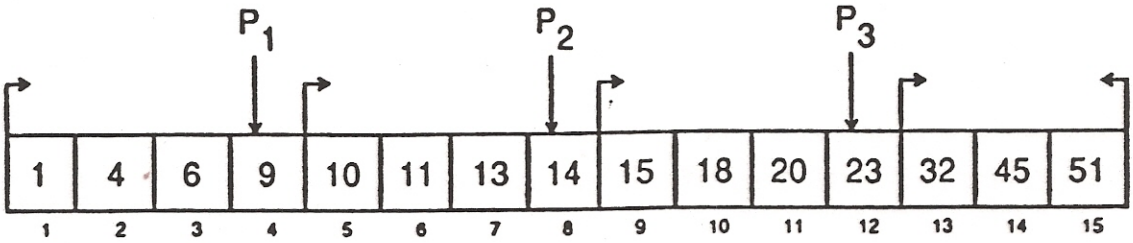
1. Υποθέτουμε ότι $N=3$, $x=45$. Αρχικά $g=1$, $r=15$, $k=0$ και $g=2$. Στην πρώτη επανάληψη του βήματος 3, ο P_1 υπολογίζει $j_1 = 4$ και συγκρίνει το s_4 με το x. Επειδή $9 < 45$, $c_1 = \text{right}$. Ταυτόχρονα οι P_2 και P_3 συγκρίνουν τα s_8 και s_{12} με το x, αντίστοιχα.

Επειδή $14 < 45$ και $23 < 45$, $c_2 = \text{right}$ και $c_3 = \text{right}$. Αλλά $c_3 \neq c_4$, συνεπώς $q = 13$ και το r παραμένει αμετάβλητο. Η νέα ακολουθία που θα αναζητηθεί ξεκινά από το s_{13} μέχρι το s_{15} (βλ. Σχήμα (β)) με $g = 1$. Στη δεύτερη επανάληψη, (Σχήμα (b)) ο P_1 υποψήφιος $j_1 = 12 + 1$ και συγκρίνει το s_{13} με το x. Επειδή $32 < 45$, $c_1 = \text{right}$. Ταυτόχρονα, ο P_2 συγκρίνει το s_{14} με το x και επειδή είναι ίσα, θέτει $k = 14$. Επίσης ο P_3 συγκρίνει το s_{15} με το x. Επειδή $51 > 45$, $c_3 = \text{left}$. Τώρα $c_3 \neq c_2$. Συνεπώς, $q = 12 + 2 + 1 = 15$ και $r = 12 + 3 - 1 = 14$ και η διαδικασία σταματάει με $k = 14$.

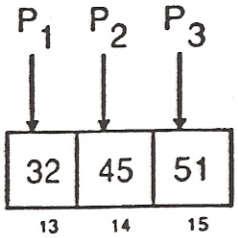
2. Έστω ότι $x = 9$ με $N = 3$. Στην πρώτη επανάληψη, ο P_1 συγκρίνει το s_4 με το x και επειδή είναι ίσα, θέτει $k = 4$.

3. Έστω ότι $N = 2$ και $x = 21$. Αρχικά $g = 3$. Στην πρώτη επανάληψη ο P_1 υποψήφιος $j_1 = 9$ και συγκρίνει το s_9 με το x, επειδή $15 < 21$, $c_1 = \text{right}$. Ταυτόχρονα ο P_2 υποψήφιος $j_2 = 16$ και $c_2 = \text{left}$. Τώρα $c_2 \neq c_1$, συνεπώς $q = 10$ και $r = 15$ έτσι η ακολουθία που θα αναζητηθεί θα είναι από s_{10} μέχρι s_{15} και $g = 2$. (βλ. Σχήμα (c)).

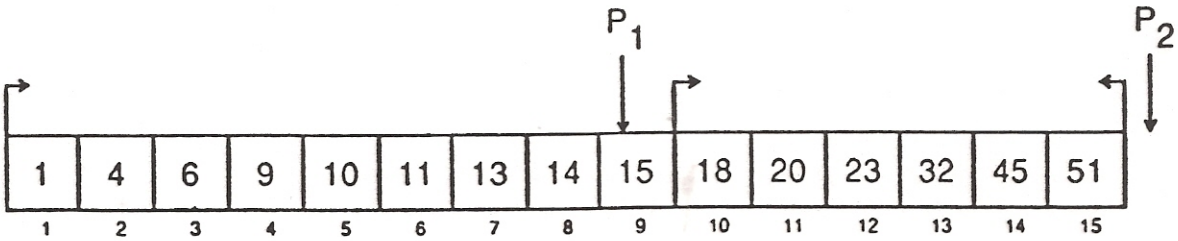
Στη δεύτερη επανάληψη ο P_1 υποδηλώνει $j_1 = q+3$ και συγκρίνει το S_{12} με το x, επειδή $51 > 11$, $c_2 = left$. Τώρα $c_1 \neq c_0$ και συνεπώς $r = 11$ και το q παραμένει αμετάβλητο (βλ. Σχήμα (d)). Στην τέταρτη επανάληψη, $q = 1$ και ο P_1 υποδηλώνει $j_1 = q+1$, συγκρίνει το S_{10} με το x. Επειδή $19 < 11$, $c_1 = right$. Ταυτόχρονα, ο P_2 υποδηλώνει $j_2 = q+2$ και συγκρίνει το S_{11} με το x, επειδή $20 < 21$, $c_2 = right$. Τώρα $c_2 \neq c_3$ και $q = 12$. Επειδή $q > r$ η διαδικασία διαμοιράζει αντιστοιχώς και επιστρέφει $k = 0$.


P1
P2
P3
1 4 6 9 10 11 13 14 15 18 20 23 32 45 51
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

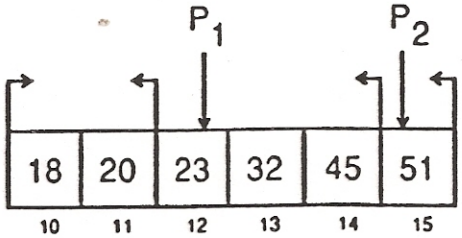
(a)


P1
P2
P3
32 45 51
13 14 15

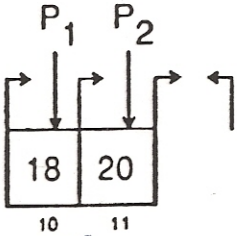
(b)


P1
P2
1 4 6 9 10 11 13 14 15 18 20 23 32 45 51
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

(c)


P1
P2
18 20 23 32 45 51
10 11 12 13 14 15

(d)


P1
P2
18 20
10 11

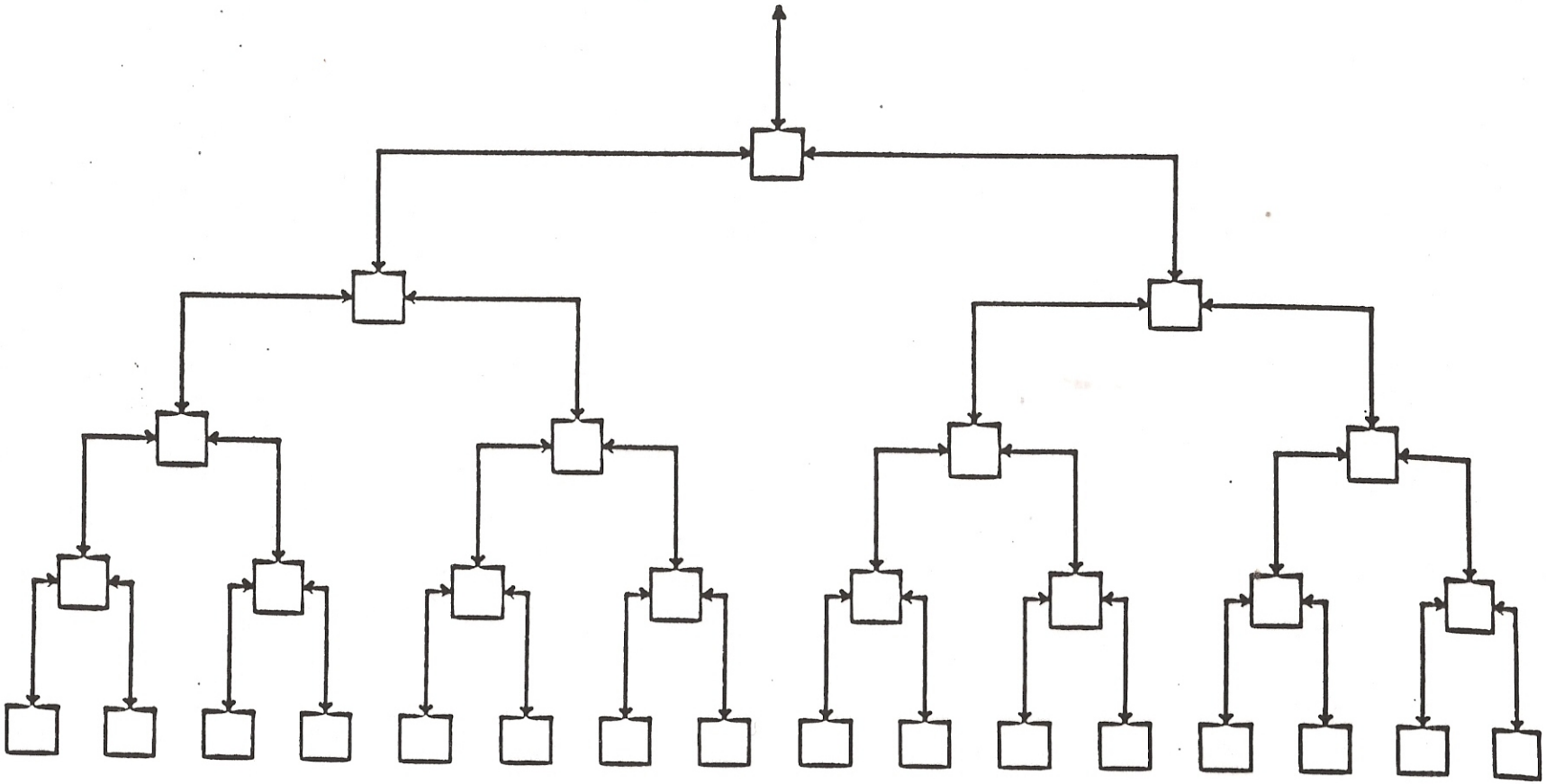
(e)

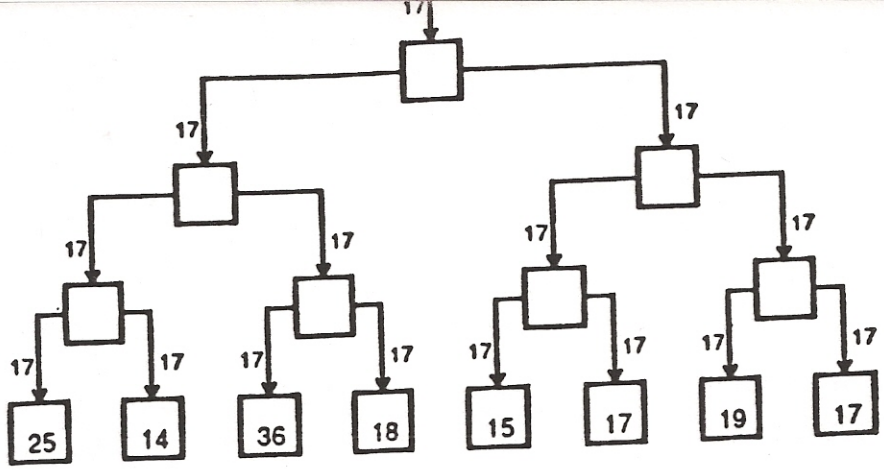
Procedure SM SEARCH (S, x, k)

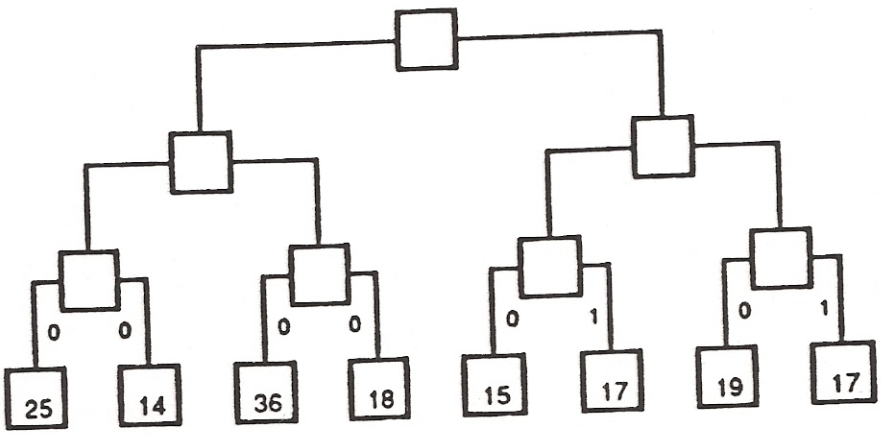
Βήμα 1: **για** i = 1 **μέχρι** N **κάνε παράλληλα**
διάβασε το x
τέλος για.

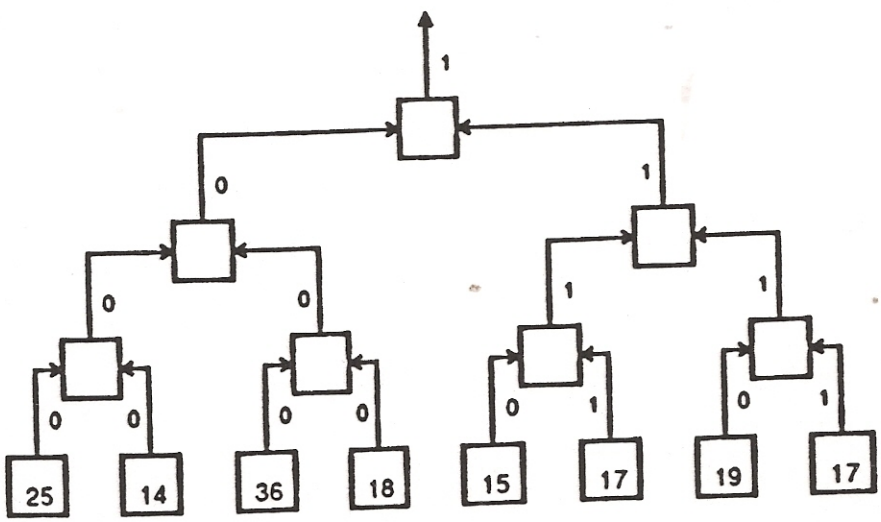
Βήμα 2: **για** i = 1 **μέχρι** N **κάνε παράλληλα**
(2.1) $S_i \leftarrow \{s_{(i-1)(n/N)+1}, s_{(i-1)(n/N)+2}, \ldots, s_{i(n/N)}\}$
(2.2) SEQUENTIAL SEARCH (S_i, x, k_i)
τέλος για.

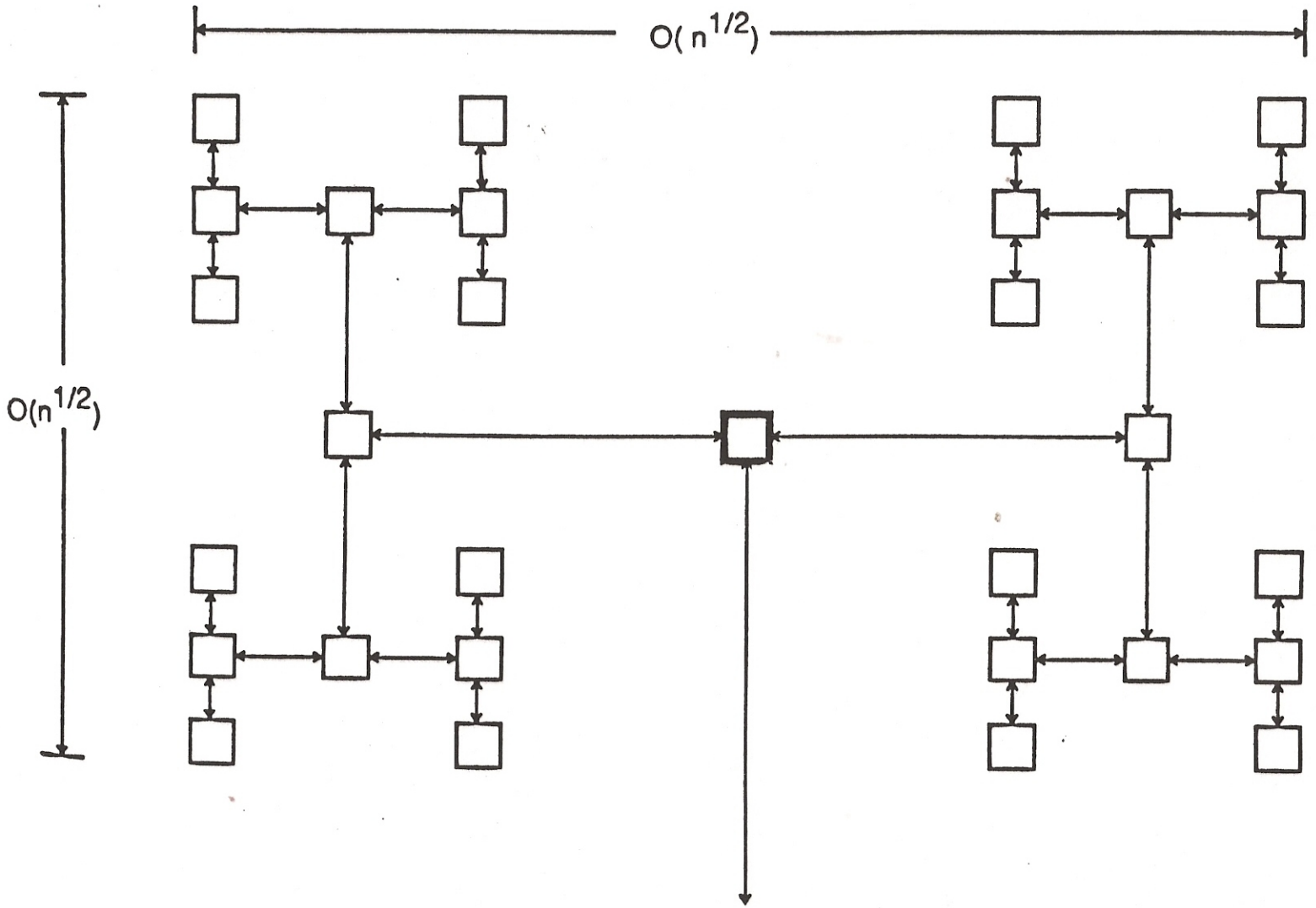
Βήμα 3: **για** i = 1 **μέχρι** N **κάνε παράλληλα**
αν $k_i > 0$ **τότε** $k \leftarrow k_i$ **τέλος αν**
τέλος για.

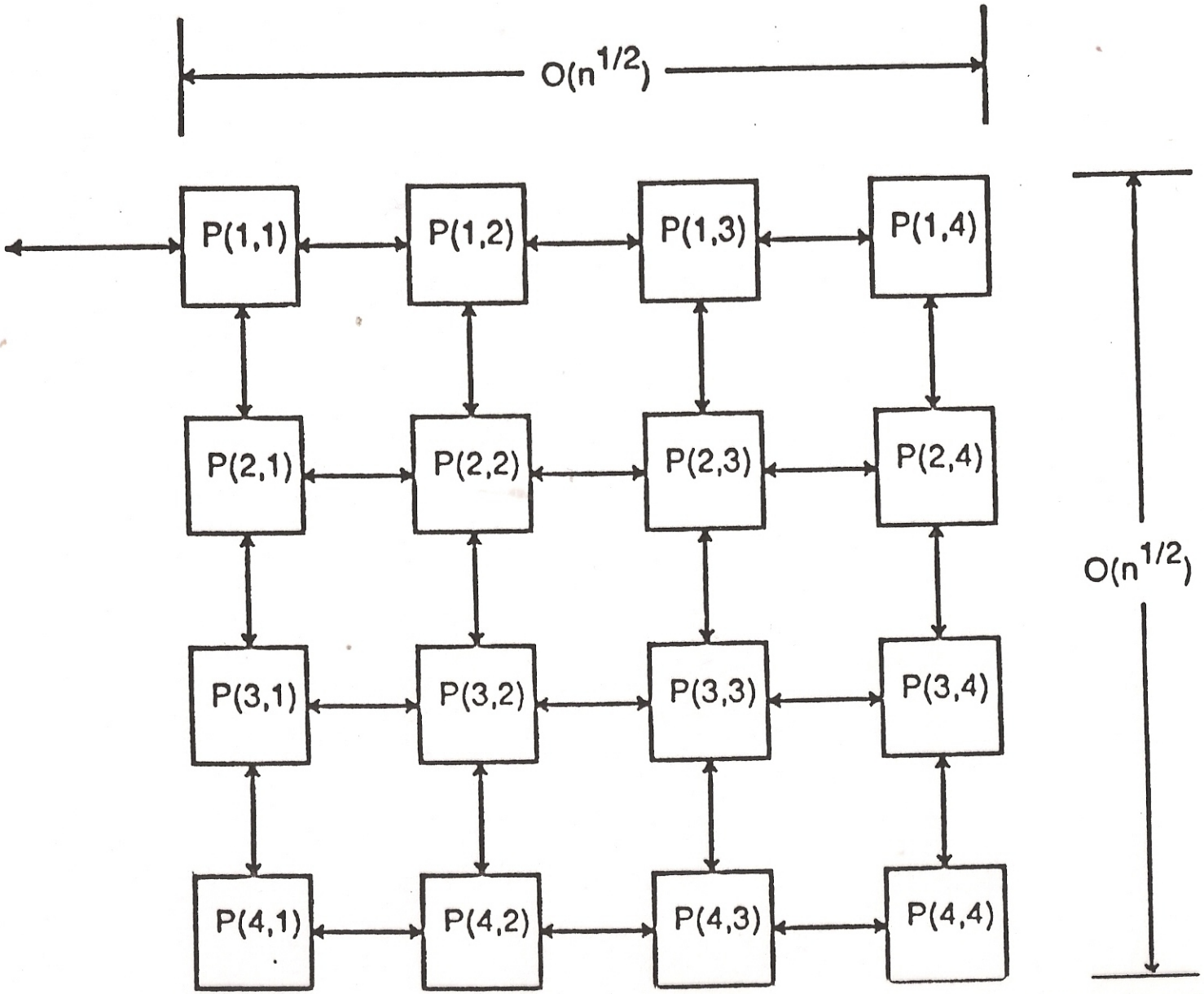



17
17
17
17
17
17
17
17
17
17
17
17
17
17
17
17
25
14
36
18
15
17
19
17


0
0
0
0
0
1
0
1
25
14
36
18
15
17
19
17


1
0
1
0
0
1
1
0
0
0
0
1
0
1
25
14
36
18
15
17
19
17


O(n^1/2)
O(n^1/2)


O(n^1/2)
P(1,1)
P(1,2)
P(1,3)
P(1,4)
P(2,1)
P(2,2)
P(2,3)
P(2,4)
P(3,1)
P(3,2)
P(3,3)
P(3,4)
P(4,1)
P(4,2)
P(4,3)
P(4,4)
O(n^1/2)

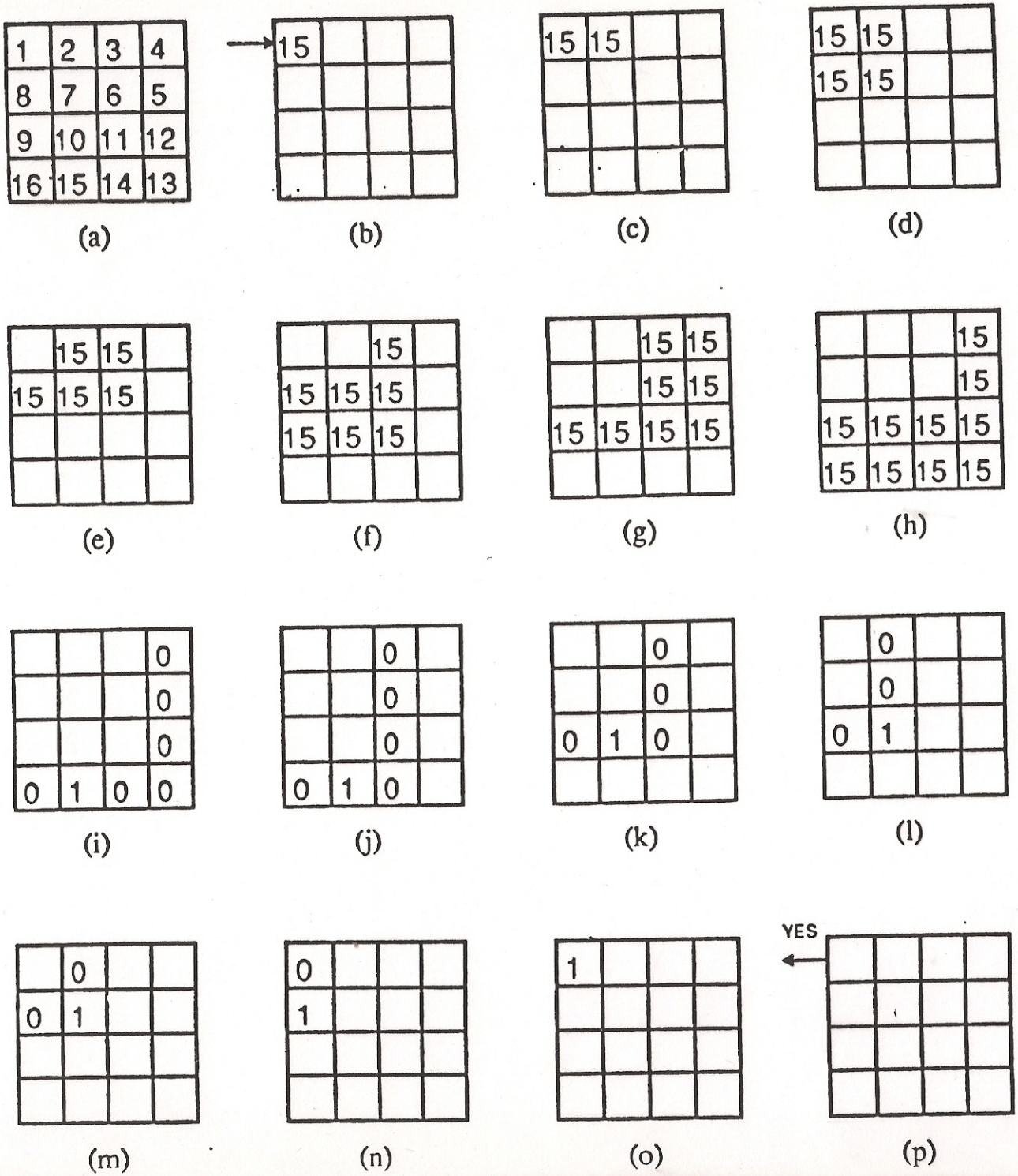
Procedure MESH SEARCH (S, x, απάντηση)

Βήμα 1: {Ο P(1,1) διαβάζει τα δεδομένα}
αν $x = s_{1,1}$ **τότε** $b_{1,1} \leftarrow 1$
αλλιώς $b_{1,1} \leftarrow 0$
τέλος αν.

Βήμα 2: {ξεδίπλωμα}
για i = 1 **μέχρι** $n^{1/2} - 1$ **κάνε**
(2.1) **για** j = 1 **μέχρι** i **κάνε παράλληλα**
(i) Ο P(j,i) μεταδίδει το $(b_{j,i}, x)$ στο P(j,i+1)
(ii) **αν** $(x = s_{j,i+1}$ **ή** $b_{j,i} = 1)$ **τότε** $b_{j,i+1} \leftarrow 1$
αλλιώς $b_{j,i+1} \leftarrow 0$
τέλος αν
τέλος για
(2.2) **για** j = 1 **μέχρι** i+1 **κάνε παράλληλα**
(i) Ο P(i,j) μεταδίδει το $(b_{i,j}, x)$ στο P(i+1, j)
(ii) **αν** $(x = s_{i+1,j}$ **ή** $b_{i,j} = 1)$ **τότε** $b_{i+1,j} \leftarrow 1$
αλλιώς $b_{i+1,j} \leftarrow 0$
τέλος αν
τέλος για
τέλος για.

Βήμα 3: {δίπλωμα}
για $i = n^{1/2}$ **μέχρι** 2 **κάνε**
(3.1) **για** j = 1 **μέχρι** i **κάνε παράλληλα**
Ο P(j,i) μεταδίδει το $b_{j,i}$ στο P(j,i-1)
τέλος για
(3.2) **για** j = 1 **μέχρι** i - 1 **κάνε παράλληλα**
$b_{j,i-1} \leftarrow b_{j,i}$
τέλος για
(3.3) **αν** ($b_{i,i-1} = 1$ **ή** $b_{i,i} = 1$) **τότε** $b_{i,i-1} \leftarrow 1$
αλλιώς $b_{i,i-1} \leftarrow 0$
τέλος αν
(3.4) **για** j = 1 **μέχρι** i - 1 **κάνε παράλληλα**
Ο P(i,j) μεταδίδει το $b_{i,j}$ στο P(i-1, j)
τέλος για
(3.5) **για** j = 1 **μέχρι** i - 2 **κάνε παράλληλα**
$b_{i-1,j} \leftarrow b_{i,j}$
τέλος για
(3.6) **αν** ($b_{i-1,i-1} = 1$ **ή** $b_{i,i-1} = 1$) **τότε** $b_{i-1,i-1} \leftarrow 1$
αλλιώς $b_{i-1,i-1} \leftarrow 0$
τέλος αν
τέλος για.

Βήμα 4: {Ο P(1,1) δημιουργεί τα αποτελέσματα}
αν $b_{1,1} = 1$ **τότε** απάντηση ← ναι
αλλιώς απάντηση ← όχι
τέλος αν


1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13
(a)
15
(b)
15 15
(c)
15 15
15 15
(d)
15 15
15 15 15
(e)
15
15 15 15
15 15 15
(f)
15 15
15 15
15 15 15 15
(g)
15
15
15 15 15 15
15 15 15 15
(h)
0
0
0
0 1 0 0
(i)
0
0
0
0 1 0
(j)
0
0
0 1 0
(k)
0
0
0 1
(l)
0
0 1
(m)
0
1
(n)
1
(o)
YES
(p)