Δ.Π.Θ Δομημένος Προγραμματισμός

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

Γενικές Ασκήσεις ΙΙ

Dr. Αθανάσιος Μπαλαφούτης
Εργαστηριακό Διδακτικό Προσωπικό
Τομέας Συστημάτων Παραγωγής
Εργαστήριο Ρομποτικής και Αυτοματισμών
abalafou@pme.duth.gr
Γραφείο 304, τηλ.: 25410 – 79892

Μοναδικοί τυχαίοι αριθμοί

Θέλω να γεμίσω έναν πίνακα 5 θέσεων με τυχαίους ακέραιους αριθμούς στο διάστημα 0-9, εξασφαλίζοντας ότι κάθε αριθμός θα εμφανίζεται μόνο μια φορά μέσα στον πίνακα.

π.χ.

A[5]	7	9	0	2	4

Μοναδικοί τυχαίοι αριθμοί

Θέλω να γεμίσω έναν πίνακα 5 θέσεων με τυχαίους ακέραιους αριθμούς στο διάστημα 0-9, εξασφαλίζοντας ότι κάθε αριθμός θα εμφανίζεται μόνο μια φορά μέσα στον πίνακα.

π.χ.

A[5]	7	9	0	2	4

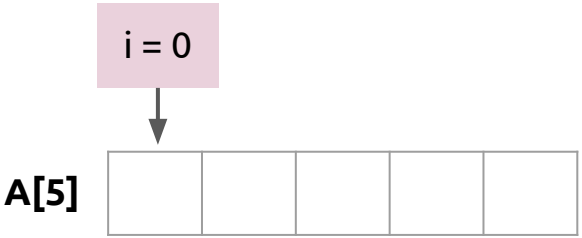
Ιδέα: Θα δημιουργήσω έναν πίνακα συχνοτήτων με τόσες θέσεις όσες και το διάστημα των τυχαίων αριθμών.

used[10]	0	0	0	0	0	0	0	0	0	0

Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: **5**

Αν **used[5] = 0**, τότε βάλε το 5 στον πίνακα Α (**A[0] = 5**) και άλλαξε το **used[5] = 1**

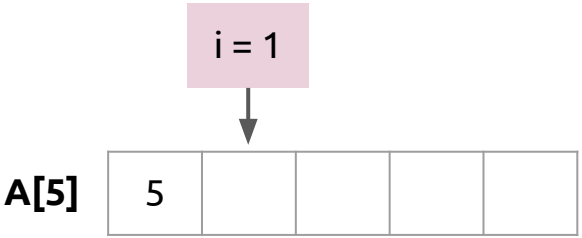


Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: **5**

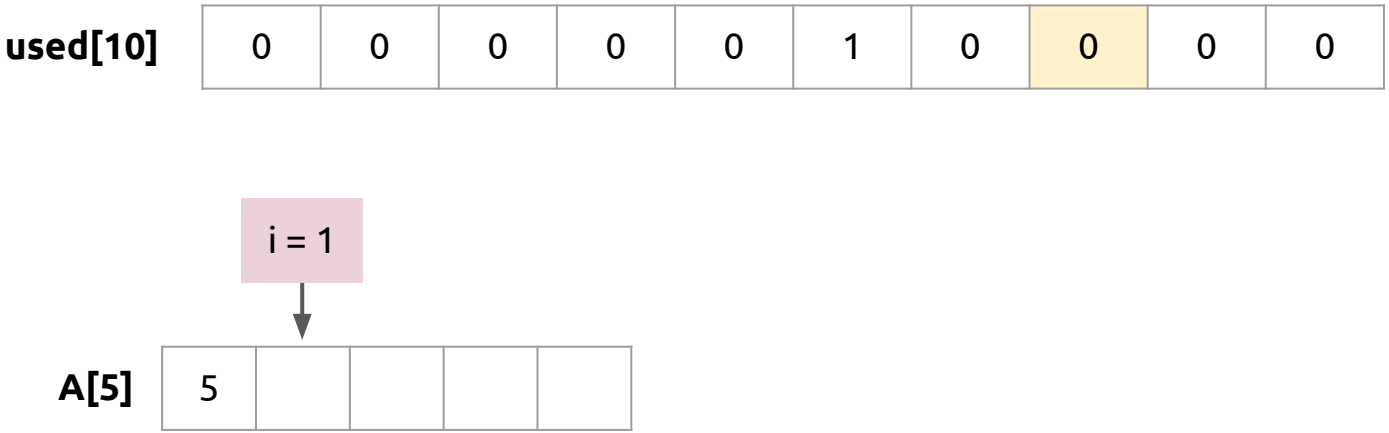
Αν **used[5] = 0**, τότε βάλε το 5 στον πίνακα Α (**A[0] = 5**) και άλλαξε το **used[5] = 1**

used[10]	0	0	0	0	0	1	0	0	0	0



Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: **7**



Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: **7**

used[10]	0	0	0	0	0	1	0	1	0	0

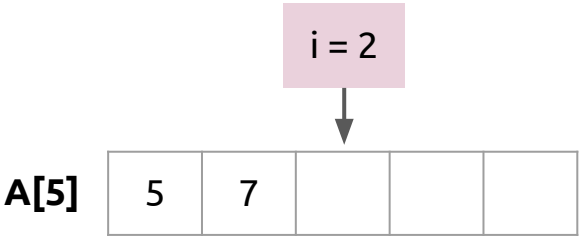
i = 2

A[5]	5	7			

Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: **0**

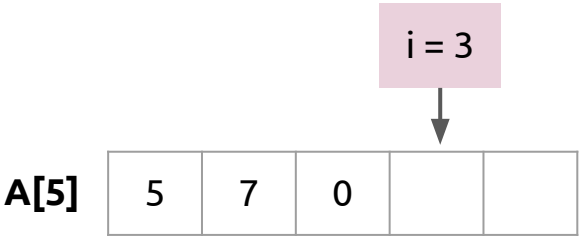
used[10]	0	0	0	0	0	1	0	1	0	0



Μοναδικοί τυχαίοι αριθμοί

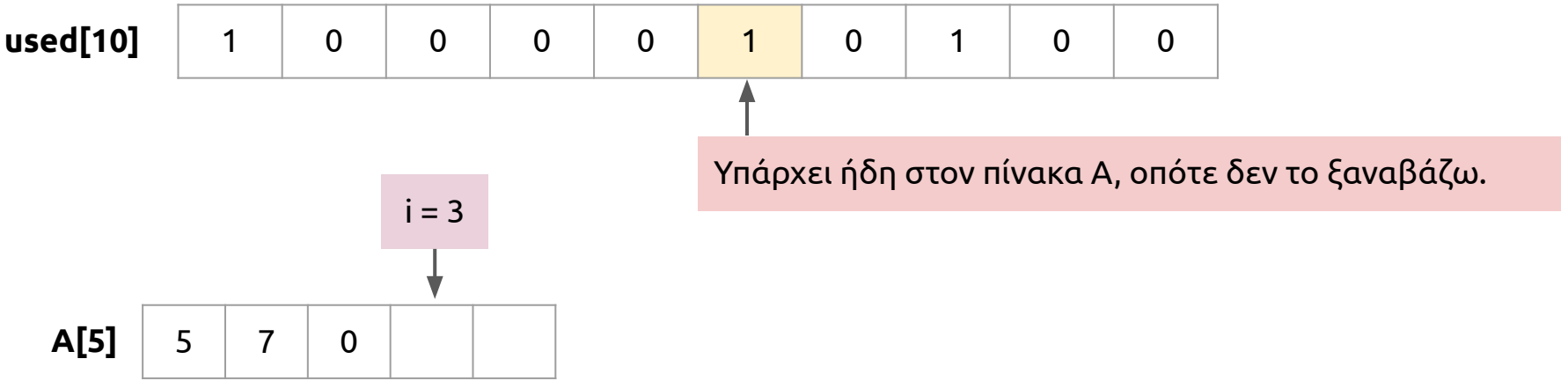
Τυχαίος Αριθμός: **0**

used[10]	1	0	0	0	0	1	0	1	0	0



Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: **5**



Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: 4

used[10]	1	0	0	0	0	1	0	1	0	0

i = 3

A[5]	5	7	0		

Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: 4

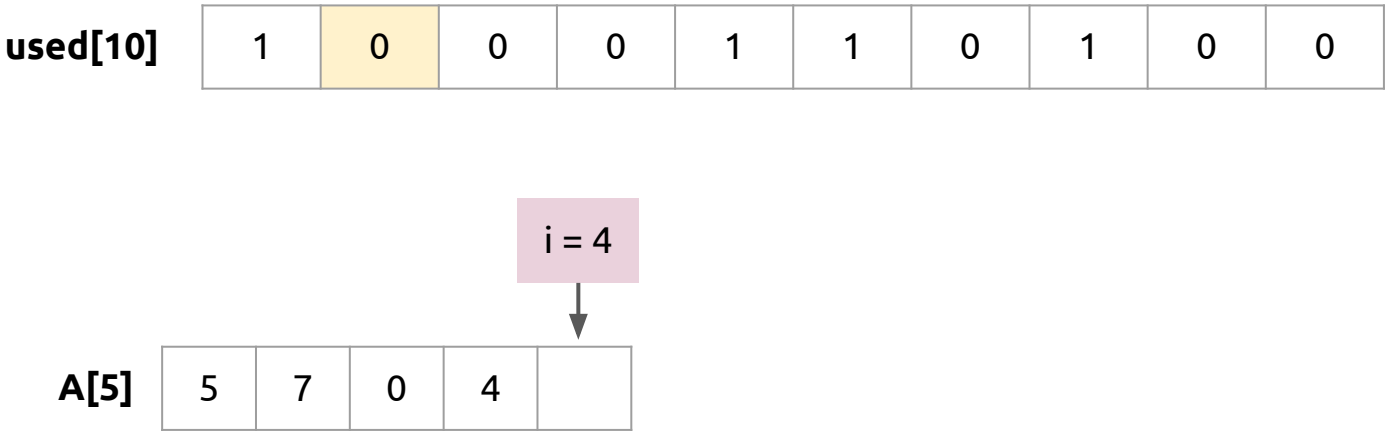
used[10]	1	0	0	0	1	1	0	1	0	0

i = 4

A[5]	5	7	0	4	

Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: 1



Μοναδικοί τυχαίοι αριθμοί

Τυχαίος Αριθμός: 1

used[10]	1	1	0	0	1	1	0	1	0	0

i = 5

A[5]	5	7	0	4	1

Μοναδικοί τυχαίοι αριθμοί


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    int A[5];
    int used[10] = {0};
    int count = 0;
    srand(time(0));
    while (count < 5) {
        int num = rand() % 10;
        if (used[num] == 0) {
            A[count] = num;
            count++;
            used[num] = 1;
        }
    }
    printf("Πίνακας: ");
    for (int i = 0; i < 5; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

Με χρήση Συνάρτησης


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void fill_unique_random(int A[], int size){
    int used[10] = {0};
    int count = 0;
    while (count < size) {
        int num = rand() % 10;
        if (used[num] == 0) {
            A[count] = num;
            count++;
            used[num] = 1;
        }
    }
}
```

```c
int main() {
    int A[5];

    srand(time(0));
    fill_unique_random(A, 5);

    printf("Πίνακας: ");
    for (int i = 0; i < 5; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

Για άγνωστο μέγεθος πίνακα


```c
int main() {
    int size;
    printf("Δώσε μέγεθος πίνακα: ");
    scanf("%d", &size);
    int A[size];

    srand(time(0));
    fill_unique_random(A, size);

    printf("Πίνακας: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", A[i]);
    }
    return 0;
}
```

Δημιουργία του πίνακα στη μνήμη STACK

Για άγνωστο μέγεθος πίνακα - malloc

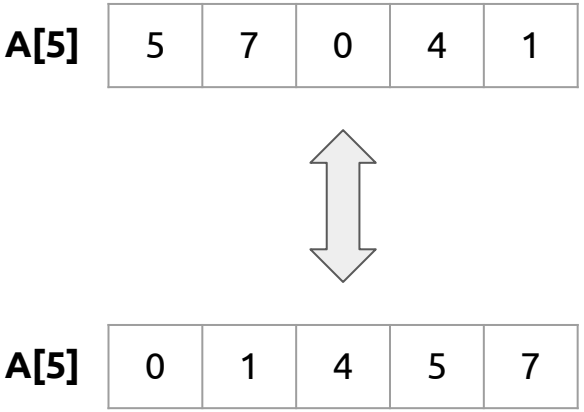

```c
int main() {
    int size;
    printf("Δώσε μέγεθος πίνακα: ");
    scanf("%d", &size);

    int *A = malloc(size * sizeof(int));

    if (A == NULL) {
        printf("Σφάλμα: Ανεπαρκής μνήμη.\n");
        return 1;
    }

    srand(time(0));
    fill_unique_random(A, size);
```

- Δυναμική παραχώρηση μνήμης
- Δημιουργία του πίνακα στη μνήμη HEAP

```c
    printf("Πίνακας: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    free(A);
    return 0;
}
```

Πίνακας σε Αύξουσα σειρά

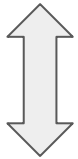
Αφού γεμίσαμε τον πίνακα Α με μοναδικούς τυχαίους αριθμούς, θέλουμε να αναδιατάξουμε τα στοιχεία του σε αύξουσα σειρά, χωρίς να χρησιμοποιηθεί κάποιος αλγόριθμος ταξινόμησης.



Πίνακας σε Αύξουσα σειρά

Αφού γεμίσαμε τον πίνακα Α με μοναδικούς τυχαίους αριθμούς, θέλουμε να αναδιατάξουμε τα στοιχεία του σε αύξουσα σειρά, χωρίς να χρησιμοποιηθεί κάποιος αλγόριθμος ταξινόμησης.

A[5]	5	7	0	4	1



A[5]	0	1	4	5	7

- Γνωρίζουμε ότι ο πίνακας περιέχει αριθμούς στο διάστημα **0-9**
- Έχουμε ήδη συμπληρώσει τον πίνακα **used[10]**

used[10]	1	1	0	0	1	1	0	1	0	0

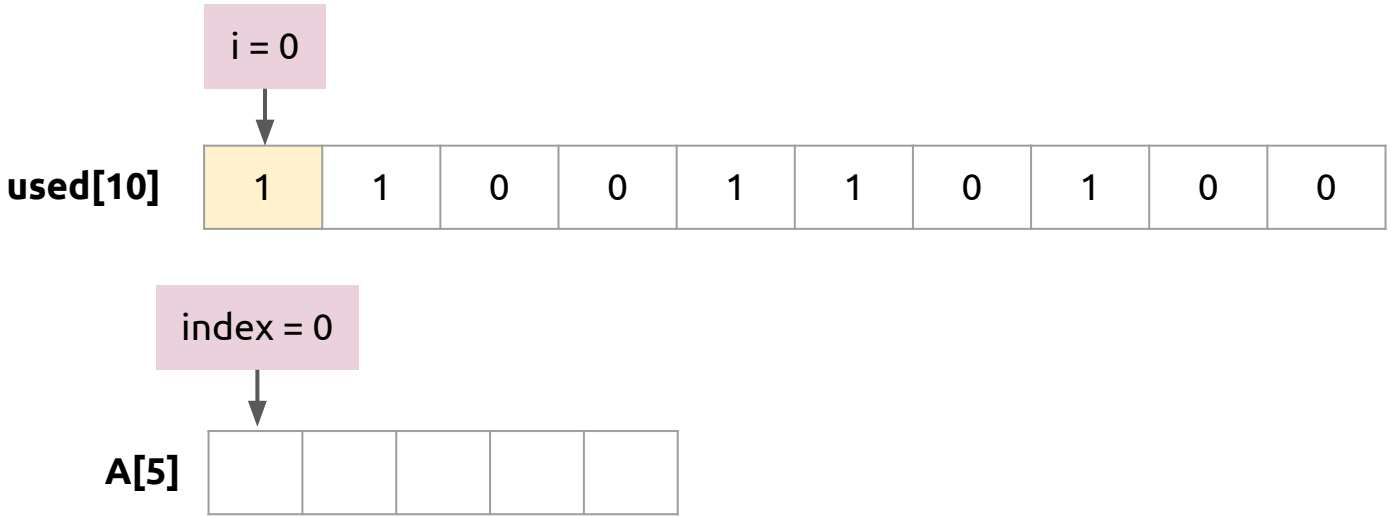
Τροποποίηση Συνάρτησης fill_unique_random


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void fill_unique_random(int A[], int size,
                        int *used){
    int count = 0;
    while (count < size) {
        int num = rand() % 10;
        if (used[num] == 0) {
            A[count] = num;
            count++;
            used[num] = 1;
        }
    }
}
```

```c
int main() {
    int A[5];
    int used[10] = {0};

    srand(time(0));
    fill_unique_random(A, 5, used);

    printf("Πίνακας: ");
    for (int i = 0; i < 5; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

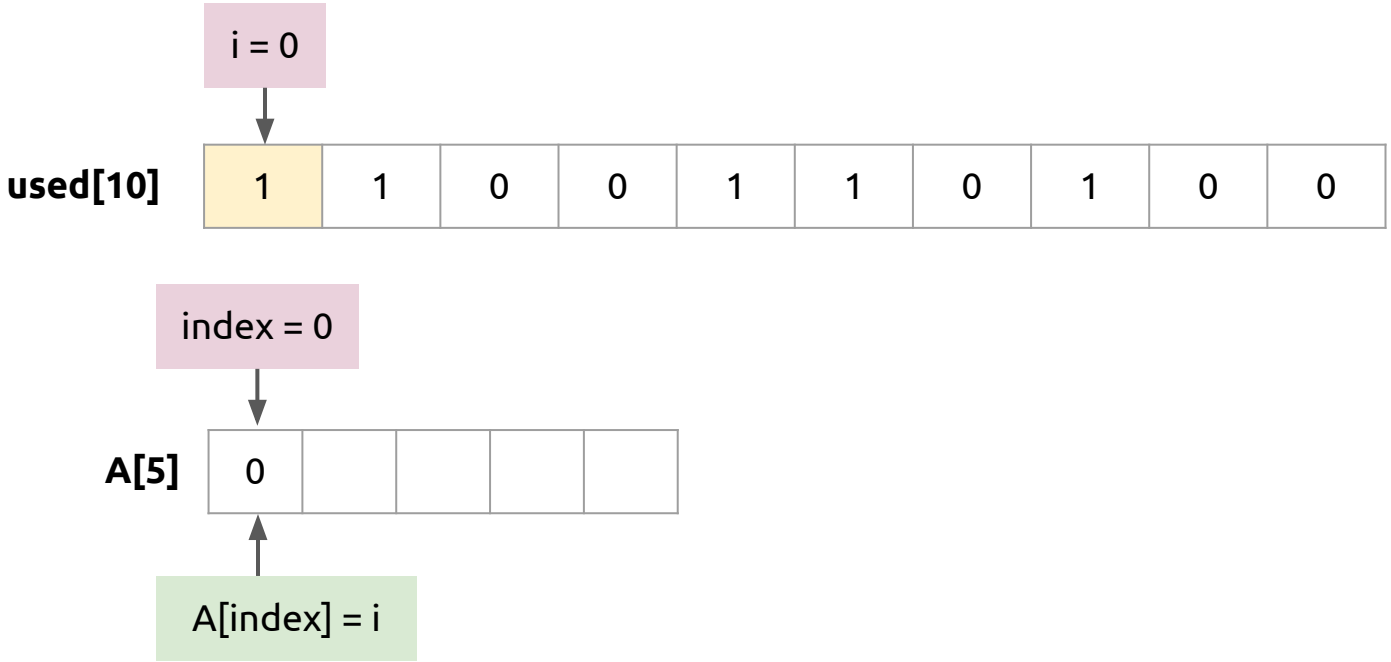
Πίνακας σε Αύξουσα σειρά

Ο πίνακας used[10] "περιέχει" τα στοιχεία του πίνακα A[5], οπότε με ασφάλεια "διαγράφουμε" τον πίνακα A[5]



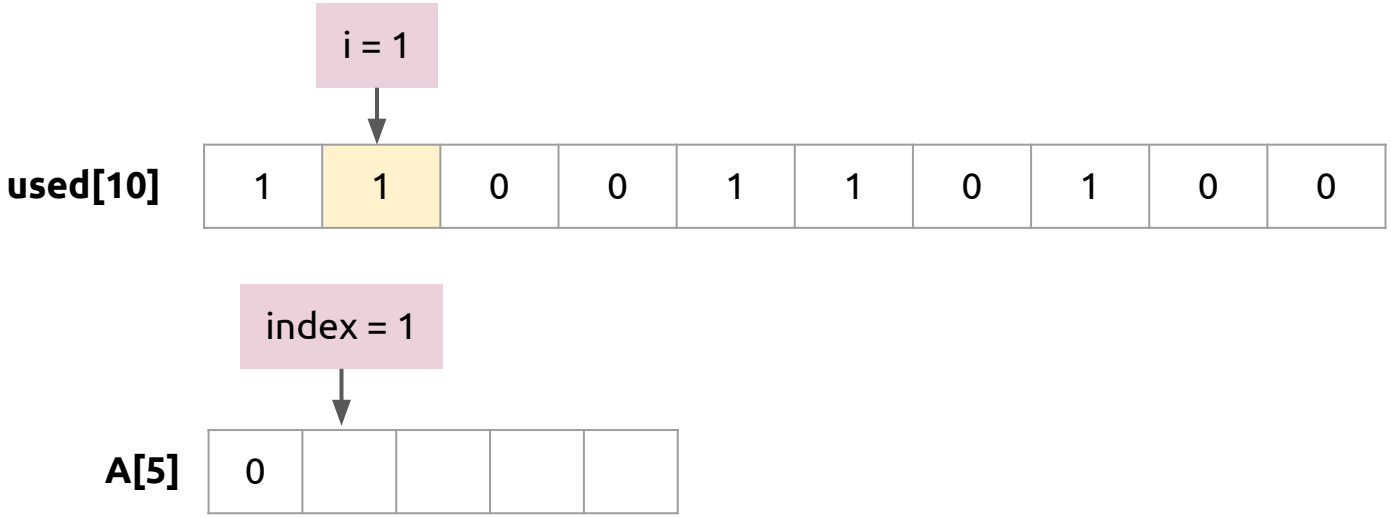
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



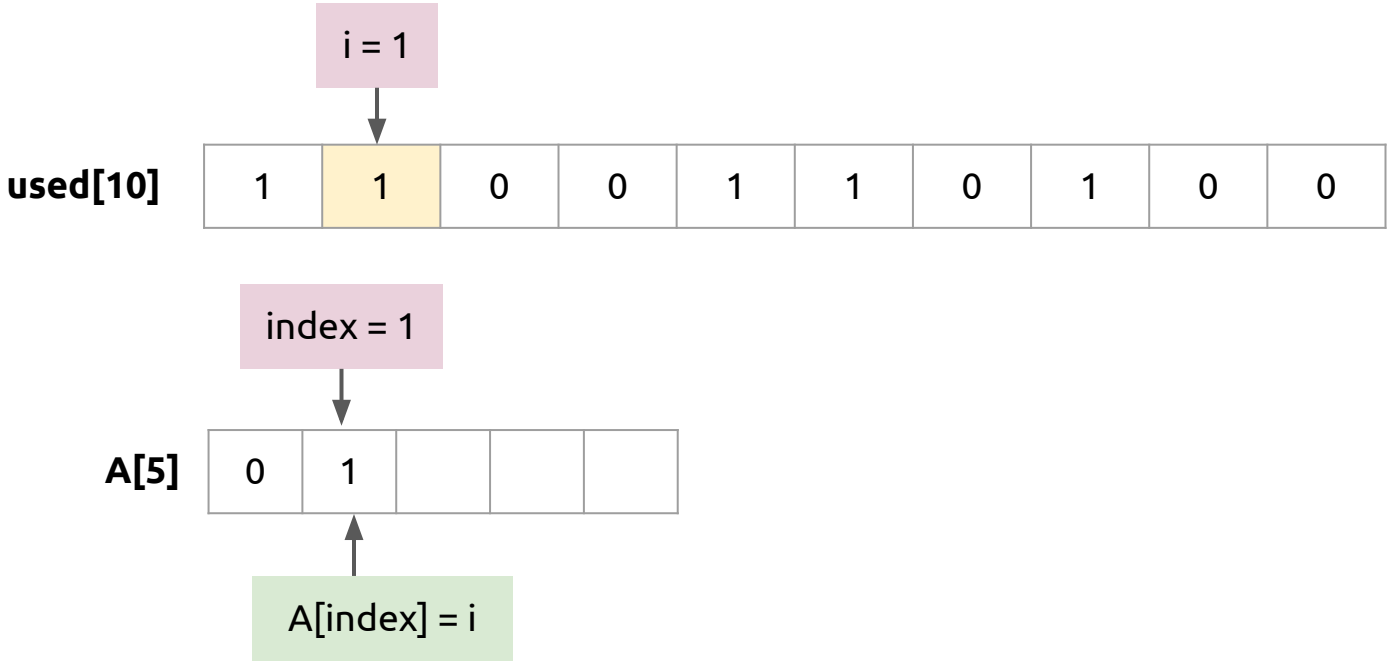
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



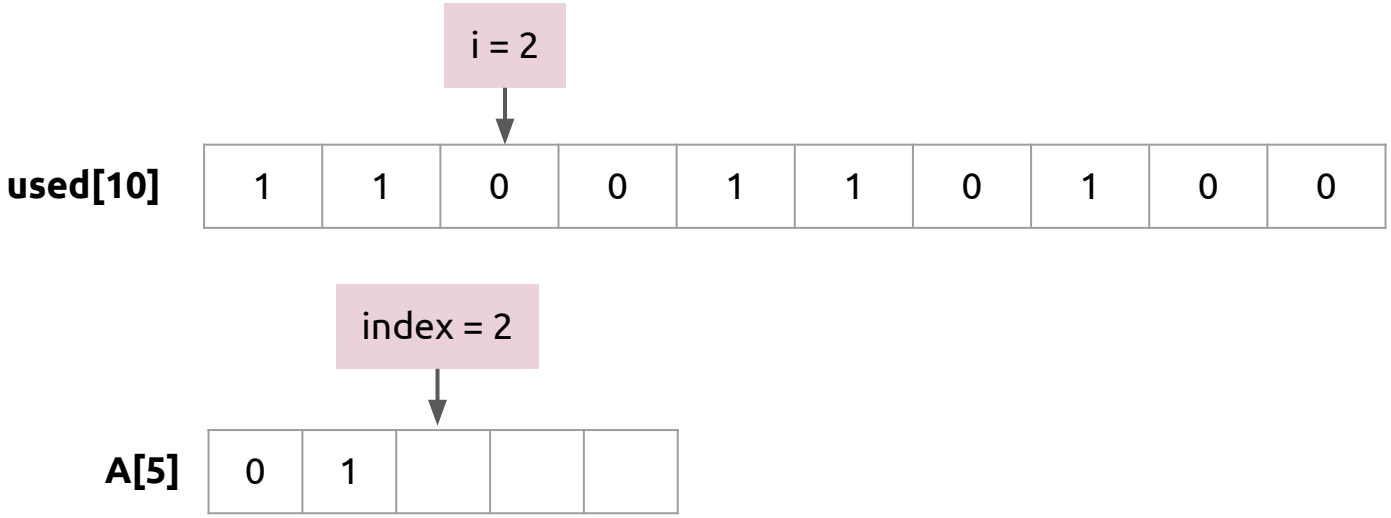
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



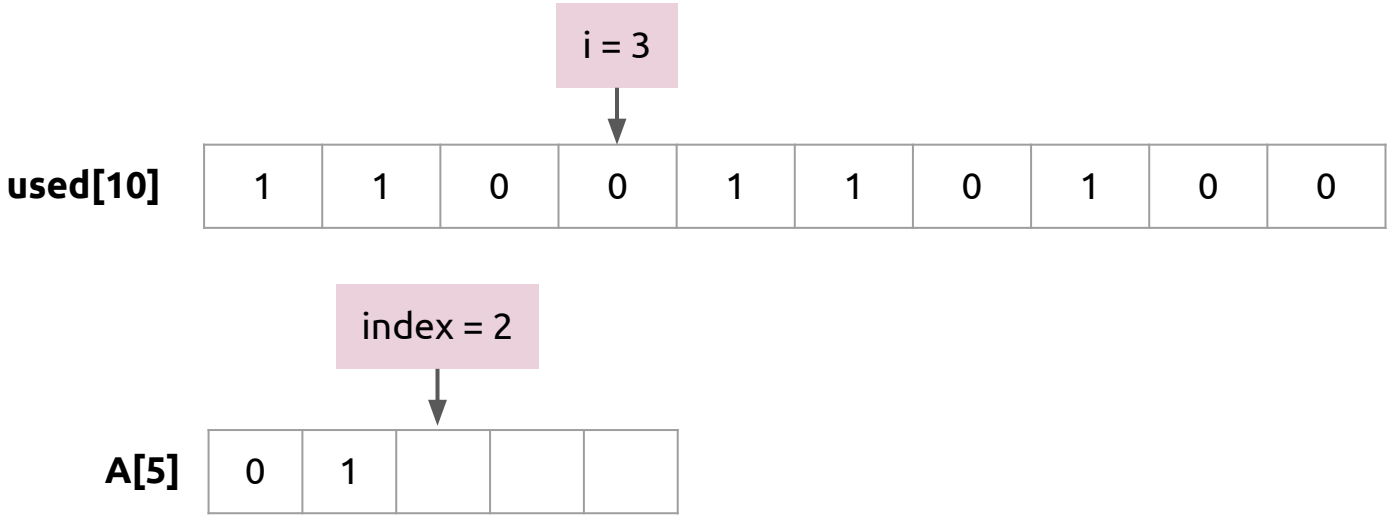
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



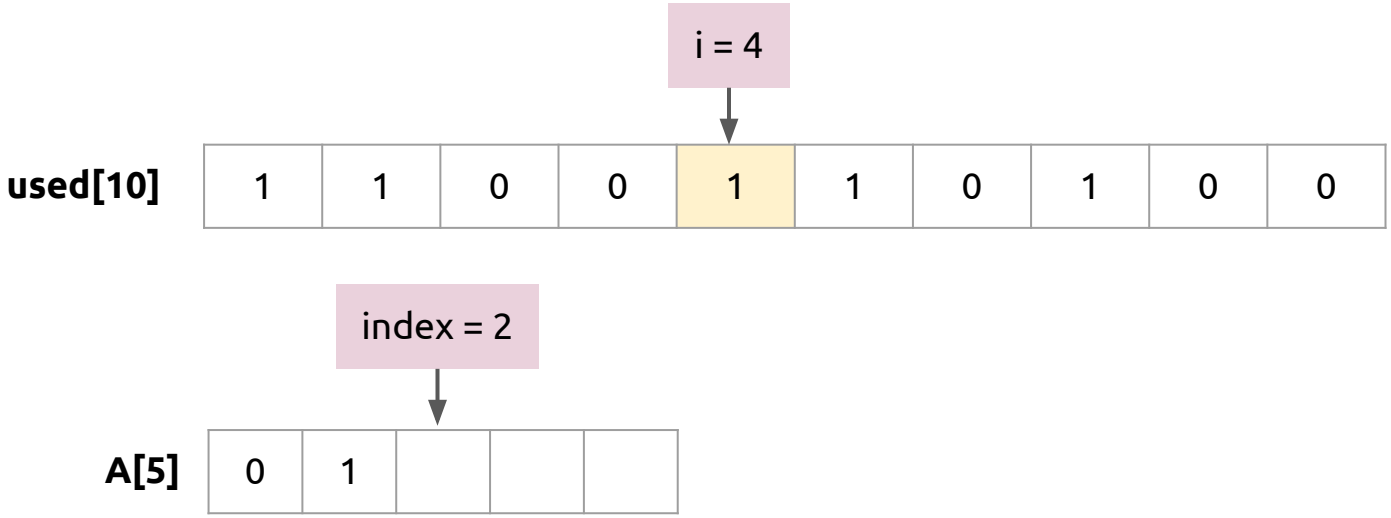
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



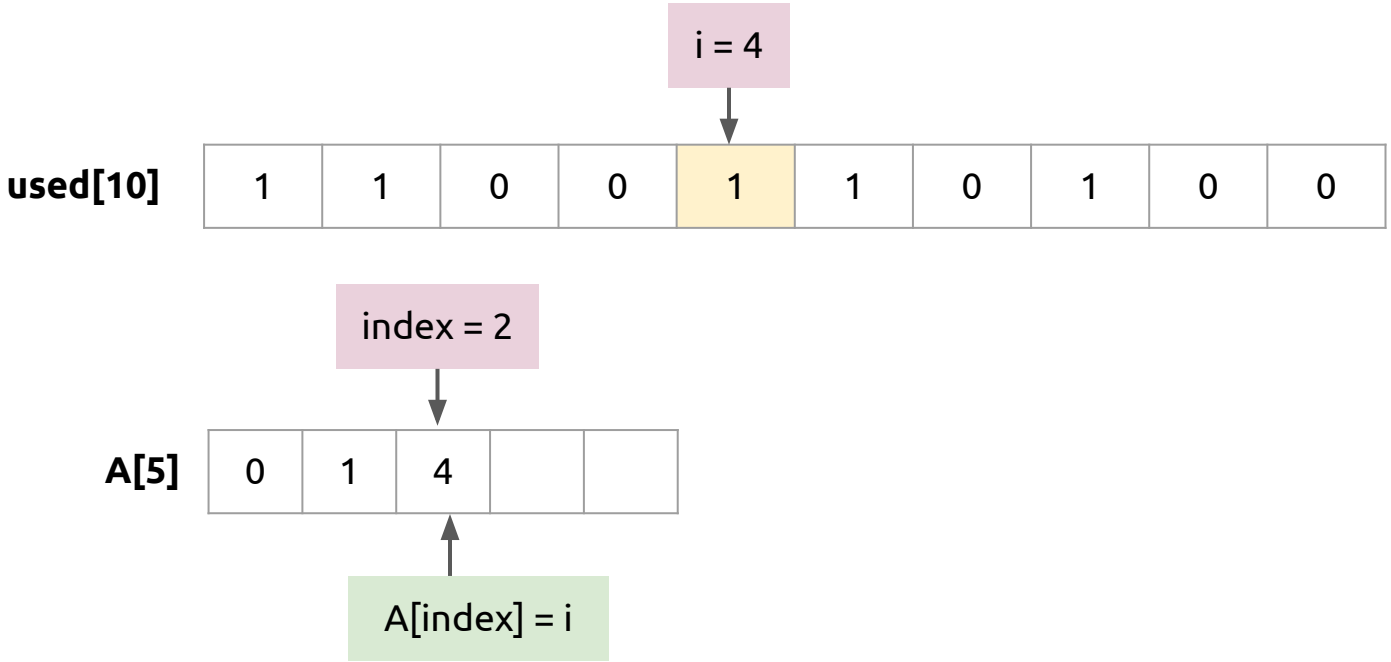
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



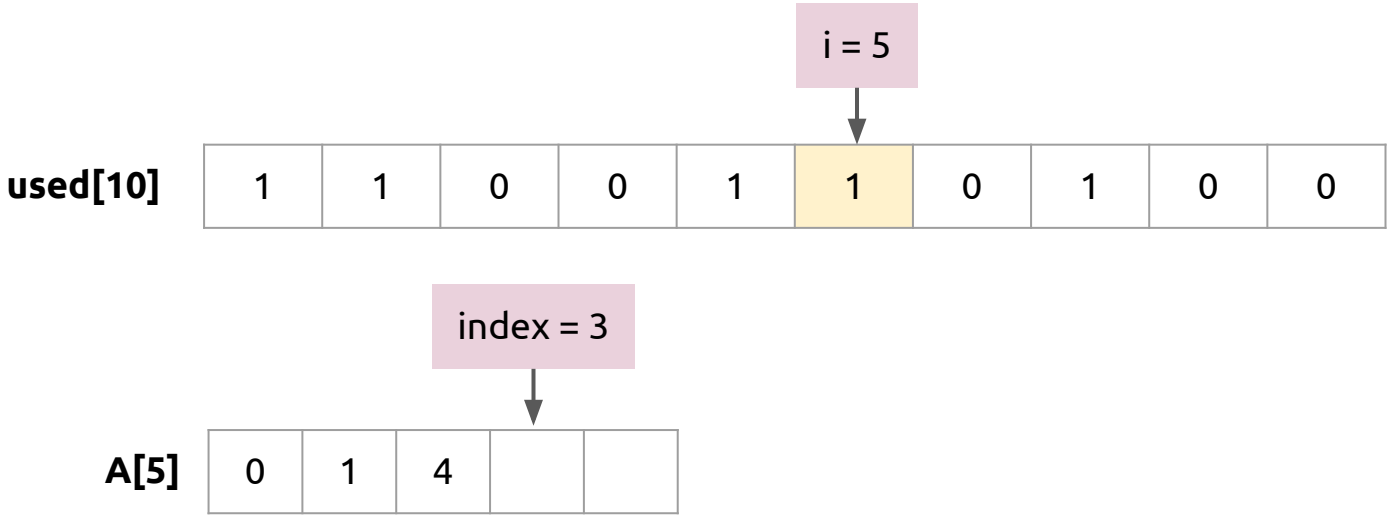
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



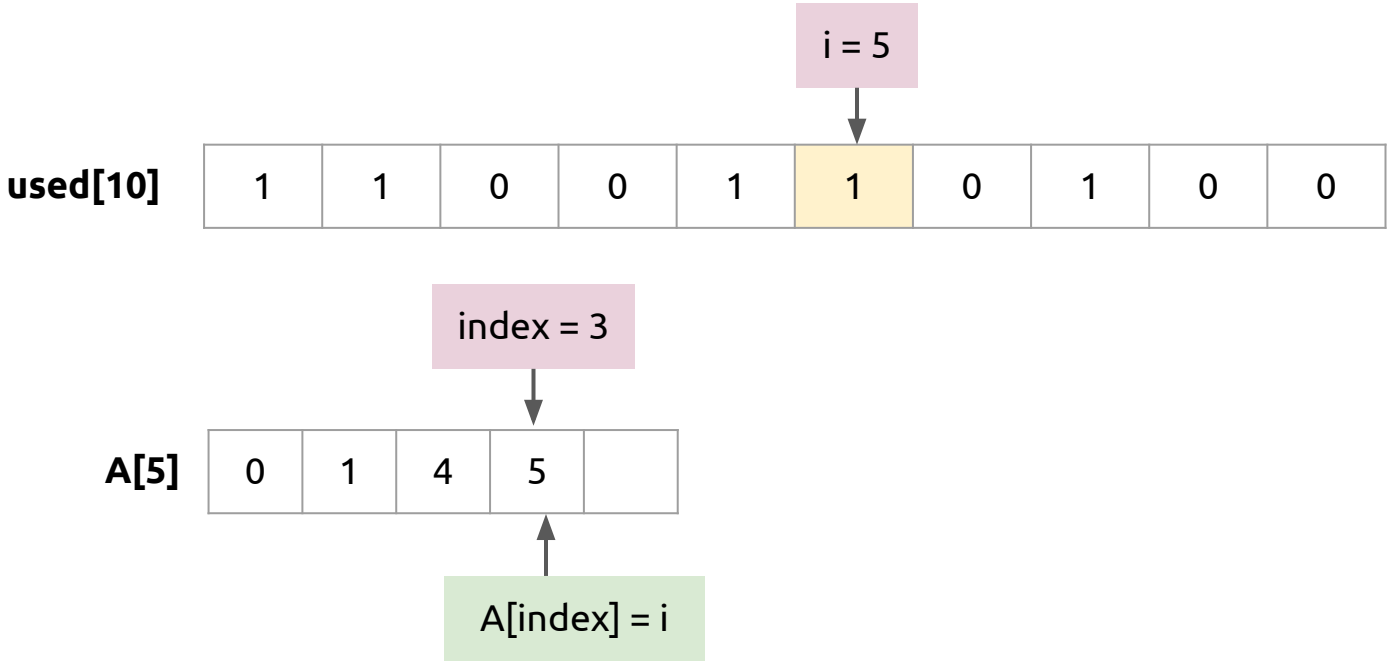
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



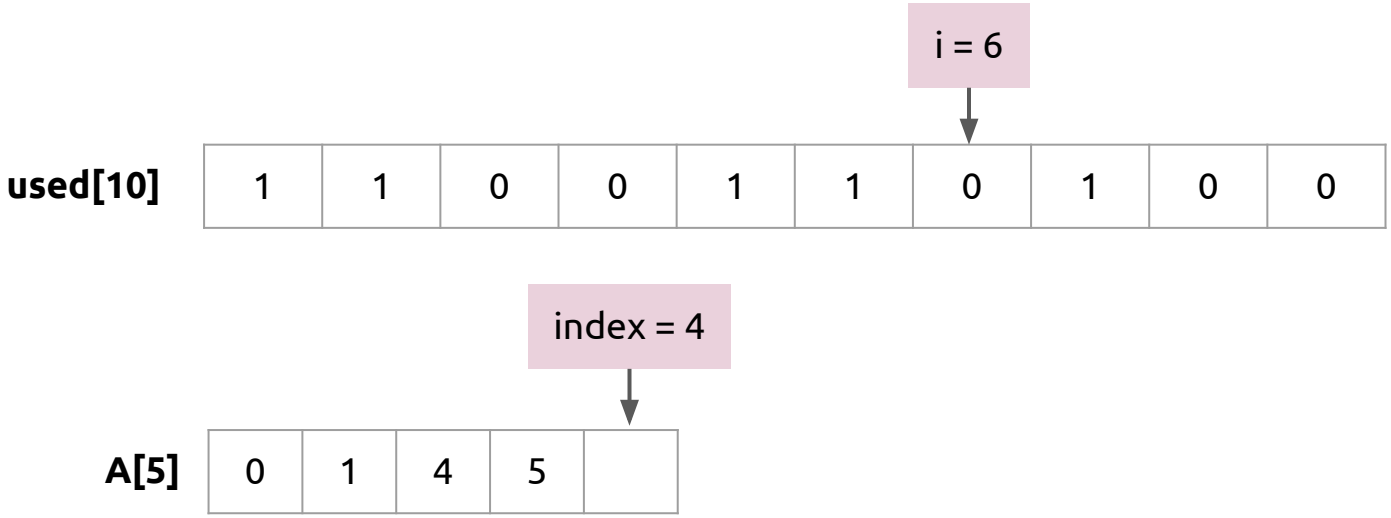
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



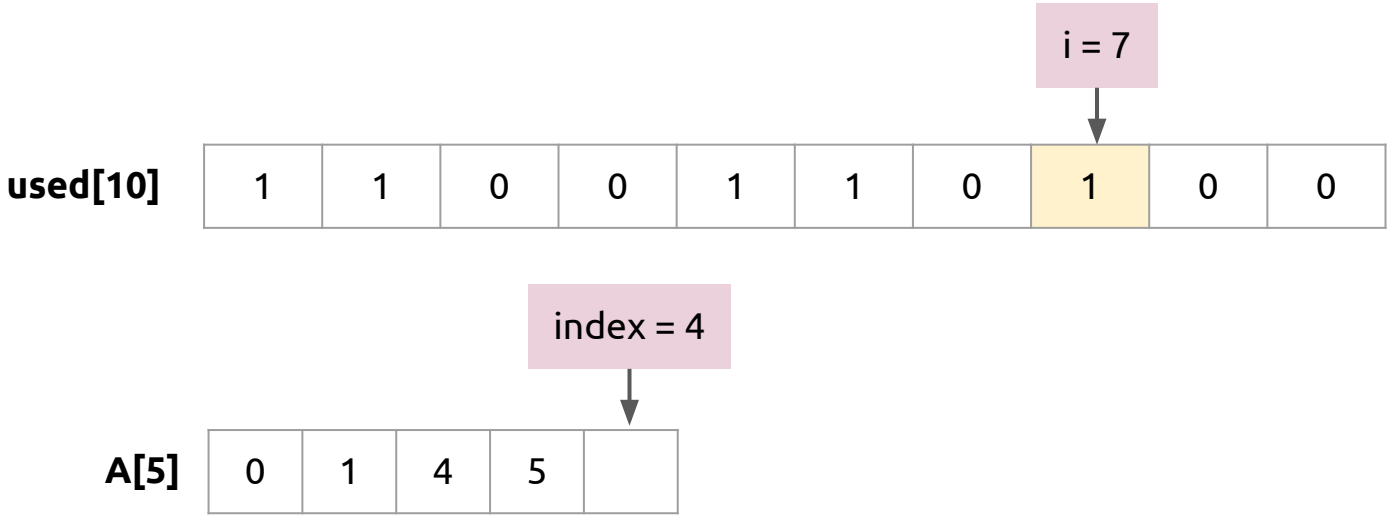
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



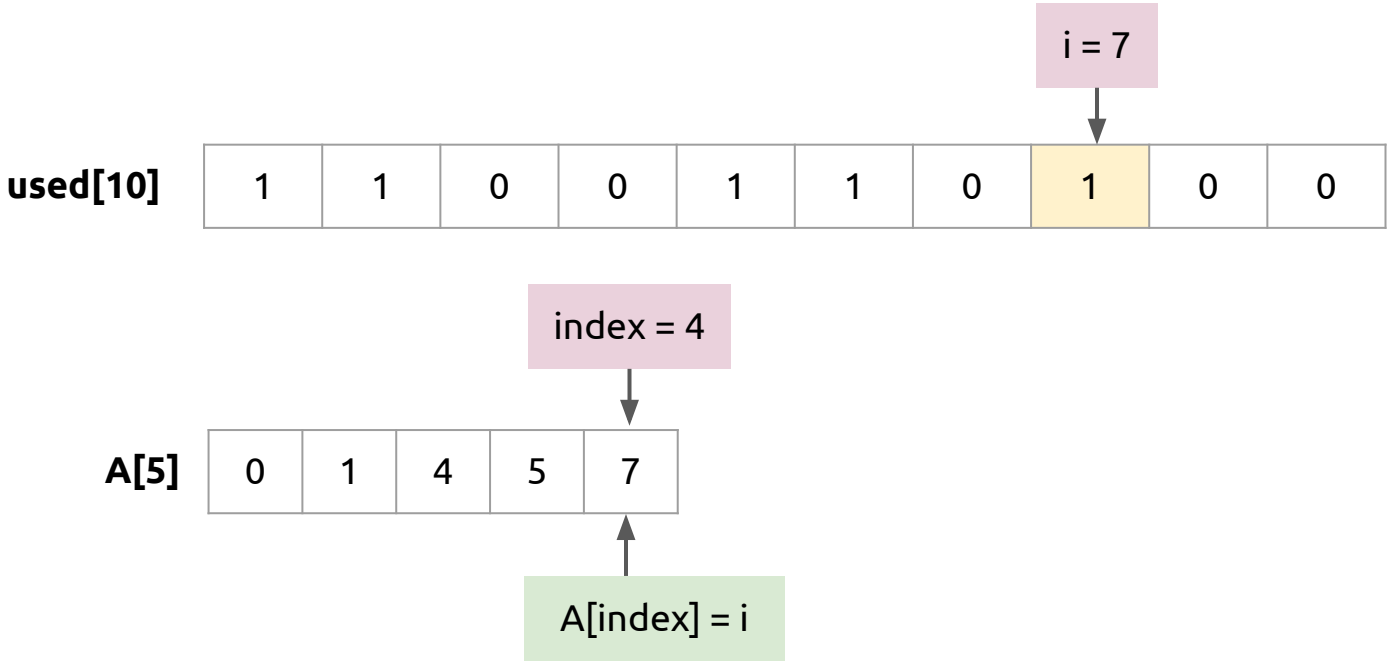
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



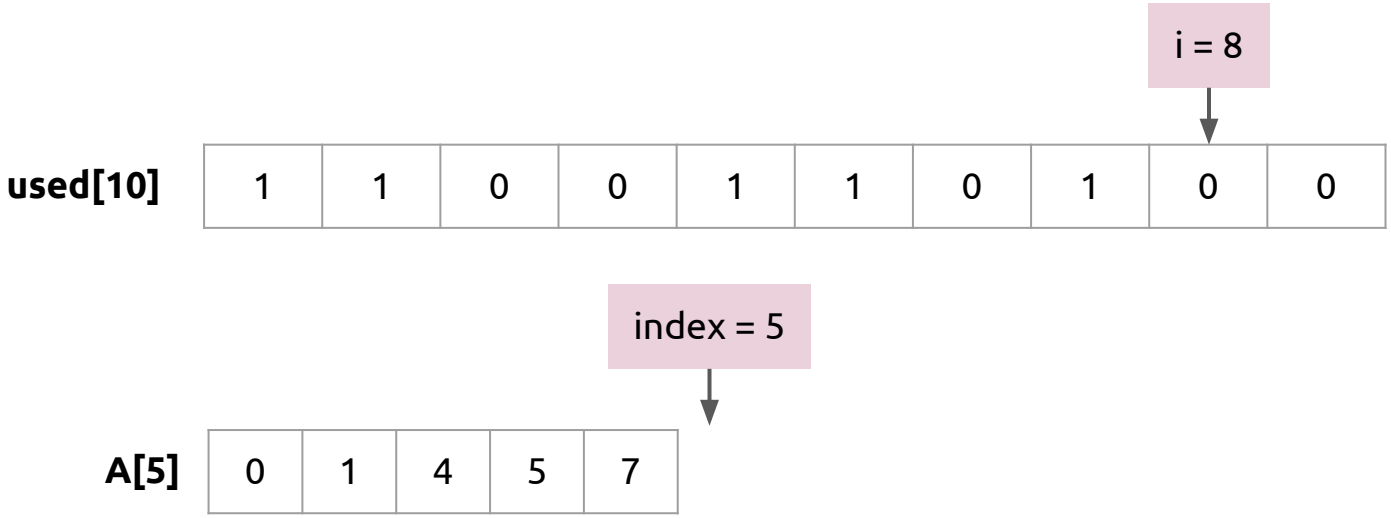
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



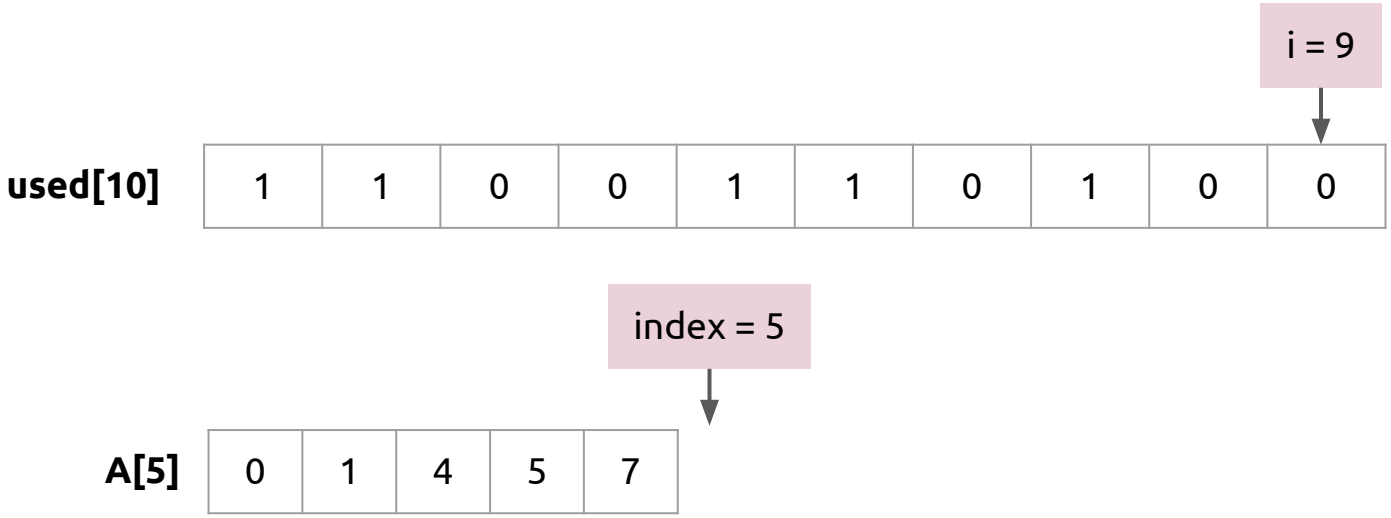
Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



Πίνακας σε Αύξουσα σειρά

Η μεταβλητή i, δείχνει το τρέχον στοιχείο του πίνακα used[10] και η μεταβλητή index, το τρέχον στοιχείο του πίνακα A[5].



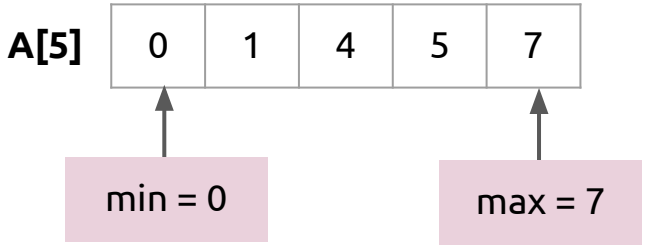
Συνάρτηση reorder_ascending


```c
void reorder_ascending(int A[], int size,
                       int used[]) {
   int index = 0;
   for (int i = 0; i < size; i++) {
       if (used[i] == 1) {
           A[index] = i;
           index++;
       }
   }
}
```

```c
int main() {
    int A[5];
    int used[10] = {0};

    srand(time(0));
    fill_unique_random(A, 5, used);
    reorder_ascending(A, 10, used);

    printf("Πίνακας: ");
    for (int i = 0; i < 5; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");
    return 0;
}
```

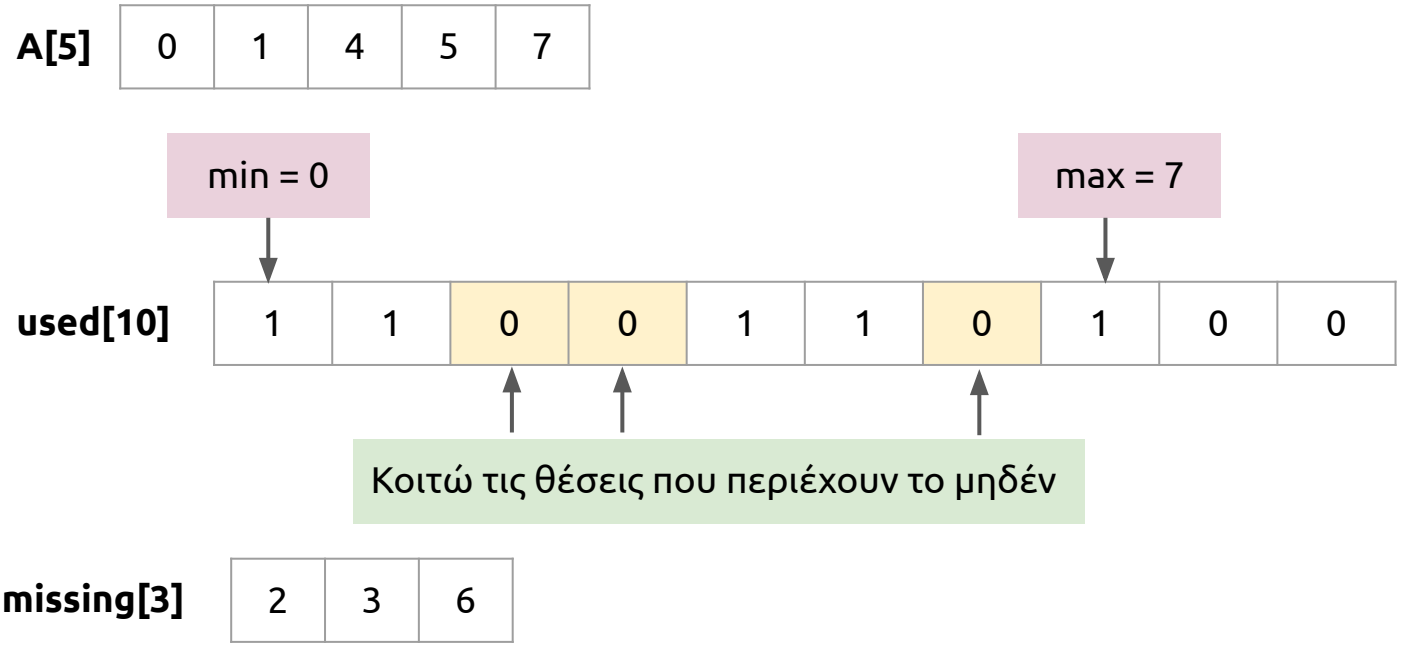
Τιμές που λείπουν στον Πίνακα

Να γραφεί συνάρτηση που να επιστρέφει σε μορφή πίνακα, τις τιμές που λείπουν, στο διάστημα τιμών που οριοθετούν το πρώτο και το τελευταίο στοιχείο του νέου πίνακα.



Τιμές που λείπουν στον Πίνακα

Να γραφεί συνάρτηση που να επιστρέφει σε μορφή πίνακα, τις τιμές που λείπουν, στο διάστημα τιμών που οριοθετούν το πρώτο και το τελευταίο στοιχείο του νέου πίνακα.



Συνάρτηση missing_values


```
int missing_values(int used[], int min,
                   int max, int missing[]) {
   int count = 0;
   for (int i = min; i <= max; i++) {
       if (used[i] == 0) {
           missing[count] = i;
           count++;
       }
   }
   return count;
}
```

Συνάρτηση main (1/2)


```c
int main() {

    int A[5];
    int used[10] = {0};
    int missing[10];

    srand(time(0));
    fill_unique_random(A, 5, used);

    printf("Αρχικός πίνακας: ");
    for (int i = 0; i < size; i++)
       printf("%d ", A[i]);
    printf("\n");

    reorder_ascending(A, 5, used);
```

Συνάρτηση main (2/2)


```
    printf("Ταξινομημένος πίνακας: ");
    for (int i = 0; i < size; i++)
       printf("%d ", A[i]);
    printf("\n");
    int min = A[0];
    int max = A[size - 1];
    int m_count = find_missing_values(used, min, max, missing);
    if (m_count > 0) {
        printf("Αριθμοί που λείπουν: ");
        for (int i = 0; i < m_count; i++)
           printf("%d ", missing[i]);
        printf("\n");
    } else
        printf("Δεν λείπει κανένας αριθμός.\n");
    return 0;
}
```

Άσκηση

Δύο μονοδιάστατοι αριθμητικοί πίνακες θετικών ακεραίων αριθμών, έστω `a, b` με πλήθος στοιχείων `m, n` αντίστοιχα, περιέχουν θετικούς ακέραιους αριθμούς στην περιοχή [`0-99`]. Να εξασφαλίσετε ότι: κάθε ακέραιος αριθμός εμφανίζεται μόνον μία φορά σε κάθε πίνακα και ότι οι δύο πίνακες `a, b` δεν έχουν κανένα κοινό στοιχείο.

Να γραφεί μια συνάρτηση με όνομα `check_successive` με ορίσματα εισόδου τους πίνακες `a, b`. Η συνάρτηση αυτή να επιστρέφει στη συνάρτηση `main()`:

- Έναν νέο πίνακα που θα περιέχει σε αύξουσα διάταξη, ΧΩΡΙΣ να χρησιμοποιηθεί διαδικασία ταξινόμησης, όλα τα στοιχεία των δύο πινάκων `a, b`.
- Αν τα στοιχεία του νέου πίνακα δεν είναι απολύτως διαδοχικά θα πρέπει να επιστρέφονται επίσης σε μορφή μονοδιάστατου αριθμητικού πίνακα οι τιμές που λείπουν, στο διάστημα τιμών που οριοθετούν το πρώτο και το τελευταίο στοιχείο του νέου πίνακα.

(ΠΡΟΣΟΧΗ: η συνάρτηση ΔΕΝ θα πρέπει να περιέχει εντολές `printf`)

Άσκηση

Στη συνέχεια να γραφεί ένα πρόγραμμα σε γλώσσα C που :

- θα γεμίζει τους πίνακες `a, b` με κατάλληλη χρήση της συνάρτησης δημιουργίας τυχαίων αριθμών `rand()` ώστε οι τιμές να ανήκουν στο διάστημα [`0-99`]. Το πλήθος των στοιχείων κάθε πίνακα θα προσδιορίζεται κατά τη διάρκεια της εκτέλεσης της `main()`.
- θα καλεί τη συνάρτηση `check_successive` και θα εμφανίζει τα σχετικά αποτελέσματα.

Παράδειγμα με `m = 8` και `n = 6`:

Πίνακας `a`:

14	13	8	17	21	20	19	9

Πίνακας `b`:

7	5	22	11	16	25

Αποτελέσματα στη `main()`:

Ο Νέος Πίνακας είναι:

5	7	8	9	11	13	14	16	17	19	20	21	22	25

Πίνακας με τις τιμές που λείπουν στο διάστημα [`5, 25`]:

6	10	12	15	18	23	24

Άσκηση - fill_unique_random_array


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define RANGE 100 // τιμές στο [0-99]

void fill_unique_random_array(int *arr, int size, int *used) {
   int i = 0;
   while (i < size) {
       int num = rand() % RANGE;
       if (used[num] == 0) {
           used[num] = 1;
           arr[i] = num;
           i++;
       }
   }
}
```

Άσκηση - check_successive


```
void check_successive(int *merged, int *used) {
  int index = 0;
  for (int i = 0; i < RANGE; i++) {
      if (used[i] == 1) {
          merged[index] = i;
          index++;
      }
  }
}
```

Άσκηση - missing_values


```
int missing_values(int used[], int min, int max, int missing[]) {
  int count = 0;
  for (int i = min; i <= max; i++) {
      if (used[i] == 0) {
          missing[count] = i;
          count++;
      }
  }
  return count;
}
```

Άσκηση - main (1/3)


```c
int main() {
    srand(time(NULL));
    int n, m;
    printf("Δώσε το μέγεθος του πρώτου πίνακα: ");
    scanf("%d", &n);
    printf("Δώσε το μέγεθος του δεύτερου πίνακα: ");
    scanf("%d", &m);


    int a[n], b[m], merged[n+m], missing[n+m];
    int used[RANGE] = {0};

    fill_unique_random_array(a, n, used);
    fill_unique_random_array(b, m, used);
    check_successive(merged, used);
```

Άσκηση - main (2/3)


```c
printf("Πίνακας A: ");
for (int i = 0; i < n; i++) printf("%d ", a[i]);
printf("\n");

printf("Πίνακας B: ");
for (int i = 0; i < m; i++) printf("%d ", b[i]);
printf("\n");

printf("Συγχωνευμένος πίνακας (αύξουσα σειρά): ");
for (int i = 0; i < n+m; i++) {
    printf("%d ", merged[i]);
}
printf("\n");
```

Άσκηση - main (3/3)


```
    int min = merged[0];
    int max = merged[n+m - 1];

    int missingSize = missing_values(used, min, max, missing);

    if (missingSize > 0) {
        printf("Απουσιάζουν οι διαδοχικοί αριθμοί: ");
        for (int i = 0; i < missingSize; i++) {
            printf("%d ", missing[i]);
        }
        printf("\n");
    } else {
        printf("Όλα τα στοιχεία είναι διαδοχικά.\n");
    }
    return 0;
}
```