Mảng

Duc-Minh VU @ ORLab - Phenikaa

Mảng

- Tập hợp dữ liệu cùng kiểu, được lưu trữ liên tiếp trong bộ nhớ.
- Các thao tác trên mảng:
  - Truy xuất phần tử theo chỉ số
  - Duyệt
  - Chèn phần tử mới
  - Xóa phần tử khỏi mảng
  - Trộn mảng
  - Tìm kiếm
  - Sắp xếp

```
Algorithm: Traverse (A, N)
[A is the name of the array and N is the number of elements of the
array]
1. Set I=0
2. Repeat steps 3 and 4 while I < N
3.    Visit A[I]
4.    Set I=I+1
   [End of loop]
5. Return
```

Duyệt qua từng phần tử của mảng, mỗi phần tử một lần.

Thực hiện các thao tác tìm kiếm.

Độ phức tạp: O(N)

Duyệt mảng/tìm kiếm trên mảng

```
Algorithm: Insert (A, N, KEY)
[A is the sorted array of N elements and KEY is the inserted item]
1. Set I = N - 1
2. Repeat steps 3 and 4 while A[I]>KEY
3.     Set A[I+1]=A[I]
4.     Set I=I-1
   [End of loop]
5. Set A[I+1]=KEY
6. Set N=N+1
7. Return
```

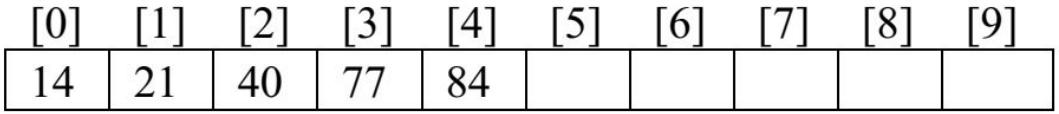
Chèn phần tử X vào vị trí thứ k:

+ Dịch chuyển các phần tử từ vị trí thứ k sang phải một đơn vị.
+ Cập nhật A[k] = X.

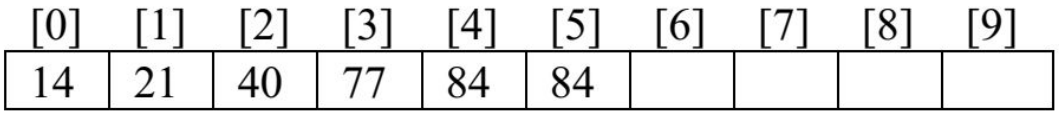
Độ phức tạp: O(N)

Chèn phần tử vào mảng

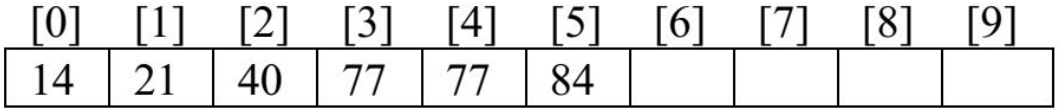
Insert 50 into a sorted array with 5 elements.



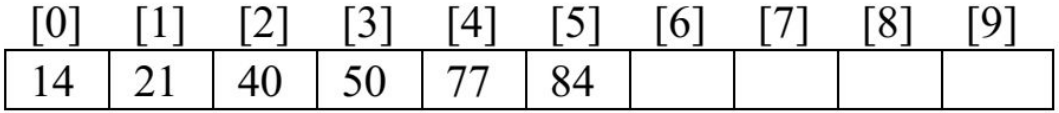
a) Initial Array



b) Move from 4th position to 5th position



c) Move from 3rd position to 4th position



d) Insert in 3rd position since 50 is greater than 2nd element

Figure 3.6: (a-d) Insertion in the sorted Array

Chèn 50 vào vị trí thứ 3 của mảng

```
Algorithm: Delete (A, N, P, KEY)
[A is an of N number of elements, P is the position of the deleted
item; KEY is the deleted item returns by reference]
1. Set I = P
2. KEY = A[P]
3. Repeat steps 4 and 5 while I < N - 1
4.     Set A[I] = A[I+1]
5.     Set I = I + 1
   [End of loop]
6. Set N = N - 1
7. Return
```

- Xóa theo khóa (cần tìm vị trí khóa để xóa).
- Xóa theo chỉ số
- Cả 2 trường hợp:
  - Tìm đến vị trí cần xóa.
  - Xóa phần tử bằng cách dịch các phần tử bên phải sang một đơn vị

Xóa phần tử khỏi mảng

```
Algorithm: Delete (A, N, KEY)
[A is the name of the array, N is the number of elements of the
array, P is the position of the deleted item, KEY is the deleted
item]
1. Set I = 0
2. Repeat while A[I] ≠ KEY AND I < N
3.     Set I = I + 1
   [End of loop]
4. Repeat steps 5 and 6 while I < N-1
5.    Set A[I] = A[I+1]
6.    Set I = I + 1
   [End of loop]
7. Set N = N - 1
8. Return
```

Xóa phần tử khỏi mảng

Delete an element from 3rd position (i.e. 21) of the array with 6 elements.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
40	84	50	21	14	77				

a) Initial Array

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
40	84	50	14	14	77				

b) Move from 4th position to 3rd position

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
40	84	50	14	77	77				

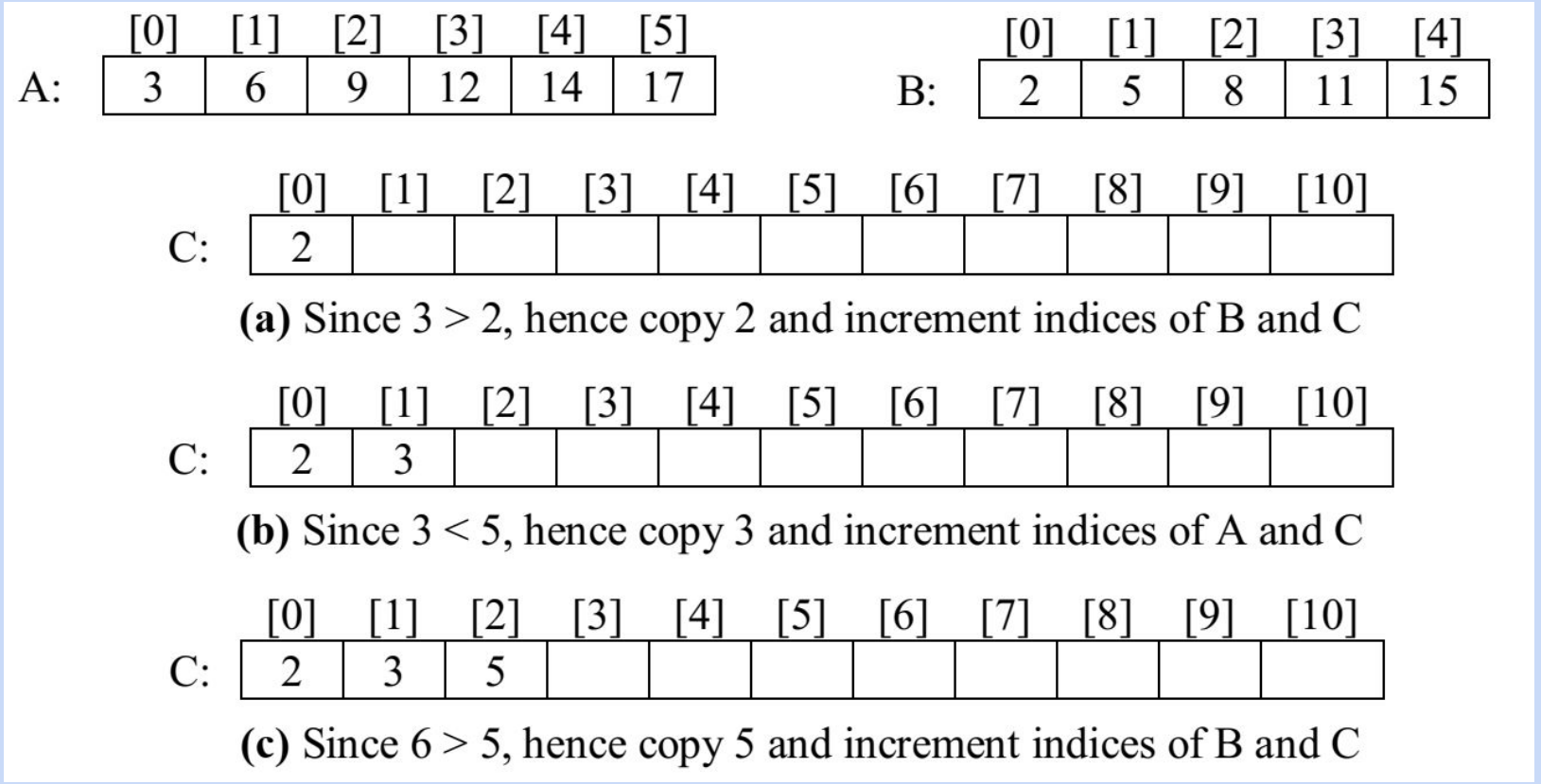
c) Move from 5th position to 4th position

Figure 3.7: (a-c) Delete element from a specific position

Xóa phần tử có chỉ số thứ 3

Trộn hai dãy tăng dần

1. Cho hai dãy tăng dần, tạo ra một dãy tăng dần mới gồm các phần tử là của hai dãy ban đầu.
    1.1. Ví dụ [1, 2, 4], [2, 3, 5] => [1, 2, 2, 3, 4, 5]
2. Thuật toán:
    2.1. Tạo mảng kết quả rỗng.
    2.2. Đưa phần tử bé nhất của 2 mảng vào mảng kết quả;
    2.3. Xóa phần tử bé nhất ở trong mảng tương ứng kể trên.
    2.4. Lặp lại bước 2 cho đến khi 1 trong 2 mảng rỗng.
    2.5. Đưa các phần tử của mảng còn lại vào dãy kết quả.
    2.6. Trả về dãy kết quả



Trộn hai dãy tăng dần

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6							

(d) Since 6 < 8, hence copy 6 and increment indices of A and C

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6	8						

(e) Since 9 > 8, hence copy 8 and increment indices of B and C

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6	8	9					

(f) Since 9 < 11, hence copy 9 and increment indices of A and C

Trộn hai dãy tăng dần

	[0]	[1]	[2]	[3]	[4]	[5]
A:	3	6	9	12	14	17

	[0]	[1]	[2]	[3]	[4]
B:	2	5	8	11	15

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6	8	9	11				

(g) Since 12 > 11, hence copy 11 and increment indices of B and C

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6	8	9	11	12			

(h) Since 12 < 15, hence copy 12 and increment indices of A and C

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6	8	9	11	12	14		

(i) Since 14 < 15, hence copy 14 and increment indices of A and C

Trộn hai dãy tăng dần

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6	8	9	11	12	14	15	

(j) Since 17 > 15, hence copy 15 and increment indices of B and C

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
C:	2	3	5	6	8	9	11	12	14	15	17

(k) Since the B array is empty, hence remaining element of A (i.e. 17) copy to C

Figure 3.8 (a-k): Merging two sorted arrays

Trộn hai dãy tăng dần

```
Algorithm: MERGE (A, B, M, N, C)
[A and B are two sorted arrays, M is the number of elements of A, N
is the number of elements of B and C is the resultant array]
1. Set I=0, J=0, K=0
2. Repeat while I<M and J<N
      If A[I]<B[J] then
      a) Set C[K]=A[I]
      b) Set I=I+1
      c) Set K=K+1
      ELSE
      a) Set C[K]=B[J]
      b) Set J=J+1
```

Trộn hai dãy tăng dần

```
      c) Set K=K+1
      [End of If]
  [End of Loop]
3. Repeat while I<M
      a) Set C[K]=A[I]
      b) Set I=I+1
      c) Set K=K+1
  [End of Loop]
4. Repeat while J<N
      a) Set C[K]=B[J]
      b) Set J=J+1
      c) Set K=K+1
  [End of Loop]
5. Return
```

Trộn hai dãy tăng dần: O(m+n) với m, n là kích thước của 2 dãy đầu vào.

Trộn hai dãy tăng dần

Mảng trong C++

- C++ hỗ trợ cấu trúc dữ liệu mảng gồm có
  - array: kích thước cố định, không thay đổi được kích thước
  - vector: kích thước thay đổi được.
  - C++ có các hàm hỗ trợ khởi tạo giá trị/tìm kiếm (tuyến tính/nhị phân) trên mảng
- Array:
  - Không hỗ trợ chèn/xóa
  - Hỗ trợ duyệt mảng/lấy kích thước mảng.
- Vector trong C++:
  - Hỗ trợ các thao tác chèn/xóa trong thời gian tuyến tính.
    - Phương thức insert()/erase()
  - Hỗ trợ thao tác lấy kích thước/số lượng phần tử
    - size(), length()
  - Hỗ trợ các thao tác duyệt toàn bộ mảng

Tóm tắt

- Cấu trúc dữ liệu mảng
- Các thao tác chính trên cấu trúc dữ liệu mảng và độ phức tạp tương ứng
- Mảng trong C++