Bài tập 5. Sắp xếp mảng

Bài 1 (sắp xếp 0-1). Cho mảng a chứa n số nguyên 0 hoặc 1. Hãy sắp xếp mảng a theo thứ tự tăng với độ phức tạp thời gian là O(n).

Hướng dẫn

- ***Ý tưởng***: Mảng a chỉ chứa số 0 và 1. Mảng sắp xếp tăng thì số 0 đứng trước hết rồi đến số 1. Vậy đếm số số 0 trong mảng để biết có bao nhiêu số 0 trong mảng hiển thị trước, sau đó hiển thị số 1.

Ví dụ: a={0, 0,0,0,1,1,1,1,0,0}, đếm có 6 số 0. Duyệt i=0 đến (n-1), kiểm tra điều kiện i<6 thì a[i]=0 ngược lại a[i]=1. Dãy sau khi sắp xếp a={0,0,0,0,0,0,1,1,1,1}

- ***Thuật toán***:

{ Các em tự viết}

Bài 2. Cho một mảng a gồm n ký tự là các chữ cái thường. Hãy sắp xếp các ký tự của mảng a theo thứ tự giảm với độ phức tạp tính toán là O(n) và độ phức tạp bộ nhớ là O(1).

Hướng dẫn

Lưu ý: *Độ phức tạp về bộ nhớ O(1) của một thuật toán có nghĩa là lượng bộ nhớ cần thiết để thực thi thuật toán không phụ thuộc vào kích thước của dữ liệu đầu vào*.

Ý tưởng*:*

1. Đếm số lần xuất hiện của mỗi ký tự trong mảng a.
2. Tạo một vòng lặp để duyệt qua mảng a và cập nhật số lần xuất hiện của mỗi ký tự trong mảng.
3. Tạo một vòng lặp khác để duyệt qua các ký tự từ 'z' đến 'a' (từ giá trị ASCII 122 đến 97) và đặt lại các ký tự trong mảng a theo số lần xuất hiện.

Thuật toán:

{ các em tự viết}

Bài 3. Cho mảng a có n số nguyên và đã sắp tăng, mảng b có m số nguyên và đã sắp giảm. Hãy trộn hai mảng a và b vào mảng c sao mảng c được sắp tăng và độ phức tạp thời gian là O(m+n).

Hướng dẫn: Để trộn hai mảng a và b thành một mảng c sao cho mảng c là một mảng sắp tăng và độ phức tạp thời gian là O(m + n), sử dụng phương pháp trộn

Ý tưởng:

Bước 1: Khởi tạo ba chỉ số:

index_a: chỉ số cho mảng a (bắt đầu từ 0).

index_b: chỉ số cho mảng b (bắt đầu từ 0).

index_c: chỉ số cho mảng c (bắt đầu từ 0).

Bước 2: Lặp qua cả hai mảng a và b, so sánh từng phần tử của hai mảng và chèn phần tử nhỏ hơn vào mảng c. Cụ thể:

- Nếu phần tử của mảng a nhỏ hơn hoặc bằng phần tử của mảng b, chèn phần tử của mảng a vào mảng c và tăng index_a lên 1.
- Ngược lại, chèn phần tử của mảng b vào mảng c và tăng index_b lên 1.
Sau mỗi lần chèn, tăng index_c lên 1.
Bước 3: Nếu một trong hai mảng a hoặc b còn phần tử chưa được duyệt hết, chèn toàn bộ các phần tử còn lại của mảng đó vào cuối mảng c.
Thuật toán

{ các em tự viết}

Bài 4. Cho mảng a có n số nguyên a, k là một số nguyên dương nhỏ hơn n. Hãy trình bày thuật toán và cài đặt thuật toán tìm số nhỏ thứ k trong mảng a với độ phức tạp thời gian không quá O(nlogk) và độ phức tạp bộ nhớ không quá O(1).
Hướng dẫn
- ***Bước 1:*** Sắp xếp mảng a theo thứ tự tăng dần bằng một thuật toán sắp xếp hiệu quả như **QuickSort** hoặc **MergeSort**. Độ phức tạp thời gian của bước này là O(nlogn).
- ***Bước 2:*** Trả về phần tử a[k-1] từ mảng đã sắp xếp. Độ phức tạp thời gian của bước này là O(1).
Thuật toán
{ các em tự viết}

Bài tập 6. Tìm kiếm mảng

Bài 1. Tìm số x trong mảng a ở lần xuất hiện thứ k.
Hướng dẫn
Duyệt qua mảng a và kiểm tra lần lượt từng phần tử của mảng a bằng x hay không, đếm số lần xuất hiện của x trong mảng a (*dem),* kiểm tra điều kiện nếu *dem* bằng k thì trả về vị trí của số đó trong mẳng a
Thuật toán
{Các em tự viết }
Bài 2. Tìm số gần với số x nhất trong mảng a.
Hướng dẫn
Bước 1*:* Khởi tạo biến *min* với giá trị lớn nhất có thể để đảm bảo tìm được giá trị nhỏ nhất sau khi so sánh.
Bước 2: Duyệt qua mảng a, và tính khoảng cách tuyệt đối giữa mỗi phần tử và số x. So sánh khoảng cách này với giá trị của *min*, nếu nó nhỏ hơn *min*, cập nhật *min* và lưu giá trị của phần tử đó.
Bước 3: Trả về giá trị của phần tử mà có khoảng cách tuyệt đối nhỏ nhất với x.
Thuật toán { các em viết tự viết code}

Bài 3. Kiểm tra trong mảng A có hay không phần tử đa số và tìm phần tử đó nếu nó tồn tại? Một phần tử x được gọi là phần tử đa số trong A nếu tập hợp {i|A[i]=x} có nhiều hơn n/2 phần tử.

Hướng dẫn

Bước 1: Bỏ phiếu cho phần tử hiện tại: Ban đầu, chúng ta chọn phần tử đầu tiên của mảng là phần tử đa số ứng cử viên. Chúng ta gán một biến đếm (*counter*) với giá trị ban đầu là 1, đại diện cho một phiếu bầu dành cho ứng cử viên hiện tại.

Bước 2: Duyệt qua các phần tử tiếp theo: Tiếp theo, chúng ta duyệt qua các phần tử còn lại trong mảng, bắt đầu từ phần tử thứ hai.

Bước 3: Giảm/gặp bỏ phiếu cho phần tử hiện tại: Khi duyệt qua mỗi phần tử, chúng ta kiểm tra xem phần tử đó có bằng phần tử đa số hiện tại hay không. Nếu bằng, chúng ta tăng biến đếm lên 1, đại diện cho việc "gặp bỏ phiếu" cho ứng cử viên hiện tại. Nếu không bằng, chúng ta giảm biến đếm đi 1, đại diện cho việc "giảm bỏ phiếu" cho ứng cử viên hiện tại.

Bước 4: Chuyển đổi ứng cử viên: Nếu biến đếm giảm xuống 0, đó có nghĩa là số phiếu bầu dành cho ứng cử viên hiện tại đã hết. Trong trường hợp này, chúng ta chuyển sang bỏ phiếu cho ứng cử viên tiếp theo trong mảng và đặt lại biến đếm về 1.

Bước 5: Phần tử cuối cùng: Sau khi duyệt qua toàn bộ mảng, phần tử cuối cùng mà chúng ta chọn sẽ là ứng cử viên được bầu là phần tử đa số.

Bài 4: Kiểm tra trong mảng A có hay không phần tử ở giữa mảng? Số x gọi là phần tử ở giữa mảng A nếu tồn tại một chỉ số i sao cho: A[i] = x, số lượng phần tử nhỏ hơn x không vượt quá n/2, số lượng phần tử lớn hơn x không vượt quá n/2.

Hướng dẫn

Bước 1: Sắp xếp mảng

Sắp xếp mảng để dễ dàng tìm phần tử ở giữa.

Bước 2: Tìm phần tử ở giữa

Tìm phần tử ở giữa của mảng bằng cách tìm phần tử ở vị trí n/2 trong mảng đã được sắp xếp.

Bước 3: Đếm số lượng phần tử nhỏ hơn và lớn hơn phần tử ở giữa

Duyệt qua mảng và đếm số lượng phần tử nhỏ hơn và lớn hơn phần tử ở giữa. Để đếm số lượng phần tử nhỏ hơn và lớn hơn, bạn chỉ cần duyệt qua mảng một lần và kiểm tra mỗi phần tử.

Bước 4: Kiểm tra điều kiện phần tử ở giữa

Kiểm tra điều kiện để xem có phải phần tử ở giữa không. Bạn cần kiểm tra xem số lượng phần tử nhỏ hơn và lớn hơn không vượt quá n/2.

Bước 5: Trả về kết quả

Trả về kết quả sau khi kiểm tra xong.