Họ và tên: Dương Thị Bích Diệp

MSSV: 24120247

Câu 1: Cho mảng a gồm n phần tử, để tìm khóa k có trong mảng hay không đề bài yêu cầu ta xây dựng hàm linearSearch, hàm này duyệt từ phần tử đầu đến phần tử cuối danh sách, nếu có tìm thấy k thì trả về chỉ số phần tử tại đó, không có trả về -1.

Câu 2: Giống bài 1, khác là sử dụng linear search with sentinel. Trong hàm này, ta gán cho phần tử cuối cùng bằng khóa k làm lính canh (sử dụng biến tạm để lưu lại giá trị của phần tử cuối cùng), ta duyệt theo biến chỉ số i và cho i tăng đến khi gặp phần tử a[i] = k thì ngừng, nếu chỉ số i nhỏ hơn chỉ số của phần tử cuối cùng hoặc chính phần tử cuối cùng bằng k thì có k xuất hiện trong mảng.

Câu 3: Cho mảng đã sắp xếp quay m lần, yêu cầu bài toán tìm phần tử nhỏ nhất trong mảng với thời gian chạy O(logn) => nghĩ ngay tới Binary Search tìm kiếm nhị phân. Với bài toán này, ta sẽ chia mảng lần lượt 2^k lần, nghĩa là không cần duyệt toàn bộ mảng mà chỉ xét chia nửa, sẽ có left giữ đầu mảng và right cuối mảng xét. Khi xét phần tử ở giữa của không gian xét (mid), ta thấy rằng nếu phần tử đó mà nhỏ hơn phần tử đứng trước nó (mid -1), thì nó chính là MIN, hoặc phần tử ở giữa lớn hơn phần tử đứng sau nó (mid +1), thì phần tử đứng sau là MIN. Và nếu phần tử ở mid mà nhỏ hơn phần tử ở right (cuối mảng xét) thì MIN sẽ nằm ở mảng bên trái, ngược lại nằm ở mảng bên phải. Từ ý tưởng đó, ta rút ngắn được không gian xét và giảm thời gian chạy để tìm được MIN đối với mảng đã sắp xếp và xoay m lần.

Câu 4:

Bài toán tàu chở hàng, tìm sức chứa tối thiểu của tàu để có thể ship đủ các thùng chứa hàng có tải trọng khác nhau, đã được sắp xếp tăng dần trong n days.

Ta cũng áp dụng tìm kiếm nhị phân trên sức chứa của tàu. Ta biết rằng sức chứa tàu ít nhất phải chở được kiện hàng nặng nhất (left = kiện hàng nặng nhất), và trường hợp phải chở trong 1 ngày thì sức chứa của tàu phải chở được tất cả các kiện hàng (right = tổng tất cả các kiện hàng). => Ý tưởng:

- Chọn sức chứa mid = (left + right)/2, ta kiểm tra xem với sức chứa mid có thể vận chuyển trong days ngày hay không. Bằng cách cộng dồn các kiện hàng đến khi vượt qua mid, ta cần thêm 1 ngày mới. Cứ như thế đến kiện hàng cuối cùng, ta tìm được số ngày cần để vận chuyển tất cả với sức chứa mid đó.

- Nếu số ngày cần với tải trọng mid vượt quá days, nghĩa là sức chứa mid đang nhỏ, ta tăng left = mid + 1; ngược lại mid lớn, ta giảm right = mid. Đến khi left = right thì đó chính là sức chứa nhỏ nhất để vận chuyển trong days ngày.

Câu 5: Cho mảng nums chứa các phần tử nguyên dương và cho 1 số target, ta sẽ tìm xem độ dài nhỏ nhất của một dãy con liên tiếp trong nums có tổng lớn hơn bằng target là bao nhiêu. Ý tưởng : dùng kỹ thuật hai con trỏ để kiểm soát độ dài các mảng con có tổng >= target (nhanh với độ phức tạp O(n)). Ta dùng hai con trỏ này để tạo một vùng xét (bắt đầu từ left= 0, right = 0 là hai chỉ số trong mảng), ta sẽ dùng right để mở rộng vùng xét (tăng tổng dãy con) đến khi tổng >= target, khi tổng đủ lớn ta sẽ thử thu hẹp lại, dịch left sang phải để tìm xem có độ dài nhỏ hơn không. Trong quá trình tìm ta dùng biến minLength để lưu độ dài của vùng có tổng >= target, và sẽ cập nhật mỗi khi tìm được độ dài ngắn hơn. Xuất ra minLength.

Câu 6: Cho một mảng nums số nguyên đã sắp xếp và target tổng, tìm xem có hai phần tử nào trong mảng có tổng = target hay không. Ý tưởng dùng hai trỏ left và right giữ đầu mảng và cuối mảng, đặt biến sum lưu tổng nums[left] + nums[right], nếu tổng = target thì return true. Nếu lớn hơn target tức là sum đang lớn, ta giảm right--, ngược lại sum nhỏ ta tăng left++. Đến khi tìm được sum = target hoặc left > right thì ta dừng, nhưng khi left lớn hơn right rồi tức là mảng không chứa hai phần tử nào bằng target, ta return false.

Câu 7: Cũng cho mảng nums số nguyên, ta tìm xem có các bộ ba nào trong mảng có tổng bằng 0 hay không, xuất ra các bộ ba đó. Ý tưởng: Ta sẽ sắp xếp mảng tăng trước để dễ dàng trong việc tìm kiếm hơn. Vẫn là dùng kỹ thuật hai con trỏ, cho vòng lặp chạy từ i = 0, left giữ phần tử tiếp theo sau phần tử thứ i, right giữ phần tử cuối mảng.

Trong vòng lặp while(left < right), cho tổng sum = nums[i] + nums[left] + nums[right].

- Nếu sum = 0, ta xuất ra bộ ba nums[i], nums[left], nums[right]. Ta biết rằng đối với một số nguyên, sẽ có các cặp số khác nhau không trùng lặp số nào với nhau để tổng số nguyên đó với các cặp số bằng 0, tức là với a[i] = 1, a[left] = 0, thì a[right] chỉ có thể bằng -1, và ngược lại. Do đó ta sẽ giảm right đồng thời tăng cả left để xét các phần tử mới.

- Nếu sum < 0, tổng nhỏ nên ta tăng left++.

- Nếu sum > 0, tổng lớn nên ta giảm right--.

Cứ như thế cho đến khi điều kiện left < right không còn đúng nữa thì ta đã tìm được các bộ ba cho tổng bằng 0, hoặc trong mảng không có bộ ba nào.