Báo cáo Project – Tuần 2

1. Exercise 1: Linear Search

- Ý tưởng: Tìm kiếm trực tiếp: Duyệt từng số hạng của dãy cho tới khi tìm được số hạng cần tìm.

- Giải thích code: Cho 1 biến i là chỉ số đầu tiên của dãy. Duyệt i từ 0 tới n – 1.

+ Khi tìm được phần tử cần tìm, ngày lập tức trả về chỉ số của phần tử.

+ Khi không tìm được phần tử nào, return -1.

2. Exercise 2: Linear Search with Sentinel

- Ý tưởng: Cũng giống như Linear Search, nhưng sử dụng phần tử cuối làm "lính canh" bằng cách gán key bằng phần tử cuối, sau đó duyệt từng số hạng của dãy như Ex 1. Nếu như không tìm được phần tử key cho đến cuối dãy, ta so sánh xem key có bằng phần tử cuối không. Nếu có, trả về phàn tử cuối, ngược lại trả về -1.

- Giải thích code: Gán key vào phần tử cuối, lưu phần tử cuối vào 1 biến tạm, sau đó thực hiện Linear Search.

+ Nếu như tìm thấy phần tử key trong dãy, hoặc không tìm thấy nhưng phần tử cuối == key, trả về chỉ số i.

+ Nếu như không tìm thấy phần tử key, phần tử cuối != key, trả về -1.

3. Exercise 3:

- Ý tưởng: Cho 2 chỉ số left và right đặt ở đầu và cuối dãy. Đặt mid = (left + right) / 2 là phần tử ở giữa dãy (chính giữa left và right).

So sánh a[mid] với a[right]:

+ Nếu như a[right] < a[mid] thì phần tử nhỏ nhất có khả năng nằm ở bên phải mid. Vì thế, đặt left = mid + 1 để dịch mid về phía phải.

+ Nếu ngược lại, phần tử nhỏ nhất có khả năng nằm về phía bên trái mid hoặc là mid. Đặt right = mid để dịch mid dần về phía trái.

Tiếp tục đặt mid = (left + right) và làm lại các bước trên cho đến khi left >= right. Lúc này không còn phần tử nào nằm giữa left và right, khi đó a[right] là kết quả.

- Giải thích code: Đặt các chỉ số left, right.

Khi left còn nhỏ hơn right, ta đặt mid = (left + right) / 2.

So sánh a[mid] với a[right] để xét xem min của mảng nằm ở đâu.

Nếu a[right] < a[mid] -> nằm bên phải, dịch mid qua phải.

Nếu a[right] >= a[mid] -> nằm bên trái hoặc chính nó, dịch mid qua trái.

Tiếp tục cho đến khi kết thúc vòng lặp, trả về a[right].

4. Exercise 4:

- Ý tưởng:

+ Sức chứa của thuyền phải nằm trong phạm vi từ giá trị lớn nhất của mảng tới tổng của tất cả các phần tử trong mảng, vì tổng của các phần tử trong mảng bao hàm hết tất cả các cách xếp các món hàng vào 1 ngày, còn GTLN của mảng lớn hơn các giá trị còn lại, bao hàm mỗi món hàng xếp vào 1 ngày riêng lẻ.

+ Gọi GTNN có thể đạt được của kết quả là GTLN của mảng, còn GTLN có thể có của kết quả là tổng các phần tử của mảng.

Ta đặt mid = (GTLN + GTNN) / 2 làm giá trị tạm thời, xét xem mid có thoả mãn bài toán hay chưa:

Nếu mid thoả yêu cầu, xét xem còn số nào nhỏ hơn mid thoả yêu cầu bài toán không, bằng cách đặt GTLN = mid, rồi thực hiện tiếp vòng lặp mid = (GTLN + GTNN) / 2.

Nếu mid chưa thoả yêu cầu -> tăng mid lên để chở hết hàng rồi thực hiện tiếp vòng lặp.

Cách kiểm tra mid: Xét xem cần tối thiểu bao nhiêu ngày để có thể chở được hết mid. Nếu số ngày này > số ngày mà người dùng nhập vào, tức là mid chưa thoả mãn -> giảm mid xuống. Ngược lại, tăng mid lên.

- Giải thích code:

+ Gọi 2 biến least, largest là GTNN và GTLN mà kết quả có thể có, trong đó GTNN là max của mảng, GTLN là tổng các phần tử trong mảng.

+ Dùng 1 biến mid = (GTLN + GTNN) / 2

+ Chạy vòng lặp while(GTNN < GTLN), xét xem cần bao nhiêu ngày để chở hết số món hàng, với mỗi ngày chớ không vượt quá mid.

+ Nếu như nhỏ hơn số ngày mà người dùng nhập, giảm mid xuống bằng cách đưa GTLN = mid, sau đó mid = (GTLN + GTNN) / 2.

+Nếu như ngược lại, tăng mid lên bằng cách đưa GTNN = mid + 1, sau đó mid = (GTLN + GTNN) / 2.

Tiếp tục vòng lặp như thế, cho đến khi GTNN > GTLN. Kết quả là GTNN.

5. Exercise 5:

- Ý tưởng: Bắt đầu từ chỉ số i = 0 -> n.

Tạo 1 biến sum để tính tổng và 1 biến đếm để đếm số phần tử lấy để tính tổng. Cho 1 chỉ số j chạy từ i -> n, tính tổng các phần tử a[j] cho đến khi tổng này >= target thì dừng lại và đếm số phần tử đã đem đi tính tổng, sau đó i tăng lên 1 và lặp lại các bước trên. Lấy GTNN của số phần tử trong mỗi trường hợp.

- Giải thích code:

+ Tạo 1 biến min để lưu GTNN và 1 biến i để chạy từ 0 -> n.

+ Tạo biến sum và cnt.

+ Chạy 1 biến j từ i -> n để tính tổng các phần tử từ i cho tới khi >= target thì dừng lại. Đó là 1 dãy thoả mãn >= target. Đếm số phần tử trong dãy trong trường hợp này.

+ So sánh với min. Nếu nhỏ hơn min, đây hiện tại là GTNN.

+ Tiếp tục vòng lặp cho đến khi tìm được GTNN.

6. Exercise 6:

- Ý tưởng: Khởi tạo 2 con trỏ left và right nằm ở biên trái và phải của mảng.

Khi left < right, ta xét:

+ Nếu a[left] + a[right] == target -> kết thúc ngay vòng lặp và trả về true.

+ Nếu a[left] + a[right] > target: vì mảng sắp xếp tăng dần nên tổng này lớn hơn nghĩa là giá trị right đang lớn. Ta cần đẩy right lùi về phía trái để tổng này nhỏ lại, tiếp tục xét tổng này với target.

+ Nếu a[left] + a[right] < target: tổng này đang nhỏ hơn, nghĩa là giá trị left đang bị nhỏ. Ta cần đẩy left tiến lên bên phải để tổng này lớn hơn.

Tiếp tục vòng lặp như vậy cho đến khi left == right. Nếu vẫn không tìm được left, right thoả mãn, trả về false.

- Giải thích code: Giống như ý tưởng.

7. Exercise 7:

- Ý tưởng: Đầu tiên, ta cần sắp xếp lại mảng theo thứ tự tăng dần.

Từ biểu thức a[i] + a[j] + a[k] = 0 => a[j] + a[k] = -a[i].

Ta tìm tất cả các bộ j, k sao cho a[j] + a[k] = -a[i]. Cách tìm này tương tự như Exercise 6.

- Giải thích code:

+ Sắp xếp lại mảng theo thứ tự tăng dần: Dùng Heap Sort.

+ Cho 1 biến i chạy từ 0 -> n – 3. Với mỗi a[i], khai báo 1 biến left = i + 1, right = n – 1. Khi left < right, tìm bộ l, r sao cho a[l] + a[r] == -a[i], giống với Exercise 6. Khi tìm được, in hết các phần tử a[i], a[l], a[k].