BÁO CÁO THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Ngày 11/03/2025

Họ và tên : Bùi Huỳnh Tây

MSSV : 24521589

NỘI DUNG BÁO CÁO

Bài 1: VQ44_FLOWERS

Xây dựng một mảng mới gồm các kí tự khác nhau ở đầu mảng, còn các kí tự còn thừa đẩy ra sau.

Vậy nên ta sử dụng cấu trúc dữ liệu Map để xem xét một số có phải là số đầu tiên xuất hiện trong dãy hay không và thêm nó vào một mảng mới. Và các kí tự thừa còn lại sẽ cho vào sau mảng mới đó. In *k* kí tự đầu tiên của mảng mới.

Độ phức tạp O(n * log(n))

Bài 2: Point2D

Với mỗi x, ta xây dựng một vector chứa các số y_i để thể hiện cho điểm x, y_i.

Ta sử dụng cấu trúc dữ liệu map<int, vector<int>>, sau đó duyệt map thì x sẽ tự động được sắp xếp tăng dần và với mỗi x ta sắp xếp vector chứa các điểm y_i của x đó giảm dần.

Độ phước tạp O(n * log(n))

Bài 3: VS14_Gifts

Với mỗi quà a[i], ta cần tìm số a[j] lớn nhất mà bé hơn hoặc bằng k – a[i]. Ta sử dụng thuật toán chặt nhị phân để tìm số này, và lưu ý một món quà chỉ được chọn một lần nên nếu j == i, thì j phải giảm xuống một đơn vị.

Độ phức tạp O(n * log(n))

Bài 4: PasswordStrength

Ta kiểm tra điều kiện: *"tồn tại các kí tự không hợp lệ", "toàn bộ kí tự là viết thường", "toàn bộ kí tự là số", "độ dài nhỏ hơn 8".*

Ta đếm các số lượng : *"Hiệu giữa độ dài mật khẩu và 8", "Số lượng chữ cái in hoa trong mật khẩu", "Số lượng chữ số trong mật khẩu", "Số lượng kí tự đặc biệt trong mật khẩu".*

Sau đó dựa vào bảng số liệu đề cho và tính theo công thức:

score = BaseScore +

(Num_Excess x Bonus_Excess) +

(Num_Upper x Bonus_Upper) +

(Num_Numbers x Bonus_Numbers) +

(Num_Symbols x Bonus_Symbols) +

Bonus_Combo +

Bonus_FlatLower +

Bonus_FlatNumber

Độ mạnh của mật khẩu sẽ tùy vào số điểm tính được.

Bài 5: CaesarCipher

Với mỗi kí tự nhận được, nếu nó là kí tự từ 'A' đến 'Z' thì phải thay đổi theo công thức:

$$\mathbf{E(x)=(x+k) \bmod 26}$$

Sau đó in ra kết quả sau khi sửa.

Độ phức tạp O(n)

Bài 6: ReversingEncryption

Để giải mã tìm được chuỗi ký tự ban đầu (plaintext) thì ta đi ngược lại thuật toán mã hóa.

Chạy i từ 1 đến n, sau đó kiểm tra i có là ước của n hay không. Nếu có thì đảo ngược chuỗi đang xét lại từ đoạn 1 đến i.

Độ phực tạp O(n^2)

Bài 7: Messages

Để tìm được độ dài xâu S ngắn nhất, thì phải tìm phần giao Sb và Se lớn nhất có thể.

Chạy i từ min(Sb, Se) về 1. Sau đó kiểm tra Sb và Se có chung với nhau độ dài i hay không. Nếu có dừng lại và in ra đáp án.

Độ phức tạp O(n ^ 2).

Bài 8: Binary Search 1

Với mỗi lần while chặt nhị phân lặp lại một lần thì ta cộng biến điếm count lên để đếm **số lần duyệt qua các phần tử.**

Độ phức tạp O(n * log(n))

Bài 9: Binary Search 2

Ý tưởng thuật toán giống như bài 8, nhưng thay vì vector chứa các int thì thay bằng string.

Bài 10: Linear Search 1

Duyệt i từ 0 đến n-1, nếu gặp số x phải tìm thì in ra vị trí i và số lần duyệt qua các phần từ là i + 1 thì dừng lại.

Tương tự duyệt i từ n-1 về 0, nếu gặp số x thì in ra i và số lần duyệt là n-1-i.

Độ phức tạp O(n)

Bài 11: Linear Search 2

Duyệt i từ 0 đến n-1, nếu gặp số x phải tìm thì in ra vị trí i và số lần duyệt qua các phần từ là i + 1.

Độ phức tạp O(n)

Bài 12: Linear Search 3

Dùng mảng đánh dấu (exist[]) để theo dõi các số đã xuất hiện trong tập hợp A[0] đến A[i]

Biến mex giữ giá trị nhỏ nhất chưa xuất hiện.

Duyệt mảng:

- Nếu A[i] nhỏ hơn độ dài mảng, đánh dấu là đã xuất hiện.
- Cập nhật mex nếu giá trị mex hiện tại đã xuất hiện trong tập.

Lưu mex[i] vào mảng kết quả.

Độ phức tạp

- Mỗi phần tử của A được duyệt 1 lần → O(n)
- mex chỉ tăng dần nên kiểm tra cũng O(n) tổng cộng.
- Tổng độ phức tạp: O(n)

Bài 13: Linear Search 5

Nếu n = 2: kết quả là max – min.

Nếu n > 2:

Để tối ưu tổng trọng số, ta cần tập trung vào sự khác biệt lớn nhất giữa các phần tử.

Trọng số của tập S là max(s) – min(s), nên tổng trọng số tốt nhất sẽ là: (max1−min1) + (max2−min2).

Do đó, ta chỉ cần:

1. Sắp xếp mảng.
2. Chọn 2 phần tử lớn nhất và 2 phần tử nhỏ nhất.

3. Tính tổng trọng số tối đa.

Độ phức tạp O(n * log(n))

Bài 14: VW05p_Enrichement

Vì n * m <= 10^6, nên với mỗi a_ij ta xem ô này là ô trên cùng và trái cùng của một ô 3x3 và tính tổng ô 3x3 đó, duyệt hết a_ij để tìm ô có tổng bé nhất.

Độ phức tạp O(9 * n * m)