Bài 1: VQ44_FLOWERS

Tóm tắt: Bài toán yêu cầu chọn k màu khác nhau từ danh sách bóng đèn sao cho số lượng màu khác nhau là lớn nhất.

Hướng giải quyết:

Chia bài toán ra hai trường hợp:

- Nếu số lượng màu khác nhau lớn hơn hoặc bằng k, chọn ngẫu nhiên k màu từ tập hợp này.
- Nếu số lượng màu khác nhau ít hơn k, lấy tất cả màu khác nhau rồi bổ sung thêm màu sao cho đủ k.

Bài 2: Point2D

Tóm tắt: Bài toán yêu cầu sắp xếp N điểm (x, y) trên mặt phẳng Oxy theo quy tắc tăng dần theo hoành độ x, nếu x bằng nhau thì sắp xếp giảm dần theo tung độ y.

Hướng giải: Sử dụng hàm $sort$ của thư viện $algorithm$ và truyền key so sánh vào hàm $sort$ theo quy tắc tăng dần theo x, nếu x bằng nhau thì giảm dần theo y.

Bài 3: VS14_Gifts

Tóm tắt: Bài toán cho n món hàng khác nhau và yêu cầu chọn ra 2 món quà khác nhau sao cho tổng giá trị lớn nhất nhưng không được vượt quá x.

Hướng giải quyết:

- Sắp xếp n món hàng theo giá trị tăng dần.
- Với mỗi món quà i có giá trị là a_i, sử dụng tìm kiếm nhị phân để tìm món quà thứ j $(i < j)$ với giá trị a_j lớn nhất nhưng không được vượt quá $(x - a_i)$.

Bài 4: PasswordStrength

Tóm tắt: Viết chương trình xác định độ mạnh của mật khẩu dựa trên công thức cho trước với các mức $\{KhongHopLe, Yeu, Vua, Manh, RatManh\}$.

Hướng giải quyết:

- Kiểm tra tính hợp lệ của mật khẩu: Độ dài bé hơn 8 hoặc chứa ký tự không hợp lệ thì trả về kết quả $KhongHopLe$
- Tính điểm $score$ cho mật khẩu: Duyệt qua mật khẩu và tổng điểm các tiêu chí theo quy tắc đã cho.
- Phân loại mật khẩu dựa trên $score$.

Bài 5: CeasarCipher

Tóm tắt: Viết chương trình nhận một key là **K** với một chuỗi kí tự viết hoa (plaintext). Yêu cầu trả về chuỗi kết quả (cipher text) sau khi áp dụng mã hóa Ceasar với **K**.

Hướng giải quyết:

- Duyệt qua từng vị trí trong chuỗi plaintext. Với mỗi vị trí I, thực hiện biến đổi theo công thức: $s_i = char\big((s_i - \text{'A'} + \text{K})\%26 + \text{'A'}\big)$.
- Do chuỗi plaintext ban đầu có thể chứa khoảng trắng nên phải dùng lệnh $getline(cin, s)$

Bài 6: ReversingEncryption

Tóm tắt:

- Cho một chuỗi t là kết quả đã được mã hóa (cipher text) của chuỗi s thông qua thuật toán:
  - Duyệt qua các ước số **d** theo thứ tự giảm dần của số **n** cho trước, với mỗi **d** thì đảo ngược chuỗi con **s[1…d].**
- Yêu cầu giải mã chuỗi **t** để tìm lại chuỗi **s** ban đầu (plaintext).

Hướng giải quyết: Để giải quyết bài toán này, ta sẽ thực hiện ngược lại thuật toán mã hóa:

- Duyệt qua các ước số **d** của **n** theo thứ tự tăng dần.
- Với mỗi ước số **d,** ta thực hiện đảo ngược chuỗi con **s[1…d]**.
- Sau khi thực hiện xong cho tất cả các **d**, kết quả cuối cùng sẽ là chuỗi **s** ban đầu.

Bài 7: Messages

Tóm tắt:

- Cho hai xâu **Sb** và **Se**. Xây dựng xâu **S** thỏa mãn các điều kiện:
  - Xâu **Sb** là phần đầu của xâu **S**
  - Xâu **Se** là phần cuối của xâu **S**
  - Xâu **S** phải ngắn nhất có thể, nghĩa là các phần trùng lặp giữa **Sb** và **Se** cần được tối ưu hóa để tránh lặp lại quá nhiều ký tự.

Hướng giải quyết:

- Ta cần phải tìm một xâu **substr** có độ dài lớn nhất vừa là hậu tố của **Sb**, vừa tiền tố của **Se**.
- Duyệt lần lượt qua các hậu tố của xâu **Sb** có độ dài giảm dần và kiểm tra xâu đó có phải là tiền tố của xâu **Se**. Giả sử hậu tố của **Sb** thỏa mãn điều kiện trên là **Sb[i…len(Sb)-1]** thì xâu kết quả **S** cần tìm là: **Sb[0…i-1] + Se**.

Nhận thấy việc so sánh hậu tố khớp với tiền tố, ta có thể sử dụng thuật toán KMP để cải tiến cho thuật toán:

- Tạo xâu **T = Se + "#" + Sb** và tính mảng **π** là hàm tiền tố của xâu **T**, trong đó $\pi[i]$ là là độ dài của tiền tố chuẩn dài nhất của xâu con $T[0 \dots \text{i}]$ mà cũng là hậu tố của xâu con này.
- Khi đó, kết quả của xâu $\boldsymbol{S} = Sb + Se[\pi[len(T) - 1] \dots len(Se) - 1]$

Bài 8: Binary Search 1

Tóm tắt: Cho một mảng A gồm N số nguyên đã được sắp xếp tăng dần và không trùng nhau. Cho một số x cần tìm trên mảng A. Yêu cầu thực hiện tìm kiếm nhị phân trả về vị trí của x hoặc -1 nếu không tìm thấy và số lần duyệt của thuật toán.

Hướng giải quyết:

- Với bài toán này, ta chỉ cần đơn giản thực hiện lại thuật toán tìm kiếm nhị phân và số lần truy cập phần tử $A[mid]$ cũng chính là số lần duyệt của thuật toán.
- Trả về vị trí của x nếu có hoặc -1 nếu không tìm thấy và số lần duyệt của thuật toán.

Bài 9: Binary Search 2

Tóm tắt: Cho một mảng A gồm N chuỗi đã được sắp xếp theo thứ tự bảng chữ cái. Cho một chuỗi x cần tìm trên mảng A. Yêu cầu thực hiện tìm kiếm nhị phân trả về vị trí của x hoặc -1 nếu không tìm thấy và số lần duyệt của thuật toán.

Hướng giải quyết:

- Về ý tưởng cơ bản của bài toán này cũng chỉ là sử dụng phương pháp tìm kiếm nhị phân giống bài 8. Nhưng điểm khác biệt ở đây chính là đối tượng so sánh là chuỗi nên ta cần phải định nghĩa các phép toán so sánh của xâu theo thứ tự từ điển.
- Nhắc lại về so sánh theo thứ tự từ điển, một xâu **a** nhỏ hơn xâu **b** trong các trường hợp sau:
    - **a** là tiền tố của **b**
    - Tồn tại vị trí i sao cho **a[0…i-1] = b[0…i-1]** và **a[i] < b[i]**

Bài 10: Linear Search 1

Tóm tắt: Cho mảng A gồm N số nguyên dương và số nguyên x cần tìm. Yêu cầu cài đặt thuật toán tìm kiếm tuyến tính để tìm vị trí của x đầu tiên trong mảng có N phần tử và đếm số lần duyệt qua các phần tử.

Hướng giải quyết:

- Duyệt tuần tự qua các phần tử từ đầu đến cuối mảng A, nếu gặp phần từ x cần tìm thì trả về vị trí và số lần duyệt.
- Nếu ở lượt đầu không tìm thấy x thì trả về -1.
- Nếu tìm thấy x ở lượt đầu, thực hiện lại thuật toán trên nhưng theo thứ tự từ cuối về đầu mảng A.

Bài 11: Linear Search 2

Tóm tắt: Cho mảng A gồm N số nguyên dương và số nguyên x cần tìm. Yêu cầu cài đặt thuật toán tìm kiếm tuyến tính để tìm vị trí của x đầu tiên trong mảng có N phần tử và đếm số lần duyệt qua các phần tử.

Hướng giải quyết:

- Duyệt tuần tự qua các phần tử từ đầu đến cuối mảng A, lưu lại các vị trí x tìm được và só phép duyệt để tìm được vị trí x đó.

Bài 12: Linear Search 3

Tóm tắt: Cho một mảng A có N phần tử. Yêu cầu tìm MEX_i $(0 \leq i < n)$, với MEX_i là số nguyên nhỏ nhất lớn hơn hoặc bằng 0 mà không xuất hiện trong từ đầu mảng cho đến phần từ thứ i.

Hướng giải quyết:

- Tạo một set chứa (n + 1) phần tử có giá trị từ 0 đến n.
- Duyệt qua lần lượt các vị trí i của mảng A, nếu phần tử A[i] tồn tại trong set ta sẽ xóa khỏi set và MEX_i là phần tử có giá trị nhỏ nhất trong set.

Bài 13: Linear Search 5

Tóm tắt:

- Cho mảng A gồm N phần tử. Chia dãy A thành 2 dãy con S1 và S2 (có thể rỗng) sao cho:
- Mỗi phần tử của mảng A chỉ thuộc duy nhất một trong hai dãy con S1 hoặc S2
- Tổng trọng số của S1 và S2 là lớn nhất.
- Trọng số của một mảng được định nghĩa là hiệu giữa phần từ lớn nhất và phần tử nhỏ nhất trong mảng.

Hướng giải quyết:

- Sắp xếp mảng A tăng dần.
- Cách chia tối ưu là xếp A[0] và A[N-1] vào S1, A[1] và A[N-2] vào S2.
- Khi đó, kết quả bài toán là (A[N-1] + A[N-2] – A[0] – A[1])

Bài 14: VW05p_Enrichement

Tóm tắt: Cho một ma trận A có kích thước N.M ô gồm các số nguyên. Tìm một ma trận con 3x3 sao cho tổng các phần tử của ma trận con đó là nhỏ nhất.

Hướng giải quyết:

- Ta xây dựng mảng tổng tiền tố của ma trận A. Sau đó, duyệt qua từng ma trận con 3x3 và tính nhanh tổng của từng ma trận con bằng mảng tổng tiền tố và lưu lại ma trận có tổng nhỏ nhất.

Source code tham khảo: https://github.com/ngocthinh09/IT003---Data-Structure-Algorithms-/tree/main/04-03-2025