# Lời giải [IT003.P21.CTTN.1] Assigment1 (Basic)

Bảo Quý Định Tân - 24520028

Ngày 4 tháng 3 năm 2025

## Mục lục

# 1 VQ44-FLOWERS

Ở bài toán này ta cần in ra các màu sao cho số lượng màu khác nhau là nhiều nhất có thể. Có nhiều cách để tiếp cận như ta có thể sử dụng **map** trong **c++** để có thể lưu trữ lại số lượng của từng màu.

Ta bắt đầu bằng việc tham lam in ra nhiều màu khác nhau nhiều nhất có thể, nếu vẫn còn thừa, ta lấy ra bất kỳ màu còn sót lại chưa được lấy hết.

Code mẫu: https://ideone.com/r5M0hP

# 2 Point2D

Ở bài toán này, ta có thể lưu trữ mảng ở dạng **pair** cho thuận tiện trong việc sắp xếp và làm theo như đề.

Code mẫu: https://ideone.com/tTI2X5

# 3 VS14-Gifts

Ở bài toán này, ta sẽ thử lấy từng số $a_i$ là món quà đầu tiên, rồi sau đó sẽ tìm cách để tìm số $a_j$ $(j \neq i)$ thứ hai sao cho giá trị là lớn nhất trong khả năng cho phép.

Để thực hiện việc này nhanh, ta có thể sắp xếp mảng lại, với mỗi lần thử lấy $a_i$ là món quà đầu tiên, ta sẽ chặt nhị phân để tìm $a_j$ lớn nhất sao cho $a_i + a_j \leq x$.

Code mẫu: https://ideone.com/D6QyRb

# 4 PasswordStrength

Ở bài này, ta chỉ thực hiện như đề bảo, quan trọng là cài đặt như nào cho thuận tiện với tránh sai vặt.

Ta sẽ kiểm tra chuỗi nhập vào có phải chuỗi hợp lệ hay không trước tiên rồi mới thực hiện tính điểm.

Code mẫu: https://ideone.com/6xXCxt

# 5 CaesarCipher

Ở bài này, khi cài đặt ta chú ý tránh để khỏi *mod* sai, như lấy $k \ mod \ 26$ trước khi thực hiện mã hoá.

Code mẫu: https://ideone.com/0rplMs

# 6 ReversingEncryption

Ở bài này ta cũng cài đặt như đề bảo, có thể sử dụng hàm **reverse()** trong **c++** để cài đặt tiện hơn.

Code mẫu: https://ideone.com/8Lhx1O

# 7 Messages

Có nhiều cách để thực hiện bài này như duyệt trâu $O(n^2)$ vì giới hạn dữ liệu khá nhỏ.

Ở đây mình sử dụng **kmp** để tối ưu hơn chỉ trong $O(n)$.

Code mẫu: https://ideone.com/JHq0tF

# 8 Binary Search 1

Ở bài này ta chỉ đơn giản thực hiện y như đề bảo, và thêm một biến đếm trong lúc chặt nhị phân.

Code mẫu: https://ideone.com/K8lJBu

# 9 Binary Search 2

Ở bài này, cách cài đặt cũng tương tự bài ở trên, chỉ thay đổi kiểu dữ liệu số thành chuỗi thôi.

Code mẫu: https://ideone.com/qZ4rYp

# 10 Linear Search 1

Ở bài này, ta chỉ đơn giản thực hiện y chang đề bảo.

Để ý thêm là số lần duyệt có thể tính nhanh bằng vị trí của số.

Code mẫu: https://ideone.com/INIZ9m

## 11 Linear Search 2

Ở bài này, ta chỉ đơn giản thực hiện y chang đề bảo.

Để xuất đáp án, ta có thể lưu mỗi vị trí của số là đủ, vì có vị trí, ta có thể dễ dàng suy ra số lần cần duyệt để tìm.

Code mẫu: https://ideone.com/Yr1ZvX

## 12 Linear Search 3

Ở bài toán này, ta cũng chỉ đơn giản thực hiện y chang đề bảo.

Để đánh dấu số nào đã xuất hiện, ta có thể sử dụng **map**. Còn để tìm **MEX**, ta chỉ đơn giản là duyệt trâu, nếu số **MEX** hiện tại không còn là **MEX** nữa, ta sẽ duyệt tuần tự để tìm số **MEX** tiếp theo.

Độ phức tạp là $O(n)$ bởi vì có nhiều nhất cũng chỉ $n$ lần thay đổi **MEX**.

Code mẫu: https://ideone.com/td8jOC

## 13 Linear Search 5

Ở bài này, ta có nhận xét tham lam rằng, để chia mảng ra làm hai tập sao cho khoảng cách giữa số lớn nhất và số nhỏ nhất là nhiều nhất có thể. Có một cách chia là ta bỏ số lớn nhất cùng số nhỏ nhất vào một nhóm, và số lớn nhì cùng số nhỏ nhì vào nhóm còn lại.

Ở các trường hợp đặc biệt như $n = 2$ hoặc $n = 3$ ta sẽ chia số lớn nhất cùng số nhỏ nhất vào một nhóm, các số còn lại vào nhóm còn lại. Vì thế sẽ có nhóm mà giá trị là 0.

Code mẫu: https://ideone.com/P3Ba83

## 14 VW05p-Enrichement

Ở bài này, ta chỉ đơn giản thực hiện y như đề bảo.

Tìm ô vuông có kích thước $3 \times 3$ sao cho tổng nhỏ nhất.

Code mẫu: https://ideone.com/bmEq3w