# BÁO CÁO

**Bùi Huỳnh Tây**
**MSSV: 24521589**

25/03/2025

—

Cấu trúc dữ liệu và giải thuật -
IT003.P21.CTTN

—

GV HDTH: Trần Đình Khang

# NỘI DUNG BÁO CÁO

## A. Assigment2 (Advance)

### 1. MaxMinSum

Bài toán yêu cầu tính tổng hiệu số giữa giá trị lớn nhất (Alice chọn) và giá trị nhỏ nhất (Bob chọn) trong tất cả các cách chọn K phần thưởng từ N phần thưởng, kết quả lấy modulo 1e9+7.

**Hướng giải quyết:**

1. Sắp xếp mảng a theo thứ tự tăng dần để dễ xác định min và max trong các tập con.
2. Tính trước mảng giai thừa fact và nghịch đảo để tính tổ hợp nhanh:
    - fact[i] = i! mod 1e9+7
    - Hàm inv(x) = x^(mod-2) mod 1e9+7 (theo Fermat nhỏ)
    - Hàm comb(x,y) = C(x,y) = fact[x] * inv(fact[y]*fact[x-y]) mod 1e9+7
3. Tính tổng đóng góp của từng phần thưởng:
    - Khi là max: Với mỗi giá trị val có tần số f, tính tổng val * C(cnt, k-i) * C(f,i) với i từ 1 đến min(f,k)
      (cnt là số phần tử < val)
    - Khi là min: Tương tự nhưng xét các phần tử > val và trừ đi kết quả
4. Kết quả cuối cùng là tổng hiệu số sau khi mod.

**Độ phức tạp:** O(NlogN)

### 2. khangtd.Login1

Bài toán yêu cầu tra cứu mật khẩu tương ứng với tài khoản nhập vào. Nếu tài khoản chưa đăng ký thì in thông báo "Chua Dang Ky!". Mỗi tài khoản chỉ lưu mật khẩu gần nhất nếu có nhiều lần cập nhật.

**Hướng giải quyết:**

- Sử dụng unordered_map để lưu trữ cặp (tài khoản, mật khẩu) với key là tài khoản và value là mật khẩu.

- Đọc và lưu N cặp (s, t) vào unordered_map. Nếu cùng tài khoản xuất hiện nhiều lần, giá trị sau sẽ ghi đè lên giá trị trước (đảm bảo lưu mật khẩu gần nhất).

### 3. khangtd.Login2

- Sử dụng unordered_map<string, vector<string>> để lưu trữ danh sách mật khẩu theo thứ tự thời gian cho mỗi tài khoản.
- Đọc và lưu N cặp (s, t):
  - Với mỗi tài khoản s, thêm mật khẩu t vào vector tương ứng trong map.
- Xử lý Q truy vấn:
  - Với mỗi tài khoản x, kiểm tra xem có tồn tại trong map không.
  - Nếu có, in toàn bộ mật khẩu trong vector (cách nhau bằng dấu cách).
  - Nếu không, in "Chua Dang Ky!".

### 4. khangtd.DetectVirus2 & khangtd.DetectVirus

- Sử dụng thuật toán KMP để tìm kiếm xâu con T trong xâu S một cách tối ưu.
- Xây dựng mảng p (prefix function) cho xâu kết hợp "T#S" để xác định các vị trí khớp.
- Duyệt qua mảng p để tìm các vị trí mà độ dài tiền tố bằng độ dài xâu T, đó chính là vị trí xuất hiện của T trong S.

### 5. Linear Search 4

Bài toán yêu cầu chia N học sinh thành hai đội sao cho mỗi đội có tính duy nhất (số môn học riêng biệt) chính xác bằng K. Cần kiểm tra tính khả thi của việc chia đội này.

**Hướng giải quyết:**

➢ Đếm tần suất môn học:
  - Dùng mảng freq để đếm số lần xuất hiện của mỗi môn học.
  - single: số môn học xuất hiện đúng 1 lần.
  - multiple: số môn học xuất hiện từ 2 lần trở lên.

➢ Kiểm tra điều kiện chia đội:
  - Điều kiện cần: single + multiple >= 2*K (đủ môn học để chia đều).

- Điều kiện đủ: single / 2 + multiple >= K (có thể phân bổ môn học cho cả hai đội).
- Nếu thỏa mãn cả hai điều kiện → "YES", ngược lại → "NO".

### 6. Vượt mức Pickleball v2 (time limit: 0.5s)

Bài toán yêu cầu đếm số lần một sinh viên vượt mức Pickleball, tức là khi điểm đánh bóng hiện tại ≥ 2 lần trung vị của d lần đánh trước đó.

**Hướng giải quyết:**

- Sử dụng mảng cnt để đếm tần suất xuất hiện của các điểm số từ 0 đến 200.
- Với mỗi lần đánh thứ i (i > d):
  1. Tính trung vị của d lần đánh trước đó (từ i-d đến i-1) dựa trên mảng cnt.
  2. Nếu d lẻ: trung vị là phần tử ở vị trí (d/2 + 1), nhân đôi giá trị này.
  3. Nếu d chẵn: trung vị là trung bình của hai phần tử ở giữa, tính tổng hai giá trị này.
  4. Kiểm tra nếu điểm hiện tại a[i] ≥ trung vị → tăng biến đếm res.
  5. Cập nhật mảng cnt bằng cách thêm điểm hiện tại và loại bỏ điểm cũ (cách đây d lần).

**Độ phức tạp:** O(n × 200)

### 7. Bốn ông điên

Bài toán yêu cầu tìm số lần hoán đổi tối thiểu để sắp xếp một mảng số nguyên thành mảng có thứ tự (tăng dần hoặc giảm dần). Đây là bài toán tối ưu hóa liên quan đến chu trình hoán vị.

**Hướng giải quyết:**

- Tạo một mảng a gồm các cặp (giá trị, vị trí ban đầu).
- Sắp xếp mảng a theo thứ tự tăng dần và giảm dần để xác định vị trí đúng của từng phần tử trong hai trường hợp.
- Với mỗi phần tử, nếu vị trí hiện tại khác vị trí đúng, thêm cạnh nối giữa hai vị trí này vào đồ thị.
- Đếm số chu trình trong đồ thị:
  - Sử dụng DFS để đếm số thành phần liên thông (chu trình) trong đồ thị.

- Số lần hoán đổi tối thiểu = tổng số phần tử - số thành phần liên thông.

- So sánh số lần hoán đổi trong hai trường hợp (tăng dần và giảm dần) để chọn giá trị nhỏ nhất

### 8. Huấn luyện chuột

**Hướng giải quyết:**

- Số dãy thỏa mãn là số cách chọn N phần tử từ 2N-1 phần tử, tương đương với tổ hợp C(2N-1, N-1).
- Công thức này có thể tính bằng định lý Lucas khi N lớn (đến 10^12).
  - Tiền tính giai thừa và nghịch đảo giai thừa modulo p để hỗ trợ tính tổ hợp nhanh.
  - Sử dụng định lý Lucas để chia nhỏ bài toán khi N ≥ p.

## B. Assignment 2 (Basic)

### 1. Task

**Nhận xét bài toán**

- Lớp học có n học sinh, mỗi bàn có 2 chỗ ngồi, tạo thành một danh sách liên tục từ 1 đến n.
- Vị trí Alice ngồi có thể được chuyển đổi thành số thứ tự z trong danh sách liên tục: z = (p - 1) * 2 + q
- Bob phải ngồi cách Alice đúng k vị trí, tức là tại z - k hoặc z + k.
- Cần tìm vị trí (u, v) của Bob sao cho hợp lệ và gần Alice nhất.

**Hướng giải quyết**

1. Tính vị trí của Alice trong danh sách liên tục:
   z = (p - 1) * 2 + q
2. Xác định hai vị trí Bob có thể ngồi:
   pos1 = z – k
   pos2 = z + k
3. Kiểm tra tính hợp lệ của pos1 và pos2 (phải nằm trong khoảng từ 1 đến n).
4. Chuyển đổi vị trí hợp lệ thành (u, v) bằng công thức:
   u = (pos + 1) / 2
   v = pos - 2 * (u - 1)
5. Chọn vị trí gần Alice nhất, nếu có nhiều vị trí hợp lệ.
6. Xuất kết quả hoặc in -1 nếu không tìm được vị trí hợp lệ.

**Độ phức tạp** O(1).

### 2. Point3D

**Hướng giải quyết**

Sắp xếp danh sách:

- Dùng hàm sắp xếp với tiêu chí:
  - So sánh theo x tăng dần.
  - Nếu x bằng nhau, so sánh theo y giảm dần.
  - Nếu y cũng bằng nhau, so sánh theo z tăng dần

**Độ phức tạp** O(N log N).

## 3. an.512 Trộn 2 mảng

**Hướng giải quyết**

Trộn hai mảng:

- Dùng hai con trỏ i và j:
  - Nếu A[i] >= B[j], thêm A[i] vào kết quả và tăng i.
  - Nếu A[i] < B[j], thêm B[j] vào kết quả và tăng j.
- Nếu còn phần tử nào chưa duyệt hết, thêm tất cả vào kết quả.

**Độ phức tạp** O(n + m).

## 4. Find MEX

**Nhận xét bài toán**

- Vì N có thể lên đến $10^5$ và giá trị phần tử lên đến $10^9$, việc sử dụng mảng đánh dấu sẽ tốn quá nhiều bộ nhớ.
- Giải pháp tối ưu là nén số, chỉ quan tâm đến các số từ 0 đến N vì MEX sẽ không vượt quá N.

**Hướng giải quyết**

1. Nén số lại:
   - Bỏ qua các số lớn hơn N vì chúng không ảnh hưởng đến MEX.
   - Dùng một mảng đánh dấu check có kích thước N + 1 để lưu trạng thái xuất hiện của các số trong phạm vi [0, N].
2. Duyệt qua mảng đầu vào:
   - Nếu phần tử a[i] nằm trong khoảng [0, N], đánh dấu check[a[i]] = 1.
3. Tìm MEX:
   - Duyệt từ 0 đến N, giá trị đầu tiên check[i] = 0 chính là MEX cần tìm.

**Độ phức tạp** O(n log(n))

## 5. Point2D

**Hướng giải quyết**

Dùng hàm sắp xếp với tiêu chí:

- Sắp xếp tăng dần theo x.
- Nếu x bằng nhau thì sắp xếp giảm dần theo y.

**Độ phức tạp** O(N log N).

## 6. VU33_MaxStr

**Nhận xét bài toán**

- Bài toán yêu cầu tìm số lớn nhất có thể tạo ra bằng cách sắp xếp lại các chữ số của xâu đầu vào, đồng thời có thể xóa một số chữ số nếu cần.
- Điều kiện để số hợp lệ là **chia hết cho 3**.
- Nếu không thể tạo số hợp lệ, cần in ra số **0** lớn nhất có thể.

**Hướng giải quyết**

1. Chuyển các ký tự trong xâu thành số nguyên, lưu vào mảng ar, đồng thời tính tổng sum của tất cả chữ số.
2. Sắp xếp ar theo thứ tự giảm dần để tạo số lớn nhất có thể.
3. Nếu sum chia hết cho 3, thì in ra số lớn nhất từ ar.
4. Nếu sum % 3 != 0:
    - Thử xóa một chữ số có phần dư bằng sum % 3.
    - Nếu không xóa được, thử xóa hai chữ số sao cho tổng phần dư bằng sum % 3.
5. Nếu sau khi xử lý mà mảng ar rỗng hoặc chỉ còn 0, in ra số 0 lớn nhất từ xâu đầu vào.
6. Nếu vẫn còn số hợp lệ, in ra các chữ số từ ar theo thứ tự giảm dần.

## 7. VQ44_FLOWERS

**Nhận xét bài toán**

- Bài toán yêu cầu chọn ra k bóng đèn LED sao cho số lượng màu khác nhau lắp được là nhiều nhất.
- Các màu đèn được lưu dưới dạng một danh sách số nguyên.
- Chúng ta cần đếm tần suất các màu và lựa chọn k màu phù hợp.

**Độ phức tạp** O(n log n).

## 8. Kiểm Kê

**Hướng giải quyết**

1. Sắp xếp danh sách theo thứ tự tăng dần theo quy tắc:
    - So sánh theo độ dài xâu (ngắn hơn trước).
    - Nếu bằng nhau, so sánh từ điển.
2. Duyệt danh sách và đếm các giá trị khác nhau.

**Độ phức tạp** O(n log n).

## 9. MergeSort

**Hướng giải quyết:**

1. **Hàm merge**:
    - Chia mảng thành hai phần: L (trái) và R (phải).
    - So sánh từng phần tử của L và R, chèn vào mảng ban đầu theo thứ tự tăng dần.
    - Nếu còn phần tử chưa được chèn vào, tiếp tục thêm vào mảng.
2. **Hàm mergeSort**:
    - Nếu mảng có nhiều hơn một phần tử, chia đôi mảng thành hai phần bằng nhau.
    - Gọi đệ quy sắp xếp phần bên trái và phần bên phải.
    - Gọi hàm merge để trộn hai phần đã sắp xếp.
    - Sau mỗi lần trộn, in ra cấu trúc của mảng A:
        - Các phần tử trước đoạn đang xử lý.
        - Đoạn đang được trộn trong dấu "[ ]".
        - Các phần tử sau đoạn đang xử lý.

**Độ phức tạp:** Tổng độ phức tạp của thuật toán là **O(N log N)**

## 10. InsertionSort

**Hướng giải quyết**

1. Duyệt qua các phần tử của mảng từ vị trí thứ hai (vì phần tử đầu tiên coi như đã được sắp xếp).
2. Chèn từng phần tử vào vị trí thích hợp trong dãy con đã sắp xếp:
    - Lấy giá trị của phần tử hiện tại làm "key".
    - Di chuyển các phần tử lớn hơn "key" vào vị trí sau nếu càn thiết.
    - Cố định "key" vào vị trí thích hợp.
3. Mỗi khi xảy ra swap, in ra trạng thái hiện tại của mảng.

**Độ phức tạp** O(N^2)

## 11. BubbleSort

**Hướng giải quyết**

1. Duyệt qua mảng nhiều lần, mỗi lần so sánh từng cặp phần tử liên tiếp.
2. Nếu phần tử trước lớn hơn phần tử sau, hoán đổi chúng để đưa phần tử lớn hơn về phía cuối mảng.
3. Sau mỗi lần duyệt, phần tử lớn nhất còn lại sẽ được đưa về vị trí cuối cùng của phạm vi chưa được sắp xếp.
4. Tiếp tục quá trình cho đến khi không còn swap nào xảy ra.
5. Mỗi khi xảy ra swap, in ra trạng thái hiện tại của mảng.

**Độ phức tạp** O(N^2).

## 12. SelectionSort

**Hướng giải quyết**

1. Duyệt qua từng vị trí trong mảng, giả sử vị trí đó chứa phần tử nhỏ nhất trong đoạn chưa sắp xếp.
2. Tìm phần tử nhỏ nhất trong đoạn chưa sắp xếp và ghi nhận vị trí của nó.
3. Hoán đổi phần tử nhỏ nhất với phần tử ở vị trí hiện tại, nếu cần thiết.
4. Sau mỗi lần hoán đổi, in ra trạng thái của mảng.
5. Lặp lại quá trình cho đến khi toàn bộ mảng được sắp xếp.