# Nhiệm vụ: save\_answer - phần 1 nhận input của user rồi lưu vào list answer

Viết 1 hàm, đặt tên bất kỳ, thực hiện các nhiệm vụ sau:

- nhận tham số:

+ ID câu hỏi

+ Câu trả lời của học viên

- Tạo 1 mảng global để chứa cặp giá trị [ID câu hỏi, câu trả lời học viên]

- Thực hiện được các thao tác: thêm, xóa, truy xuất cặp dữ liệu vào dạng dữ liệu mảng

**Chọn loại dữ liêu mảng (trong python) nào để lưu cặp dữ liệu, hãy viết báo cáo lý luận vì sao dùng kiểu dữ liệu đó để xử lý mà không phải cặp khác.**

Với nhiệm vụ này, khi đưa vào Chat GPT gợi ý thì có 2 phương án có thể thực hiện là List và Dictionary.  
  
Lúc đầu mình cũng phân vân không biết chọn phương án nào, cho đến khi có manh mối

Hehe mọi người xem kỹ nhen , mình sẽ trình bày ở đây

## A - KỂ CHUYỆN

|  |  |  |
| --- | --- | --- |
| List | Manh mối | Dictionary |
|  |  |  |
|  |  |  |
| MANH MỐI LẠI KHÔNG CÓ TRONG NÀY | | |
| ĐẾN KHI MÌNH DÙNG CẤU LỆNH “ hãy giải thích cho tôi ý nghĩa của từng thao tác trong đoạn mã lập trình đó và minh họa cho thật dễ hiểu.” | | |
| Lúc này bên phương án Dictionary xuất hiện cụm từ : thời gian tìm kiếm của nó là **O(1)**, độ phức tạp thời gian là **O(1).** | | |
| Từ chỗ độ phức tạp thời gian này nè, mình đi bắt con AI nó giải thích và phân tích độ phức tạp thời gian trong từng bước lập trình của 2 phương án trên và so nó cho mình, nó ra cái bảng này | | |
|  | | |
|  | | |
| Rồi tới đây là tui có thể có được căn cứ để lựa chọn phương án nào rồi nè, chắc chắn là phương án 2 rồi. | | |
| OK viết CODE và gửi thôi | | |

## B- TRÌNH TỰ SỬ DỤNG CHAT GPT

|  |
| --- |
| B- TRÌNH TỰ SỬ DỤNG CHAT GPT |
| Trình tự nhờ chat GPT hỗ trợ của tui ( phần này có Chit độ hôm chủ nhật , AI độ, và sự tò mò độ. |
| B1: Tôi là một lập trình viên Python  ( chú thích : cái này để AI nó biết khoanh vùng lĩnh vực kiến thức) |
| B2: tôi cần xử lý tác vụ lập trình liên quan đến thao tác với mảng trong python |
| B3: tôi cần bạn giúp đỡ giải thích yêu cầu và đưa ra ví dụ  Viết 1 hàm, đặt tên bất kỳ, thực hiện các nhiệm vụ sau:  - nhận tham số:  + ID câu hỏi  + Câu trả lời của học viên  - Tạo 1 mảng global để chứa cặp giá trị [ID câu hỏi, câu trả lời học viên]  - Thực hiện được các thao tác: thêm, xóa, truy xuất cặp dữ liệu vào dạng dữ liệu mảng  Tôi chưa biết chọn loại mảng nào trong python để xử lý yêu cầu trên , hãy gợi ý và phân tích cho tôi những lựa chọn hợp lí |
| B4: Tuyệt vời, tôi rất thích cách giải thích cặn kẽ, dễ hiểu này của bạn.  Tôi muốn hỏi thêm  Trong Phương án 1: Sử dụng List, mà ở phần đầu tôi đề nghị bạn đề xuất cách giải cho tôi á, hãy giải thích cho tôi ý nghĩa của từng thao tác trong đoạn mã lập trình đó và minh họa cho thật dễ hiểu |
| B5: Bạn thật sự quá đỉnh, tôi đã hiểu rõ vấn đề.  Tương tự, trong Phương án 2: Sử dụng Dictionary, mà ở phần đầu tôi đề nghị bạn đề xuất cách giải cho tôi á, hãy giải thích cho tôi ý nghĩa của từng thao tác trong đoạn mã lập trình đó và minh họa cho thật dễ hiểu. |
| B6: tôi muốn hỏi thêm: cụm từ độ phức tạp thời gian là O(1) nghĩa là sao vậy , trong python có chia cấp độ phức tạp thời gian không, nó là những cấp độ gì, bạn giải thích chi tiết cho tôi với |
| B7: Hay quá, tôi muốn bạn phân tích độ phức tạp thời gian trong 2 phương án giải đề toán mà bạn vừa giải thích chi tiết cặn kẽ cho tôi. |
| B8: Hỏi thêm hầm bà lằng các thứ như cách làm phương án 2 tối ưu hơn nữa về độ phức tạp thời gian  , rồi check độ phức tạp thời gian, nó đẻ thêm chi phí lưu trữ dữ liệu, phải dùng cơ sở dữ liệu (cần thêm công cụ , kiến thức, chi phí ),……. Bơi đã rồi mình quay về phương án 2 và trình bày code để nộp chứ tới deadline rồi, mọi người có thể bới thêm nhen nếu tò mò |
|  |

## C- Phụ lục PHÂN TÍCH ĐỘ PHỨC TẠP THỜI GIAN

### Phương án 1: Sử dụng List

Trong phương án này, chúng ta sử dụng List để lưu trữ các cặp [ID câu hỏi, câu trả lời]. Bây giờ, tôi sẽ phân tích độ phức tạp thời gian của từng thao tác:

1. **Thêm dữ liệu** (action == 'add'):
   * **Hoạt động**: Thêm một cặp mới [ID câu hỏi, câu trả lời] vào cuối List.
   * **Độ phức tạp thời gian**:
     + Việc thêm vào cuối List trong Python có độ phức tạp thời gian là **O(1)** vì thao tác này chỉ yêu cầu thêm một phần tử vào cuối mà không cần di chuyển các phần tử khác.
2. **Xóa dữ liệu** (action == 'remove'):
   * **Hoạt động**: Xóa một cặp dữ liệu dựa trên ID câu hỏi.
   * **Độ phức tạp thời gian**:
     + Trước tiên, chúng ta cần **duyệt qua toàn bộ List** để tìm cặp có ID câu hỏi trùng khớp. Việc này có độ phức tạp **O(n)**, với n là số lượng phần tử trong List.
     + Sau đó, nếu tìm thấy phần tử, chúng ta thực hiện việc xóa, thao tác này có độ phức tạp **O(1)** vì nó chỉ đơn giản xóa phần tử sau khi đã tìm được vị trí.

**Tổng kết**: Độ phức tạp của thao tác xóa là **O(n)** do cần duyệt qua List để tìm phần tử.

1. **Truy xuất dữ liệu** (action == 'get'):
   * **Hoạt động**: Truy xuất câu trả lời dựa trên ID câu hỏi.
   * **Độ phức tạp thời gian**:
     + Giống như thao tác xóa, chúng ta cần **duyệt qua toàn bộ List** để tìm cặp dữ liệu có ID câu hỏi trùng khớp. Điều này có độ phức tạp là **O(n)**.
2. **Liệt kê tất cả dữ liệu** (action == 'list\_all'):
   * **Hoạt động**: Liệt kê tất cả các cặp [ID câu hỏi, câu trả lời] trong List.
   * **Độ phức tạp thời gian**:
     + Duyệt qua toàn bộ List và in ra từng phần tử. Điều này có độ phức tạp là **O(n)**, với n là số lượng phần tử trong List.

**Tổng kết độ phức tạp của Phương án 1:**

| **Thao tác** | **Độ phức tạp thời gian** |
| --- | --- |
| **Thêm dữ liệu** | O(1) |
| **Xóa dữ liệu** | O(n) |
| **Truy xuất dữ liệu** | O(n) |
| **Liệt kê tất cả dữ liệu** | O(n) |

**Nhận xét**: Phương án sử dụng List có thời gian **thêm** rất nhanh, nhưng thao tác **xóa** và **truy xuất** có độ phức tạp **O(n)**, đặc biệt nếu danh sách rất lớn, việc tìm kiếm sẽ tốn thời gian.

### Phương án 2: Sử dụng Dictionary

Phương án này sử dụng Dictionary để lưu trữ các cặp [ID câu hỏi, câu trả lời]. Tôi sẽ phân tích độ phức tạp của từng thao tác dưới đây:

1. **Thêm hoặc cập nhật dữ liệu** (action == 'add'):
   * **Hoạt động**: Thêm hoặc cập nhật câu trả lời cho ID câu hỏi.
   * **Độ phức tạp thời gian**:
     + Truy cập và thêm phần tử trong Dictionary có độ phức tạp **O(1)**, vì Dictionary sử dụng hash table để quản lý key-value, cho phép truy xuất nhanh dựa trên key.
2. **Xóa dữ liệu** (action == 'remove'):
   * **Hoạt động**: Xóa câu trả lời dựa trên ID câu hỏi.
   * **Độ phức tạp thời gian**:
     + Tương tự như thao tác thêm, xóa một cặp key-value từ Dictionary cũng có độ phức tạp **O(1)**.
3. **Truy xuất dữ liệu** (action == 'get'):
   * **Hoạt động**: Truy xuất câu trả lời dựa trên ID câu hỏi.
   * **Độ phức tạp thời gian**:
     + Truy xuất giá trị từ Dictionary bằng key có độ phức tạp **O(1)**.
4. **Liệt kê tất cả dữ liệu** (action == 'list\_all'):
   * **Hoạt động**: Liệt kê tất cả các cặp [ID câu hỏi, câu trả lời] trong Dictionary.
   * **Độ phức tạp thời gian**:
     + Duyệt qua tất cả các phần tử trong Dictionary có độ phức tạp là **O(n)**, với n là số lượng phần tử trong Dictionary.

**Tổng kết độ phức tạp của Phương án 2:**

| **Thao tác** | **Độ phức tạp thời gian** |
| --- | --- |
| **Thêm hoặc cập nhật dữ liệu** | O(1) |
| **Xóa dữ liệu** | O(1) |
| **Truy xuất dữ liệu** | O(1) |
| **Liệt kê tất cả dữ liệu** | O(n) |

**Nhận xét**: Phương án sử dụng Dictionary tối ưu hơn rất nhiều so với List trong các thao tác **thêm**, **xóa**, và **truy xuất**, tất cả đều có độ phức tạp **O(1)**. Điều này giúp giảm thời gian xử lý đáng kể khi bạn có số lượng lớn các câu hỏi và câu trả lời.

### So sánh Phương án 1 (List) và Phương án 2 (Dictionary)

| **Thao tác** | **Phương án 1 (List)** | **Phương án 2 (Dictionary)** |
| --- | --- | --- |
| **Thêm dữ liệu** | O(1) | O(1) |
| **Xóa dữ liệu** | O(n) | O(1) |
| **Truy xuất dữ liệu** | O(n) | O(1) |
| **Liệt kê tất cả dữ liệu** | O(n) | O(n) |

* **Phương án 1** (List) có ưu điểm ở thao tác **thêm** với độ phức tạp **O(1)**, nhưng nhược điểm là các thao tác **xóa** và **truy xuất** có độ phức tạp **O(n)**.
* **Phương án 2** (Dictionary) hiệu quả hơn với các thao tác **thêm**, **xóa**, và **truy xuất**, đều có độ phức tạp **O(1)**. Tuy nhiên, cả hai phương án đều có độ phức tạp **O(n)** khi liệt kê toàn bộ dữ liệu.