

**Câu hỏi trắc nghiệm**

Lưu ý: chỉ chọn một đáp án duy nhất.

**Câu 1.** Stack và đệ quy có liên hệ như thế nào với nhau?

1. Stack thường được sử dụng để tăng hiệu suất các thuật toán đệ quy
2. Các phương thức thêm và lấy phần tử trong stack được thực hiện bằng phương pháp đệ quy
3. Thứ tự thực hiện các lệnh gọi đệ quy được lưu trữ bằng một cấu trúc stack
4. Stack là một cấu trúc dữ liệu, còn đệ quy là một phương pháp lập trình nên hai khái niệm này không có liên hệ với nhau

**Câu 2.** Stack overflow, hay tràn bộ nhớ stack xảy ra khi:

1. Class *deque* của Python lưu trữ nhiều hơn số phần tử cho phép
2. Các lệnh gọi đệ quy chiếm nhiều không gian trong vùng nhớ stack hơn mức cho phép
3. A và B
4. Không ý nào đúng

**Câu 3.** Khi nào ta có thể lấy ra phần tử đầu tiên được đưa vào stack?

1. Khi phần tử đó vừa được đưa vào stack
2. Khi phần tử đó là phần tử cuối cùng trong stack
3. Khi stack chỉ chứa duy nhất một phần tử
4. Tất cả các ý trên

**Câu 4.** Một stack đang chứa 5 phần tử **[0, 1, 2, 3, 4]** theo thứ tự phần tử mới nhất nằm bên phải. Làm thế nào để chèn phần tử **5** vào phía ngoài cùng bên trái của stack để được **[5, 0, 1, 2, 3, 4]**? Biết stack hoạt động theo nguyên tắc *vào sau ra trước*.

1. Thêm phần tử đó vào stack theo đúng nguyên tắc trên.
2. Tạo một stack mới, thêm phần tử đó vào stack mới, sau đó lần lượt lấy các phần tử trong stack cũ ra và thêm vào stack mới.
3. Tạo một stack mới, lần lượt lấy các phần tử trong stack cũ ra và thêm vào stack mới, đưa phần tử **5** vào stack cũ, sau đó lần lượt lấy các phần tử trong stack mới ra và đưa lại vào stack cũ.
4. Không thể thêm phần tử đó vào stack ở vị trí nêu trên do vi phạm nguyên tắc *vào sau ra trước*.

**Câu 5.** Tính chất *vào trước ra trước* trong queue **không** biểu hiện trong ví dụ thực tế nào dưới đây?

1. Thứ tự tháo lắp các bộ phận xe máy
2. Thứ tự chờ soát vé lên máy bay, không tính vé ưu tiên
3. Thứ tự các xe đi qua trạm thu phí
4. Thứ tự xử lý đơn đặt hàng trong quán ăn

**Câu 6.** Cho quá trình thêm và lấy các phần tử trong một cấu trúc dữ liệu dưới đây.

0 🡲 0 1 🡲 0 1 2 🡲 3 0 1 2

Cấu trúc dữ liệu này **không** thể là:

1. Stack
2. Queue
3. Deque
4. A và B

**Câu 7.** Tại sao ta dùng stack hoặc queue mà không dùng deque cho mọi trường hợp?

1. Ta cần sự giới hạn về phương thức của stack hoặc queue để đảm bảo tính chất *vào sau ra trước* hoặc *vào trước ra trước*
2. Nhiều bài toán chỉ có thể được giải quyết nếu ta cho phép thêm và lấy phần tử theo một quy tắc nhất định
3. Ta sử dụng class *deque* của Python để cài đặt stack hoặc queue một cách thuận tiện. Trong nhiều trường hợp, ta không cần sử dụng tất cả các phương thức của deque.
4. Tất cả các ý trên

**Câu 8.** Trong bài học, ta cài đặt stack và queue bằng class *deque* thay cho list vì:

1. Stack và queue không thể được cài đặt bằng list
2. Stack và queue có thể được cài đặt bằng list, tuy nhiên việc cài đặt sẽ làm cho code phức tạp hơn rất nhiều
3. Stack và queue có thể được cài đặt bằng list, tuy nhiên các phương thức thêm và xóa phần tử ở hai đầu của list có độ phức tạp về thời gian cao hơn các phương thức ở class *deque*
4. B và C

# Đáp án

| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | B | D | C | A | D | D | C |