

**Câu hỏi trắc nghiệm**

Lưu ý: chỉ chọn một đáp án duy nhất.

**Câu 1.** Ý nào sau đây là một thuật toán?

1. Các bước để tìm tất cả các số nguyên tố bé hơn 100.
2. Hướng dẫn nấu mì trên gói mì ăn liền.
3. Cách vẽ một hình tròn không sử dụng compa.
4. Tất cả các ý trên.

**Câu 2.** Phát biểu nào sau đây đúng?

1. Độ phức tạp về thời gian bằng đúng số phép tính mà thuật toán phải xử lý.
2. Độ phức tạp về thời gian thể hiện mức độ thay đổi về thời gian chạy của thuật toán khi kích cỡ dữ liệu đầu vào thay đổi.
3. Một thuật toán với độ phức tạp thời gian *O(1)* luôn luôn chạy nhanh hơn một thuật toán *O(n2)*.
4. Một thuật toán với độ phức tạp thời gian *O(1)* luôn luôn chạy nhanh hơn một thuật toán *O(n2)*, khi n > 10*6*.

**Câu 3.** Cho một số thuật toán dùng để giải quyết cùng một vấn đề nhưng có độ phức tạp về thời gian khác nhau. Thuật toán nào hiệu quả nhất về mặt thời gian?

1. O(1)
2. O(n)
3. O(2n)
4. Không xác định được.

**Câu 4.** Cho một số thuật toán dùng để giải quyết cùng một vấn đề nhưng khác nhau về tổng số phép toán cần xử lý. Giả sử mỗi phép toán có thời gian xử lý như nhau và kích cỡ dữ liệu đầu vào là n = 1000. Cho tổng số phép toán mà mỗi thuật toán phải xử lý, thuật toán nào hiệu quả nhất về mặt thời gian?

1. 106
2. 3n + 255
3. 2n + n + 4
4. Không xác định được.

**Câu 5.** Độ phức tạp nào sau đây tương đương với *O(n2)*?

1. O(106)
2. O(n + log(n))
3. O((n+1)2)
4. O(2n + n + 4)

**Câu 6.** Cho hai cách khác nhau để kiểm tra list *arr* có chứa số âm hay không.

Cách 1:

**found\_negative = False**

**for el in arr:**

**if el < 0:**

**found\_negative = True**

Cách 2:

**found\_negative = False**

**for el in arr:**

**if el < 0:**

**found\_negative = True**

**break**

Gọi n là số phần tử trong *arr*, phát biểu nào sau đây là đúng về độ phức tạp thời gian của thuật toán trong hai cách trên?

1. Cách 1 và cách 2 đều có độ phức tạp về thời gian là O(n).
2. Cách 1 có độ phức tạp O(n). Cách 2 có độ phức tạp bé hơn O(n).
3. Cách 1 có độ phức tạp O(n). Cách 2 có độ phức tạp O(k), với k là vị trí của số âm đầu tiên trong *arr*.
4. Cách 1 và cách 2 đều có độ phức tạp về thời gian là O(k), với k là vị trí của số âm đầu tiên trong *arr*.

**Câu 7.** Làm thế nào để biết độ phức tạp của một hàm trong thư viện, ví dụ hàm *sum()* được tích hợp sẵn trong Python?

1. Viết code để kiểm tra thời gian chạy thực tế của hàm đó trên các kích cỡ dữ liệu đầu vào khác nhau.
2. So sánh thời gian chạy của hàm đó với với thời gian chạy của một hàm có độ phức tạp đã biết.
3. Tìm kiếm đặc tả của hàm đó trên trang thông tin của thư viện hoặc của nhà phát triển tương ứng.
4. Không thể biết được độ phức tạp của một hàm cho sẵn.

**Câu 8.** Phát biểu nào sau đây về độ phức tạp về không gian là không đúng?

1. Độ phức tạp về không gian được tính bằng lượng bộ nhớ thực tế mà thuật toán cần sử dụng.
2. Độ phức tạp về không gian thể hiện mức độ thay đổi về không gian nhớ mà thuật toán sử dụng khi kích cỡ dữ liệu đầu vào thay đổi.
3. Độ phức tạp không gian O(1) mang ý nghĩa: Không gian nhớ mà thuật toán sử dụng không phụ thuộc vào kích cỡ dữ liệu đầu vào.
4. Độ phức tạp không gian O(n) mang ý nghĩa: Không gian nhớ mà thuật toán sử dụng có quan hệ tuyến tính với kích cỡ dữ liệu đầu vào.

**Câu 9.** Một thuật toán sử dụng đúng 2 *list* chứa các số nguyên và 3 biến phụ để tính toán, biết mỗi *list* chứa n phần tử. Như vậy, thuật toán có độ phức tạp về không gian là:

1. O(1)
2. O(n)
3. O(n2)
4. Kết quả khác

**Câu 10.** Kết quả của biểu thức nào sau đây trong Python không phải là một cấu trúc dữ liệu?

1. **[1] + [3]**
2. **2 in {1, 2, 3, 4, 5}**
3. **"MindX Technology School"[:5]**
4. **{2: "Monday", 3: "Tuesday"}[2]**

# Đáp án

| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D | B | D | B | C | A | C | A | B | B |