VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Cấu Trúc Dữ Liệu và Giải Thuật (DSA)

DSA3 - HK242

Giữa Kì

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 3/2025

Mục lục

1 Lý thuyết cơ bản

- Độ phức tạp thời gian mô tả thời gian cần thiết để thực hiện một thuật toán, thường được biểu diễn bằng ký hiệu Big-O.
- Các hằng số theo $+$, $-$, và $/$ có thể bỏ qua; ví dụ $O(2n/4+1)$ tương đương với $O(n)$.
- Vòng `for` từ 0 đến n có độ phức tạp $O(n)$.
- Vòng `for` từ 0 đến $\sqrt{n}$ có độ phức tạp $O(\sqrt{n})$.
- Các vòng for thường lồng nhau sẽ là tích độ phức tạp còn các vòng for không lồng nhau mà ngang hàng code (tâm vực) sẽ là tổng lúc này xem nào lớn hơn
- Vòng `for` từ 0 đến $\log n$ có độ phức tạp $O(\log n)$.
- Các thuật toán có độ phức tạp $O(\log n)$ thường là những thuật toán tìm kiếm trong mảng đã sắp xếp, như tìm kiếm nhị phân.
- Độ phức tạp $O(2^n)$ mô tả các thuật toán có số lượng phép toán tăng theo hàm mũ với kích thước đầu vào, ví dụ như trong các bài toán hoán vị hoặc bài toán Fibonacci không tối ưu.
- Ý tưởng đề quy dự đoán hàm đo làm gì ví dụ tên hàm foo nhưng ta độc code xác định nó là tính dây thừa

```
int foo(int n) {
    if (n == 0) return 1; // Trường hợp cơ bản
    return n * foo(n - 1); // Quy tắc đệ quy
}
```

- thông thường các hàm đệ quy trong đề sẽ là tính dây thừa, tính tích, tính tổng, tính mũ, dãy fibo

2 Câu hỏi trong các đề thi

1. Program gồm những gì để tạo thành 1 chương trình?

   a) Data structures
   b) Algorithms
   c) Logical
   d) Data structures + Algorithms

2. Bài toán nào sau đây có thể không dùng đệ quy để giải?

   a) Knight's tour
   b) Tower of Hanoi
   c) Fibonacci number
   d) Cả 3 bài toán trên đều có thể

3. Cho biết độ phức tạp của đoạn code sau:

```
for(int i = 0; i < n; i++)
    for(int j = 0; j <= i; j++)
        for(int k = 1; k < n; k = k * 3)
            print("midterm")
```

   a) $O(N^2)$
   b) $O(N^2)\log(N)$
   c) $O(N^3)$
   d) $O(N\log(N))$

4. Kỹ thuật nào được dùng để giải bài toán n-queens?

   a) Backtracking
   b) Dynamic programming
   c) Divide and conquer
   d) Greedy

5. Cho biết độ phức tạp của đoạn code sau:

```
int fibona(int n) {
    if (n <= 1) return 1;
    return fibona(n-1) + fibona(n-2);
}
```

   a) $O(2^N)$
   b) $O(N^N)\log(N)$
   c) $O(N^2)$
   d) $O(N\log(N))$

6. Cho biết độ phức tạp của đoạn code sau:

```
int ans = 0;
while (n > 0) {
    ans += n;
    n /= 2;
}
```

a) $O(N)$ b) $O(\log(N))$ c) $O(N^2)$ d) $O(N\log(N))$

7. Để đo độ phức tạp về thời gian của một thuật toán, ký hiệu Big O được sử dụng?

a) Đã đầu vào (input) thực tế khi thực thi giải thuật chương trình
b) Căn cứ ước tính, đo đạc thời gian thực thi của một thuật toán, giải thuật
c) Mô tả những giới hạn chức năng của một hàm hoặc của một giải thuật.
d) Đã ngưỡng chặn trên để xem xét và đánh giá giải thuật chạy trong trường hợp trung bình.

8. Cho biết độ phức tạp của đoạn code sau:

```
for(int i = 0; i < n; i++)
    for(int j = 0; j <= i; j++)
        for(int k = 1; k < n; k = k + 3)
            print("midterm")
```

a) $O(N^2)$ b) $O(N^2)\log(N)$ c) $O(N^3)$ d) $O(N\log(N))$

9. Cho biết độ phức tạp của đoạn code sau: giả sử $func$ có độ phức tạp là $O(N\log(N))$

```
for(int i = 0; i < n; i++)
    func(i);
```

a) $O(N^2)$ b) $O(N^2)\log(N)$ c) $O(N^3)$ d) $O(N\log(N))$

10. So sánh các độ phức tạp trên $a = O(2*N+1)$, $b = O(3*N+2)$, $c = O(2*N^2)$, $d = O(2*N\log(N))$, $e = O(2^N)$

a) a = b < c < d < e b) a < b < c < d < e
c) a = b < d < c < e d) a = b < d < e < c

11. Chỉ ra độ phức tạp của biểu thức: $n^2 + 35n + 6n\log(n)$

a) $O(n^3)$ b) $O(n^2)$ c) $O(n)$ d) $O(42)$

12. Tìm độ phức tạp lớn O của biểu thức sau (đầu ra của $f(n)$):

$$f(n) = \begin{cases} 2f(n-1)-1, & n>0 \\ 1, & n=0 \end{cases}$$

a) $O(1)$ b) $O(n)$ c) $O(n^3)$ d) $O(3^n)$

13. Chỉ ra độ phức tạp lớn O của chương trình sau:

```
i=1;
while(i<=n) i=i*3;
```

a) $O(n\cdot 3)$ b) $O(n^3)$ c) $O(i)$ d) Đáp án khác

14. Biết rằng Mệnh đề A và Mệnh đề B đều có độ phức tạp $O(1)$. Chỉ ra độ phức tạp của thuật toán sau:

```
for(int i = 0; i<n; i++) Statement A;
for(int j = 0; j<m; j++) Statement B;
```

a) $O(n+m)$ b) $O(\max(n,m))$ c) $O(n\cdot m)$ d) $O(i+j)$

15. Chỉ ra độ phức tạp của đoạn mã sau:

```
for(int i = 1; i<=n; i++)
for(int j=1; j<=n; j++){
  cout << "*";
  break;
}
```

a) $O(n^2)$ b) $O(n)$ c) $O(1)$ d) Đáp án khác

16. Cho chương trình sau:

```
s=0;
for(int i=1; i<m; i++)
for(int j=0; j<=i; j++)
s+=j; //Dòng 1
```

Dòng 1 sẽ được thực thi bao nhiêu lần?

a) $\frac{(m+1)(m-1)}{2}$ b) $\frac{m(m-1)}{2}$ c) $\frac{(m+2)(m-1)}{2}$ d) $\frac{m(m+1)}{2}$

17. Cho hàm sau:

```
void func1(int arr[], int n){
    for(int i=n-1;i>=1;i--)
        for(int j=0; j<i;j++)
            if(arr[j]>arr[j+1])
                swap(arr[j], arr[j+1]); //Dòng 1
}
```

Trong trường hợp xấu nhất, dòng 1 sẽ được thực thi bao nhiêu lần?

a) $\frac{n(n-1)}{2}$ b) n^2 c) $\frac{n(n+1)}{2}$ d) Đáp án khác

18. Cho đoạn mã sau:

```
void funcA(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            funcB(n);
        }
    }
}

void funcB(int n) {
    for (int k = 0; k < n; k++) {
        cout << "*";
    }
}
```

Tính độ phức tạp của hàm 'funcA(int n)' khi nó gọi hàm 'funcB(int n)'.

a) $O(n^3)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(n)$

19. Cho đoạn mã sau:

```
void funcC(int n) {
    for (int i = 0; i < n; i++) {
        funcD(i);
    }
}

void funcD(int m) {
    for (int j = 0; j < m; j++) {
        cout << "*";
```

```
    }
}
```

Tính độ phức tạp của hàm 'funcC(int n)' khi nó gọi hàm 'funcD(int m)'.

a) $O(n^2)$ b) $O(n \log n)$ c) $O(n)$ d) $O(n!)$

20. Xác định độ phức tạp của đoạn mã sau:

```
void funcE(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < sqrt(n); j++) {
            cout << "*";
        }
    }
}
```

Độ phức tạp của hàm 'funcE(int n)' là bao nhiêu?

a) $O(\sqrt{n})$ b) $O(\sqrt{n} \log n)$ c) $O(n)$ d) $O(\log n)$

21. Cho đoạn mã sau:

```
void funcF(int n) {
    for (int i = 1; i <= sqrt(n); i++) {
        for (int j = 1; j <= n; j *= 2) {
            cout << "*";
        }
    }
}
```

Tính độ phức tạp của hàm 'funcF(int n)'.

a) $O(n)$ b) $O(\sqrt{n} \log n)$ c) $O(\sqrt{n})$ d) $O(n \log n)$

22. Cho một hàm:

```
void my_recursive_function(int n){
    if(n == 0) return;
    my_recursive_function(n-1);
    cout << n << " ";
}
```

Kết quả của việc gọi hàm `my_recursive_function(10)` là gì?

a) In ra các số từ 10 đến 1 b) In ra các số từ 10 đến 0
c) In ra các số từ 1 đến 10 d) In ra các số từ 0 đến 10

23. Chỉ ra kết quả thực thi của chương trình sau:

```
int fibo(int n){
    if(n == 1) return 0;
    else if(n == 2) return 1;
    return fibo(n - 1) + fibo(n - 2);
}

int main() {
    int n = -1;
    int ans = fibo(n);
    cout << ans;
    return 0;
}
```

a) 0 b) 1 c) Lỗi biên dịch d) Lỗi chạy thời gian

24. Kết quả trên màn hình của chương trình sau là gì:

```
long int fun3(int x, int y){
    if(x > y) return -1;
    else if(x == y) return 1;
    else return x * fun3(x + 1, y);
}

int main() {
    cout << fun3(3, 7);
    return 0;
}
```

a) 360 b) 400 c) 300 d) 320

25. Câu nào sau đây KHÔNG ĐÚNG về đệ quy vô hạn?

a) Đệ quy vô hạn sẽ làm cho chương trình bị treo.
b) Đệ quy vô hạn tiêu tốn toàn bộ bộ nhớ của hệ thống cho chương trình và gây ra lỗi dừng bất thường.
c) Gọi đệ quy gián tiếp luôn gây ra đệ quy vô hạn.
d) Nếu lời gọi đệ quy không đến được trường hợp cơ bản, đệ quy vô hạn sẽ xảy ra.

26. Kết quả của lệnh: `func(301)` của hàm sau:

```
void func(int number){
    if(number <= 0) return;
    else{
        func(number/2);
        cout << number%2;
    }
}
```

a) 100101101 b) 101101001 c) 100101010 d) 101101010

27. Có bao nhiêu lần hàm `fibo` được gọi khi thực thi chương trình dưới đây?

```
int fibo(int n){
    if(n == 1) return 0;
    else if(n == 2) return 1;
    return fibo(n - 1) + fibo(n - 2);
}

int main(){
    int n = 5;
    int ans = fibo(n);
    return 0;
}
```

a) 5 b) 6 c) 8 d) 9

28. Vui lòng chỉ ra kết quả trên màn hình của chương trình sau:

```
int cnt = 0;

int foo(int n){
    if(n == 0) return 0;
    return n % 10 + foo(n / 10);
}

int my_function(int n, int sm){
```

```
    int i, tmp_sm;
    for(i = 1; i <= n; i++){
        tmp_sm = foo(i);
        if(tmp_sm == sm)
            cnt++;
    }
    return cnt;
}

int main(){
    int n = 50, sum = 5;
    int ans = my_function(n, sum);
    cout << ans;
    return 0;
}
```

a) 4 b) 5 c) 6 d) 7

29. kết quả của func(5).

```
int fund(int a){
    return (a == 0)?0:1+fund(a-1);
}
int func(int a){
    if(a== 1) return 1;
    return fund(a) * func(a-1);
}
```

a) 720 b) 21 c) 120 d) 20

30. Kết quả của hàm tính tổng các số nguyên tố nhỏ hơn hoặc bằng 5 là:

```
bool isPrime(int n) {
    if (n <= 1) return false;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return false;
    }
    return true;
}

int sumPrimes(int n) {
    if (n < 2) return 0; // Trường hợp cơ bản
    return (isPrime(n) ? n : 0) + sumPrimes(n - 1); // Quy tắc đệ quy
}
```

a) 10 b) 12 c) 15 d) 17

31. Kết quả tối đa lợi nhuận có thể đạt được bằng cách cắt dây có độ dài 5 và giá trị cho các đoạn dây là [2, 5, 9, 6, 10] là:

```
int cutRod(int price[], int n) {
    if (n <= 0) return 0; // Trường hợp cơ bản
    int max_val = INT_MIN; // Khởi tạo giá trị tối đa
    for (int i = 0; i < n; i++) {
        max_val = max(max_val, price[i] + cutRod(price, n - i - 1)); // Quy tắc đệ quy
    }
    return max_val;
}
```

a) 10 b) 12 c) 14 d) 20

32. Kết quả của hàm tìm kiếm phần tử đầu tiên trong mảng với giá trị 3 là:

```
int someFunction(int arr[], int n, int x, int index) {
    if (index >= n) return -1; // Trường hợp cơ bản
    if (arr[index] == x) return index; // Tìm thấy
    return someFunction(arr, n, x, index + 1); // Quy tắc đệ quy
}

int main() {
    int arr[] = {1, 2, 3, 4, 3, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 3;
    int result = someFunction(arr, n, x, 0);
    return 0;
}
```

a) 2 b) 4 c) -1 d) 3

33. Kết quả của hàm đếm số lượng phần tử với giá trị 3 trong mảng là:

```
int anotherFunction(int arr[], int n, int x) {
    if (n <= 0) return 0; // Trường hợp cơ bản
    return (arr[n-1] == x ? 1 : 0) + anotherFunction(arr, n-1, x); // Quy tắc đệ quy
}

int main() {
    int arr[] = {1, 2, 3, 4, 3, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 3;
    int result = anotherFunction(arr, n, x);
    return 0;
}
```

a) 1 b) 2 c) 3 d) 4