# Cấu trúc dữ liệu

## Ôn tập

**Câu 1:** Xác định tính đúng sai của các mệnh đề sau

1. Tìm kiếm tuần tự yêu cầu các phần tử phải được sắp xếp theo chiều tăng dần → **Sai**
2. Tìm kiếm nhị phân yêu cầu các phần tử phải được sắp xếp → **Đúng**
3. Tìm kiếm nhị phân sẽ nhanh hơn với các phần tử được sắp xếp và chậm hơn với các phần tử không được sắp xếp → **Sai**
4. Tìm kiếm nhị phân sẽ nhanh hơn với tập dữ liệu lớn nhưng tìm kiếm tuần tự sẽ nhanh hơn với tập dữ liệu nhỏ → **Sai**
5. Tìm kiếm nhị phân sẽ nhanh hơn tìm kiếm tuần tự với → **Đúng**
6. Để tìm kiếm một mảng có 1024 phần tử, tìm kiếm nhị phân thực hiện tối đa 11 lần lặp và 22 lần so sánh. → **Đúng**
7. Thuật toán selection sort tìm kiếm phần tử nhỏ nhất (hoặc lớn nhất) trong mảng và di chuyển nó về phía cuối (hoặc đầu) của mảng. → **Đúng**
8. Thuật toán quick sort và merge sort thực hiện sắp xếp theo nguyên tắc chia để trị. → **Đúng**

**Câu 2**: Do mảng với các phần tử sau

45, 78, 23, 12, 63, 90, 38, 56, 88, 15

Xác định số lần so sánh (phần tử) khi thực hiện tìm kiếm tuần tự các phần tử sau:

a) 90 → 6 lần

b) 14 → 10 lần

c) 45 → 1 lần

d) 23 → 3 lần

e) 5 → 10 lần

**Câu 3**: Do mảng với các phần tử sau

5, 12, 25, 32, 38, 46, 58, 62, 85, 90, 97, 105, 110

Xác định số lần so sánh (phần tử), giá trị đầu tiên (first), giá trị cuối cùng (last), giá trị ở giữa (middle) của mảng con khi thực hiện tìm kiếm nhị phân các phần tử sau:

a) 32

| Iteration | first | last | mid | list[mid] | No. of comparisons |
|---|---|---|---|---|---|
| 1 | 0 | 12 | 6 | 58 | 2 |
| 2 | 0 | 5 | 2 | 25 | 2 |
| 3 | 3 | 5 | 4 | 38 | 2 |
| 4 | 3 | 3 | 3 | 32 | 1 (found is true) |

b) 20

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 12 | **25** | 32 | 38 | 46 | **58** | 62 | 85 | 90 | 97 | 105 | 110 |

5, 12, 25, 32, 38, 46, **58**, 62, 85, 90, 97, 105, 110

| **Iteration** | **first** | **last** | **mid** | **list[mid]** | **No. of comparisons** |
|---|---|---|---|---|---|
| 1 | 0 | 12 | 6 | 58 | 2 |
| 2 | 0 | 5 | 2 | 25 | 2 |
| 3 | 0 | 1 | 0 | 5 | 2 |
| 4 | 1 | 1 | 1 | 12 | 2 |
| 5 | 2 | 1 | the loop stops, unsuccessful search | | |

c) 105

| **Iteration** | **first** | **last** | **mid** | **list[mid]** | **No. of comparisons** |
|---|---|---|---|---|---|
| 1 | 0 | 12 | 6 | 58 | 2 |
| 2 | 7 | 12 | 9 | 90 | 2 |
| 3 | 10 | 12 | 11 | 105 | 1 (found is true) |

d) 60

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 12 | 25 | 32 | 38 | 46 | **58** | 62 | 85 | 90 | 97 | 105 | 110 |

| **Iteration** | **first** | **last** | **mid** | **list[mid]** | **No. of comparisons** |
|---|---|---|---|---|---|
| 1 | 0 | 12 | 6 | 58 | 2 |
| 2 | 7 | 12 | 9 | 90 | 2 |
| 3 | 7 | 8 | 7 | 62 | 2 |
| 4 | 7 | 6 | the loop stops, unsuccessful search | | |

**Câu 4:** Giả sử mảng M được sắp xếp với 4096 phần tử. Số lần so sánh tối đa khi sử dụng tìm kiếm nhị phân để tìm kiếm một phần tử bao nhiêu?

→ 26 lần

**Câu 5:** Giả sử mảng M được sắp xếp với 500 phần tử. Sử dụng tìm kiếm nhị phân, xác định số lần so sánh trong trường hợp xấu nhất.

→ 18 lần

* Tính tay

501==250, 501<250,2 lần so sánh

501==375, 501<375,4 lần so sánh

501==438, 501<438,6 lần so sánh

501==469, 501<469,8 lần so sánh

501==485, 501<485,10 lần so sánh

501==493, 501<493,12 lần so sánh

501==497, 501<497,14 lần so sánh

501==499, 501<499,16 lần so sánh

501==500, 501<500,18 lần so sánh

* Áp dụng công thức: $2\log_2(n) + 2 = 2*[8.96] + 2 = 2*8+2 = 18$

* Lưu ý nếu tính 2*[8.96] + 2 = 19.92 = 19 là không đúng vì 8.96 cần được làm tròn thành số nguyên trước khi thực hiện tính toán tiếp tục.

**Câu 6:** Cho mảng sau

8, 14, 40, 52, 60, 65, 2, 90, 23

2. 8, 14, 40, 52, 60, 65, 90, 23

Giả sử mảng đã sắp xếp được 6 phần tử đầu tiên

a) Xác định số lần so sánh để sắp xếp 2 vào đúng vị trí của nó bằng insertion sort. → 6 lần
b) Sau khi 2 đã được sắp xếp, xác định số lần so sánh để sắp xếp 23 vào đúng vị trí của nó. → 6 lần

**Câu 7**: Quick sort và merge sort là hai thuật toán sắp xếp dựa trên nguyên tắc chia để trị. Giải thích sự khác nhau giữa hai thuật toán trên.

- Thuật toán quick sort chia mảng theo phần tử pivot. Sau khi phân chia mảng con bên trái nhỏ hơn giá tị pivot và mảng con bên phải lớn hơn giá trị pivot.

- Thuật toán merge sort chia đôi mảng thành hai mảng con (xấp xỉ) bằng nhau.

**Câu 8**: Cho mảng sau

80, 30, 78, 55, 48, 70, 35, 60, 90, 8, 15, 75, 10, 62, 58

Sử dụng thuật toán quick sort với giá trị pivot là phần tử ở giữa mảng trả lời các câu hỏi sau:

a) Giá trị pivot sau khi gọi hàm partition đầu tiên là bao nhiêu? → 60

b) Xác định mảng sau khi gọi hàm partition

30 55 48 35 58 8 15 10 60 70 80 75 78 62 90

c) Xác định số lần gán (assignment) sau khi gọi hàm partition lần đầu tiên → 30 lần

80 30 78 55 48 70 35 **58** 90 8 15 75 10 62 **60** (lần 1)

Swap: 30  80 (lần 2)

30 80 78 55 48 70 35 58 90 8 15 75 10 62 60

Swap: 55  80 (lần 3)

30 55 78 80 48 70 35 58 90 8 15 75 10 62 60

Swap: 48  78 (lần 4)

30 55 48 80 78 70 35 58 90 8 15 75 10 62 60

Swap: 35 80 (lần 5)

30 55 48 35 78 70 80 58 90 8 15 75 10 62 60

Swap: 58 78 (lần 6)

30 55 48 35 58 70 80 78 90 8 15 75 10 62 60

Swap: 8 70 (lần 7)

30 55 48 35 58 8 80 78 90 70 15 75 10 62 60

Swap: 15 80 (lần 8)

30 55 48 35 58 8 15 78 90 70 80 75 10 62 60

Swap: 10 78 (lần 9)

30 55 48 35 58 8 15 10 90 70 80 75 78 62 60

30 55 48 35 58 8 15 10 **60** 70 80 75 78 62 **90** (lần 10)

**Câu 9**: Giả sử danh sách liên kết được cấu trúc ở dạng info-link với kiểu dữ liệu là int. Danh sách liên kết được tạo bằng đoạn code sau:

```
ptr = new nodeType;
ptr->info = 16;
list = new nodeType;
list->info = 25;
list->link = ptr;
ptr = new nodeType;
ptr->info = 12;
ptr->link = nullptr;
list->link->link = ptr;
```

Sử dụng thêm con trỏ khi cần thiết và thực hiện các yêu cầu sau:

a) Con trỏ nào trỏ tới nút đầu tiên trong DSLK? list

b) Xác định thứ tự của các phần tử trong DSLK. 25 16 12

c) Viết code C++ khởi tạo và chèn nút có info là 45 sau nút có info là 16.

```
nodeType *q;
ptr = list->link;
q = new nodeType;
q->info = 45;
q->link = ptr->link;
ptr->link = q;
```

d) Viết code C++ khởi tạo và chèn nút có info là 58 trước nút có info là 25.

```
nodeType *q;
q = new nodeType;
q->info = 58;
q->link = list;
list = q;
```

**Câu 10:** Xác định tính đúng sai của các mệnh đề sau

1. Khi biểu diễn ngăn xếp dưới dạng mảng, con trỏ top chứa địa chỉ của phần tử ở đỉnh của ngăn xếp. → **Sai**
2. Khi biểu diễn ngăn xếp dưới dạng DSLK, hàm isFull() trả về giá trị true nếu ngăn xếp đã đầy. → **Sai**
3. Khi biểu diễn hàng đợi dưới dạng mảng, giá trị con trỏ front và rear chỉ bằng nhau khi hàng đợi rỗng. → **Sai**

**CÂU 11**: Cho đoạn code sau

```
#include <iostream>
#include <stack>
using namespace std;
int main()
{
```

```
    stack<int> stack;
    long long num;
    int temp;
    int secretNum = 0;
    cout << "Please enter a number: ";
    cin >> num;
    num = abs(num);
    while (num > 0)
    {
        stack.push(num % 100);
        num = num / 100;
    }
    while (!stack.empty())
    {
        temp = stack.top();
        stack.pop();
        secretNum = secretNum + temp;
    }
    cout << "secretNum = " << secretNum  << endl;
}
```

Xác định đầu ra và mục đích của đoạn code trên.

Bắt đầu từ trái qua phải, chương trình chứa 2 chữ số của số được nhập vào từ bàn phím vào ngăn xếp. Sau đó cộng dồn các số có hai chữ số trên.