**Câu 1:** Trong khoa học máy tính, **cấu trúc dữ liệu** duợc hiểu nhu thế nào? Cho ví dụ.

Trả lời:

* Cấu trúc dữ liệu được hiểu là sự tổ chức dữ liệu mà máy mà máy tính có thể xử lý được
* VD: các dữ liệu cơ bản như: họ tên kiểu string, ngày sinh kiểu int, …

**Câu 2:** Trong khoa học máy tính, **giải thuật** duợc hiểu nhu thế nào? Cho ví dụ.

Trả lợi:

* Giải thuật là 1 tập hữu hạn các bước (chỉ thị hay hành động) theo 1 trình tự được xác định rõ ràng nhằm mục đích giải quyết 1 bài toán nào đó.
* VD: Giải thuật tìm nghiệm của phương trình bậc 2 ax2 + bx + c.

**Câu 3:** Tại sao nói CTDL và GT có quan hệ mật thiết với nhau? Liệt kê 1 ví dụ nói về cách thiết kế cấu trúc dữ liệu sẽ ảnh huởng dến giải thuật, giải thích tại sao?

Trả lơi:

Xét tới giải thuật thì phải xét giải thuật đó tác động trên cấu trúc dữ liệu nào.

Xét tới cấu trúc dữ liệu thì phải hiểu cấu trúc dữ liệu đó cần được tác động bằng giải thuật gì để được kết quả mong muốn.

Cấu trúc dữ liệu nào thì giải thuật đó. Khi cấu trúc dữ liệu thay đổi giải thuật cũng thay đổi theo.

Mối quan hệ giữa cấu trúc dữ liệu và giải thuật được Niklaus Wirth tổng kết như sau:

Cấu trúc dữ liệu + Giải thuật = Chương trình

VD:

Giải thuật tìm kiếm:

* Nếu cấu trúc danh sách là danh sách đặc thì ta có giải thuật tìm kiếm như tìm kiếm tuần tự, tìm kiếm nhị phân.
* Nếu Nếu cấu trúc danh sách là danh sách liên kết thì ta có giải thuật tìm kiếm sử dụng cây nhị phân.

Giải thích:

* Vì đối với danh sách đặc thì ta có thể duyệt 1 cách dễ dàng với vòng lặp for. Còn với danh sách liên kết thì trong mỗi node đều có lưu trữ địa chỉ của các node khác nên ta không thể duyệt 1 cách dễ dàng như dùng vòng lặp for được.

**Câu 4:** Ðếm số **phép so sánh** trong giải thuật ở ví dụ 1.12.

Trả lời:

Với giải thuật này ta gọi a[j – 1] > xlà phép toán cơ bản

Với i = 1 thì j chạy 1 lần -> a[j – 1] > x chạy 1 lần

Với i = 2 thì j chạy 1 lần -> a[j – 1] > x chạy 2 lần

Với i = 3 thì j chạy 1 lần -> a[j – 1] > x chạy 3 lần

…

Với i = n - 1 thì j chạy 1 lần -> a[j – 1] > x chạy n - 1 lần

* T(n) = (n – 1) + (n - 2) + (n – 3) + … + 2 + 1

⬄T(n) = n2 / 2 + n / 2 ~ O(n2)