BÀI TẬP LÝ THUYẾT CHƯƠNG 1

Câu 1: Trong khoa học máy tính, cấu trúc dữ liệu được hiểu như thế nào? Cho ví dụ

Cấu Trúc dữ liệu(Sự Tổ chức ) của dữ liệu/ Thông tin lên trê máy tính, mà ở đó với cấu trúc này máy tính có thể xử lý được.

Cấu trúc này phải rõ rang, xác định , các thành phàn bên trong cấu trức cũng phải rõ rang, Và xác định

Ví dụ:

Cấu Trúc dữ liệu cơ Bản của một kho hàng (Tên Hàng , mã hàng, Giá Tiền, Tồn Kho)

**Trong Đó**

-Mã hang , Tên Hàng có Kiểu dữ liệu là Chuỗi

-Giá Tiền, tồn Kho có kiểu là Số Thực.

Câu 2: Trong khoa học máy tính, giải thuật được hiểu như thế nào? Cho ví dụ.

Giải Thuật là một Tập hữu hạn của các bước (chỉ thị hay hành động) Theo một trình tự, được xác định rõ rang nhằm mục đích để giải quyết một bài toán nào đó (dựa vào những giá trị đầu vào gọi là “InPut” và Cho kết quả đầu ra gọi là “OutPut”)

Ví Dụ:

Tìm X Trong Phương Trình ax + b = 0(\*)

**Giải Thuật:**

Đầu Vào (**input**): a,b.

Đầu Ra (**output**): Kết Luận Nghiệm.

Bước 1:

Chuyển **b** qua vế phải.(ax = -b)

Bước 2:

Chia hai Vế Cho **a**  (x=-b/a)

Kết luận:

Phương Trình (\*) Có Nghiệm Duy nhất **x= -b/a.**

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 hưởng đến giải thuật, giải thích tại sao?

**Giải thuật** là một hệ thống chặt chẽ và rõ ràng các qui tắc nhằm xác định 1 dãy các thao tác trên những đối tượng, sao cho sau 1 số bước hữu hạn thực hiện các thao tác đó ta thu được kết quả mong muốn.

**Cấu trúc dữ liệu**: là cách tổ chức, lưu trữ dữ liệu trong MTDT 1 cách có thứ tự, có hệ thống nhằm sử dụng dữ liệu 1 cách hiệu quả Ctdl và gt có mối liên hệ chặt chẽ với nhau, chúng luôn tồn tại song song đi kèm nhau theo công thức:

**Cấu Trúc Dữ Liệu + Chương Trình = Giải Thuật**

Bản thân các phần tử của dữ liệu thường có mối quan hệ với nhau, ngoài ra nếu biết tổ chức chúng theo các cấu trúc dữ liệu thích hợp thì việc thực hiện các phép xử lý trên các dữ liệu sẽ càng thuận lợi hơn, đạt hiệu quả cao hơn. Với 1 ctdl đã chọn ta sẽ có giải thuật xử lý tương ứng. Ctdl thay đổi thì giải thuật cũng thay đổi theo. Để có 1 ctrinh tốt, ta cần phải chọn được ctdl phù hợp và chọn được 1 gt đúng đắn Vd: Giả sử ta có 1 danh sách các trường đại học và cao đẳng trên cả nước mỗi trường có các thông tin sau: Tên trường, địa chỉ, sđt phòng đào tạo. Ta muốn viết một chương trình trên máy tính điện tử để khi cho biết “tên trường” máy sẽ hiện ra màn hình cho ta: “địa chỉ” và “số điện thoại phòng đào tạo” của trường đó cách đơn giản là cứ duyệt tuần tự các tên trường trong dnah sách cho tới khi tìm thấy trên trường cần tìm thì sẽ đói chiếu ra “địa chỉ” và “số điện thoại phòng đào tạo” của trường đó. Cách tìm tuần tự này rõ ràng chỉ chấp nhận được khi danh sách ngắn còn danh sách dài thì rất mất thời gian.Nếu ta biết tổ chức lại danh sách bằng cách sắp xếp theo thứ tự từ điển của tên trường, thì có thể áp dụng 1 giải thuật tìm kiếm khác tốt hơn, tương tự như ta vẫn thường làm khi tra từ điển. Cách tìm này nhanh hơn cách trên rất nhiều nhưng không thể áp dụng được với dữ liệu chưa được sắp xếp. Nếu lại biết tổ chức thêm 1 bảng mục lục chỉ dẫn theo chữ cái đầu tiên của tên trường, thì khi tìm “địa chỉ” và “số điện thoại phòng đào tạo” của Hvktmm ta sẽ bỏ qua được các tên trường mà chữ cái đầu không phải là “H”.Như vậy giữa cấu trúc dữ liệu và giải thuật có mối quan hệ mật thiết. Có thể coi chúng như hình với bóng, không thể nói gới cái này mà không nhắc tới cái kia.

Câu 4: Đếm số phép so sánh trong giải thuật ở ví dụ 1.12

Số Phép So sánh Trong ví dụ 1.12 Là: **3(n).**

BÀI TẬP THỰC HÀNH CHƯƠNG 1

Bài 1: Đếm số phép toán gán, phép so sánh được thực thi và xác định độ phức tạp trong đoạn code sau:

for (i = 0; i < n; i++)

for (j = 0; j < m; j++)

if (a[ i ][ j ] = = x)

return 1;

return -1;

**\*Số Phép Toán gán : O(n)**

**\*Số Phép So Sánh: O(n)**

**\*Độ Phức tạp: O(n^2)**

Bài 2: Đếm số phép toán gán, phép so sánh được thực thi và xác định độ phức tạp trong đoạn code sau:

sum = 0;

for (i = 0; i < n ; i++)

for (j = 0; j < i ; j++)

sum++;

**\*Số Phép Toán gán : O(n)**

**\*Số Phép So Sánh: O(n)**

**\*Độ Phức tạp: O(n^2)**

Bài 3: Đánh giá độ phức tạp của đoạn code sau:

for (i = 0; i < n; i++)

sum1+=i;

for (i = 0; i < n\*n; i++)

sum2+=i;

**\*Độ Phức tạp: O(n)**

Bài 4: Đánh giá độ phức tạp của hàm tính giai thừa sau:

int GT (int n)

{ if (n == 1)

return 1;

return n\*GT(n-1);

}

**\*Độ Phức tạp: O(n^2)**

Bài 5: Đánh giá độ phức tạp của hàm tính dãy FIBONACCI sau:

int Fibo (int n)

{

if (n <=1)

return n;

return Fibo(n-1) + Fibo(n-2);

}

**\*Độ Phức tạp: O(n^2)**