**Họ & Tên:** *Nguyễn Thị Diễm My*

**MSSV:** *1851050091*

**Nhóm:** *11*

**Lớp:** *IT01*

# BÀI TẬP CHƯƠNG 1

## **Bài tập lý thuyết**

## Câu 1:

* **CTDL** là cấu trúc (sự tổ chức) của dữ liệu/thông tin lên trên máy tính, mà ở đó với cấu trúc này máy tính có thể xử lí được.
* Cấu trúc này phải rõ ràng, xác định, các thành phần bên trong cấu trúc cũng phải rõ ràng và xác định.
* ***Ví dụ:***  Cấu trúc dữ liệu cơ bản của một sinh viên (MSSV, họ và tên, giới tính, ngày sinh, địa chỉ). Trong đó:
  + MSSV, họ và tên, địa chỉ có kiểu dữ liệu là kiểu **chuỗi**.
  + Ngày sinh của sinh viên có kiểu **date**.

## Câu 2:

* **Giải thuật** là một tập hữu hạn của các bước theo một trình tự, được xác định rõ ràng nhầm mục đích để giải quyết một bài toán nào đó (dựa vào những giá trị đầu vào gọi là “**input**” và  cho ra kết quả đẩu ra gọi là “**output**”).
* ***Ví dụ***: Trong kiến thức Toán trung học cơ sở, ta có bài toán: Tìm nghiệm phương trình bậc hai là một ẩn có dạng ax2 + bx + c = 0 (với a, b, c ϵ ℝ; a ≠ 0)

\*\*\*Ta có giải thuật **(T)** để giải bài toán tìm nghiệm cho phương trình ax2 + bx + c = 0 như sau:

**Giải thuật (T):**

Đầu vào (**input)**: a, b, c (a, b, c ϵ ℝ)

Đầu ra (**output):** kết luận nghiệm

## Câu 3:

* **Cấu trúc dữ liệu** và **giải thuật** có mối quan hệ mật thiết với nhau vì:
  + **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.
  + **Cấu trúc dữ liệu** và **giải thuật** có mối liên hệ chặt chẽ với nhau, chúng luôn tồn tại song song đi kèm nhau theo công thức: Cấu trúc dữ liệu + giải thuật = Chương trình.
  + 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 cấu trúc dữ liệu đã chọn ta sẽ có giải thuật xử lý tương ứng. Cấu trúc dữ liệu thay đổi thì giải thuật cũng thay đổi theo. Để có 1 chương trình tốt, ta cần phải chọn được cấu trúc dữ liệu phù hợp và chọn được một giải thuật đúng đắn.
  + ***Ví dụ:*** 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 đạo. 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. Một cách đơn giản là cứ duyệt tuần tự các tên trường trong danh sách cho tới khi tìm thấy tên trường cần tìm thì sẽ đối 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 thứ tự từ điển của tên , 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 khi 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 HVBCBT ta sẽ bỏ qua được các tên trường mà chữ cái đầu tiên không phải là “H”.

## Câu 4:

Với i = 1 thì j chạy phép so sánh là 2 lần

Với i = 2 thì j chạy phép so sánh là 4 lần

Với i = n thì j chạy phép so sánh là n lần

* T(n) = n\*n O(n2)

## **Bài tập thực hành**

## Bài 1:

|  |  |
| --- | --- |
| **Phép gán** | |
| + m = 0 thì có 1 phép gán  + m = 1 thì có 2 phép gán  .  .  .  + m thì có m + 1 phép gán | + n = 0 thì có m + 1 (for m) \*0 + 1 gán  + n = 1 thì có m + 1 (for m) \*1 + 1 gán  .  .  .  + n thì có m + 1 (for m)\*n + 1 gán |
| => T(m,n) = (m + 1) \* n + 1  =>  Nếu m = n thì T(n) = (n + 1)\*n + 1                         \*\*thêm phép tương đương vào đây nha ahihi\*\*O(n2) | |

|  |  |
| --- | --- |
| + m = 0 thì có 1 phép so sánh  + m = 1 thì có 3 phép so sánh  .  .  .  + m thì có m + 2 phép so sánh | + n = 0 thì có (2m + 1) \* 0 + 1 so sánh  + n = 1 thì có (2m + 1) \* 1 +  1 so sánh  .  .  .  + n thì có (2m + 1) \* n + 1 so sánh |
| =>  T(m,n) = (2m + 1)\*n + 1  =>  Nếu m = n thì T(n) = (2n + 1) \* n + 1  ⬄ O(n2) | |

## Bài 2:

Với sum=0; ta có 1 phép gán

Với 2 vòng lặp for ta có n phép gán.

Với vòng lặp for thứ 2, ta được i phép so sánh.

Độ phức tạp= O(n2)

## Bài 3:

for (i = 0; i < n; i++) ⬄O(n)

sum1+=i;

for (i = 0; i < n\*n; i++) ⬄O(n)

sum2+=i;

**Vậy:** T(n) = max( O(n), O(n) ) O(n2)

## Bài 4:

T(n) = C1 (khi n = 1)

T(n) = T(n – 1) + C2 (khi n > 1)

T(n) = T (n – 1) + C2

⬄ T(n) = (T(n – 2) + C2) + C2 = T(n – 2) + 2C2

….

⬄(T(n – k) + C2) + (k – 1)C2

Chương trình dừng khi n – k = 1 => k = n – 1

⬄T(n – n – 1) + (n – 1)C2 = T(1) + (n + 1)C2

⬄ C1 + (n – 1)C2

## Bài 5:

Chọn phép cộng Fibo(n - 1) + Fibo(n - 2) làm câu lệnh đặc trưng. T(n) là số lần thực hiện câu lệnh này. Dễ thấy:

T(n)=0 vs n <= 1

T(n)=T(n-1) + T(n-2) vs n > 1

T(n) = T(n-1) + T(n-2) + 1 = \Theta(\phi^n) \qquad \phi = (\sqrt{5} + 1)/2 \simeq 1.618

Số phép cộng cần thiết để thực hiện tính F_n là O(n).

Thuật toán quy hoạch động ở trên có thời gian chạy cỡ O(n^2), không phải O(n).