**Khái niệm cấu trúc dữ liệu (Data Structure)**

Như chúng ta đã biết, từ khi máy tính và phần mềm máy tính ra đời là kéo theo sự xuất hiện của việc lưu trữ dữ liệu thông qua các hệ thống lưu trữ dữ liệu trong các file.

Hiểu theo cách rộng rãi trong doanh nghiệp thì cấu trúc dữ liệu là cách thức tổ chức dữ liệu trên máy vi tính để có thể sử dụng hiệu quả. Hầu như tất cả các ứng dụng doanh nghiệp đều sử dụng một cấu trúc dữ liệu theo những cách khác nhau.

**Trong ngành *Khoa học máy tính*, cấu trúc dữ liệu là một cách lưu dữ liệu trong máy tính sao cho nó có thể được sử dụng một cách hiệu quả. Cụ thể, đó là một cách sắp xếp dữ liệu trên máy tính để nó có thể được truy cập và cập nhật một cách hiệu quả.**

Tùy thuộc vào yêu cầu và dự án của bạn, điều quan trọng là chọn cấu trúc dữ liệu phù hợp cho dự án của bạn. Ví dụ, nếu bạn muốn lưu trữ dữ liệu tuần tự trong bộ nhớ, thì bạn có thể sử dụng cấu trúc dữ liệu mảng.

Ngày nay, các cấu trúc dữ liệu được sử dụng rộng rãi trong hầu hết mọi chương trình hay phần mềm đã được phát triển. Hơn nữa, các cấu trúc dữ liệu đều tuân theo các nguyên tắc cơ bản của *Khoa học máy tính* và *Kỹ thuật phần mềm*. Nó cũng là một chủ đề quan trọng trong các câu hỏi phỏng vấn ở ngành Kỹ thuật phần mềm. Do đó, là một lập trình viên, chúng ta cần phải có kiến thức tốt về các cấu trúc dữ liệu.

Những đặc trưng của một cấu trúc dữ liệu

* Sự chính xác – Cấu trúc dữ liệu thực thi trên một Kiểu dữ liệu trừu tượng đã được khẳng định.
* Sự phức tạp thời gian – Thời gian chạy hay thời gian thực thi của một thao tác trên cấu trúc dữ liểu phải nhỏ và có thể thực hiện được.
* Sự phức tạp khoảng trống bộ nhớ – Bộ nhớ được sử dụng của cấu trúc dữ liệu và các thao tác trên nó phải nhỏ và khả thi.

Sự cần thiết của cấu trúc dữ liệu

Kiến thức về cấu trúc dữ liệu giúp bạn hiểu hoạt động của từng cấu trúc dữ liệu. Dựa vào đó bạn có thể chọn cấu trúc dữ liệu phù hợp cho dự án của mình. Điều này giúp bạn viết mã lệnh liên quan đến bộ nhớ và thời gian hiệu quả hơn.

Các ứng dụng cần khai khác sự phức tạp và dữ liệu phong phú, sau đây có 3 vấn đề mà ứng dụng phải đối mặt hàng ngày:

* Tìm kiếm dữ liệu: Cân nhắc một kho hàng lưu trữ 1 triệu mặt hàng. Nếu mỗi lần tìm kiếm một mặt hàng ứng dụng sẽ phải tìm kiếm trên 1 triệu mặt hàng do đó thời gian tìm kiếm sẽ chậm. Khi dữ liệu lớn lên quá trình tìm kiểm sẽ chậm hơn.
* Tốc độ bộ vi xử lý: Tốc độ bộ vi xử lý mặc dù rất cao nhưng vẫn rơi vào giới hạn nếu dữ liệu lên đến hàng tỷ bản ghi.
* Nhiều yêu cầu: Hàng nghìn người dùng cùng tìm kiếm dữ liệu đồng thời trên máy chủ web, mặc dù máy chủ nhanh thì vẫn lỗi trong khi tìm kiếm dữ liệu.

Để giải quyết những vấn đề trên thì cần phải sử dụng cấu trúc dữ liệu phù hợp. Dữ liệu có thể tổ chức trong một cấu trúc dữ liệu có thể theo cách không phải tất cả dữ liệu đều có thể tìm kiếm và dữ liệu có thể tìm kiếm được thì có ngay lập tức.

Thời gian thao tác với cấu trúc dữ liệu

Có 3 trường hợp thường xảy ra khi so sánh sự khác nhau về thời gian thực thi các thao tác trên cấu trúc dữ liệu có liên quan:

* Trường hợp tồi nhất: Đây là trường hợp đặc biệt khi thao tác trên cấu trúc dữ liệu với thời gian nhiều nhất để thực hiện thao tác này. Nếu một thao tác trong trường hợp tồi nhất là ƒ(n) thì thao tác này sẽ không nhiều hơn ƒ(n) thời gian (ở đây ƒ(n) biểu diễn hàm của n).
* Trường hợp trung bình: Đây là trường hợp mô tả thời gian trung bình thể thao tác trên cấu trúc dữ liệu. Nếu một thao tác mất ƒ(n) thời gian thực thi thì m thao tác sẽ mất m\*ƒ(n) thời gian.
* Trường hợp tốt nhất: Đây là trường hợp mô tả thời gian thực thi tối thiểu để thao tác trên cấu trúc dữ liệu. Nếu một thao tác mất ƒ(n) thời gian thực thi, thì thao tác thực tế có thể mất thời gian là một số ngẫu nhiên với giá trị lớn nhất là ƒ(n).

Các thuật ngữ cơ bản

* **Dữ liệu** (Data) − Dữ liệu là giá trị hoặc tập các giá trị.
* **Mục dữ liệu** (Data Item) − Mục dữ liệu chứa một giá trị đơn.
* **Mục nhóm** (Group Items) − Mục dữ liệu được chia ra thành các mục con được gọi là Mục nhóm.
* **Mục cơ bản** (Elementary Items) − Mục dữ liệu không thể được chia nhỏ hơn nữa thì được gọi là Mục cơ bản.
* **Thuộc tính** (Attribute/Property) và Thực thể (Entity) − Một thực thể sẽ chứa các thuộc tính, các thuộc tính này có thể gán được giá trị.
* **Tập thực thể** (Entity Set) − Các thực thể có chung các thuộc tính được gọi là tập thực thể.
* **Trường** (Field) − Trường là một khối thông tin cơ bản biểu diễn một thuộc tính của một thực thể.
* **Bản ghi** (Record) − Bản ghi là một tập hợp các trường giá trị của một thực thể xác định.
* **Tệp** (File) − Tệp là một tập hợp các bản ghi của các thực thể trong một tập thực thể xác định.

Các loại cấu trúc dữ liệu

Về cơ bản, cấu trúc dữ liệu được chia thành hai loại:

* Cấu trúc dữ liệu tuyến tính: Trong cấu trúc dữ liệu tuyến tính, các phần tử được sắp xếp theo thứ tự lần lượt. Vì các phần tử được sắp xếp theo thứ tự cụ thể nên chúng rất dễ thực hiện. Tuy nhiên, khi độ phức tạp của chương trình tăng lên, cấu trúc dữ liệu tuyến tính có thể không phải là lựa chọn tốt nhất vì sự phức tạp trong hoạt động. Các cấu trúc dữ liệu tuyến tính phổ biến như sau:
  + Cấu trúc dữ liệu Mảng
  + Cấu trúc dữ liệu Ngăn xếp (stack)
  + Cấu trúc dữ liệu Hàng đợi (queue)
* Cấu trúc dữ liệu phi tuyến tính: Không giống như cấu trúc dữ liệu tuyến tính, các phần tử trong cấu trúc dữ liệu phi tuyến tính không nằm trong bất kỳ trình tự nào. Thay vào đó, chúng được sắp xếp theo thứ bậc trong đó một phần tử sẽ được kết nối với một hoặc nhiều phần tử. Cấu trúc dữ liệu phi tuyến tính được chia thành các cấu trúc dữ liệu dựa trên đồ thị và cây. Các cấu trúc dữ liệu phi tuyến tính phổ biến như sau:
  + Cấu trúc dữ liệu cây
  + Cấu trúc dữ liệu đồ thị

Sự khác nhau giữa cấu trúc dữ liệu tuyến tính và phi tuyến tính

|  |  |
| --- | --- |
| **Cấu trúc dữ liệu tuyến tính** | **Cấu trúc dữ liệu phí tuyến tính** |
| Các phần tử dữ liệu được sắp xếp theo thứ tự tuần tự, nối tiếp nhau. | Các phần tử dữ liệu được sắp xếp theo thứ tự không tuần tự (theo cách phân cấp). |
| Tất cả các phần tử đều biểu diễn trên một tầng. | Các phần tử dữ liệu được biểu diễn trên các tầng khác nhau. |
| Cấu trúc dữ liệu này có thể được duyệt qua trong một lần chạy. Có nghĩa là, nếu chúng ta bắt đầu từ phần tử đầu tiên, chúng ta có thể duyệt qua tất cả các phần tử một cách tuần tự trong một lần chuyển. | Cấu trúc dữ liệu này yêu cầu nhiều lần chạy. Có nghĩa là, nếu chúng ta bắt đầu từ phần tử đầu tiên thì có thể không thể duyệt qua tất cả các phần tử trong một lần chuyển. |
| Việc sử dụng bộ nhớ không hiệu quả. | Các cấu trúc khác nhau sử dụng bộ nhớ theo những cách hiệu quả khác nhau tùy thuộc vào nhu cầu. |
| Độ phức tạp về thời gian tăng theo kích thước dữ liệu. | Độ phức tạp về thời gian vẫn như lúc trước khi tăng kích thước dữ liệu. |