**CÂU HỎI LÝ THUYẾT**

**CÂU HỎI 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 là cấu trúc (sự tổ chức) của dữ liệu/ thông tin lên 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 (mã số sinh viên, họ và tên, giới tính, ngày sinh, địa chỉ). Trong đó: mã số sinh viên, họ tên, địa chỉ có kiểu dữ liệu là chuỗi; còn ngày sinh của sinh viên có kiểu Date (kiểu ngày).

**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 ràng nhằm 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 đầu ra gọi là “output”)

Ví dụ: Tìm nghiệm phương trình bậc hai một ẩn có dạng ax2 + bx + c = 0(với: a, b, c ∈ 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,  ℝ)Đầu ra (output): kết luận nghiệm

**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?** Cấu trúc dữ liệu và Giải thuật có mối quan hệ chặt chẽ với nhau vì cấu trúc dữ liệu đóng vai trò quan trọng trong việc kết hợp và đưa ra cách giải quyết bài toán, cấu trúc dữ liệu cũng hỗ trợ cho các thuật toán thao tác, xử lý hiệu quả hơn. Và mối quan hệ được thể hiện qua công thức: Cấu trúc dữ liệu + Giải thuật = Chương trình

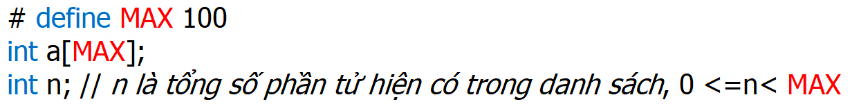
Với một cấu trúc dữ liệu đã chọn, sẽ có những giải thuật tương ứng, phù hợp. Khi cấu trúc dữ liệu thay đổi thường giải thuật cũng phải thay đổi theo để tránh việc xử lý gượng ép, thiếu tự nhiên trên một cấu trúc không phù hợp. Hơn nữa, một cấu trúc dữ liệu tốt sẽ giúp giải thuật xử lý trên đó có thể phát huy tác dụng tốt hơn, vừa đáp ứng nhanh vừa tiết kiệm vật tư, giải thuật cũng dễ hiễu và đơn giản hơn.

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

Có 3 phép so sánh ở vd 1.12 là: i<n, j>0, a[j-1]>X

**CÂU HỎI CHƯƠNG 2**

**Câu 1: Trong khoa học máy tính, danh sách đặc được hiểu như thế nào? Cho ví dụ.** Danh sách đặc là một danh sách mà các phần tử trong danh sách có cùng kiểu dữ liệu, và được cấp phát liên tục trong bộ nhớ.

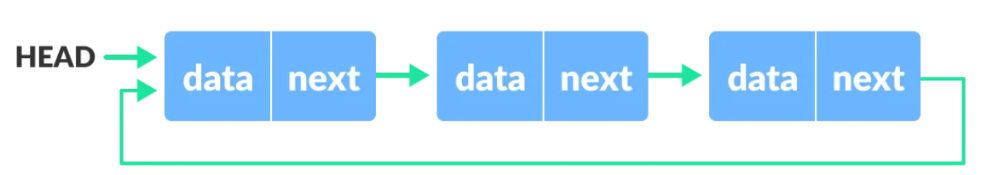
Ví dụ:

**Câu 2: Trong khoa học máy tính, danh sách liên kết được hiểu như thế nào? Có mấy loại? Cho ví dụ.** Trong khoa học máy tính, danh sách liên kết (LINKED LIST) là danh sách mà các phần tử được cấp phát rời rạc, bao gồm một chuỗi các node kết nối với nhau. Mỗi node có thể xem như một phần tử trong danh sách. Mỗi node sẽ lưu trữ dữ liệu (data) của node đó và địa chỉ (address) của node kế tiếp. Danh sách liên kết phải có node bắt đầu, được gọi là HEAD. Node cuối cùng được gọi là NULL. Mỗi node có nhiều nhất 1 phần tử đứng trước cũng như có nhiều nhất 1 phần tử đứng sau.

***Có 3 loại danh sách liên kết:***

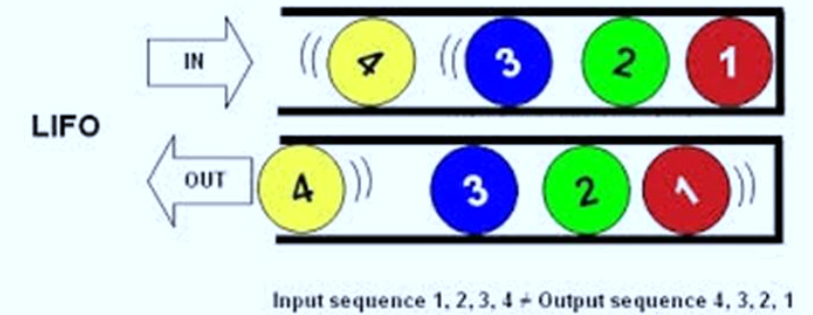
+ Danh sách liên kết đơn (SINGLY LINKED LIST):

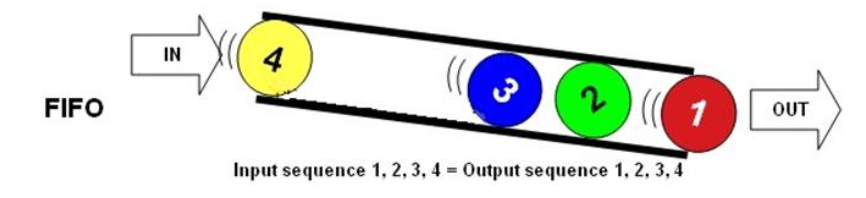
+ Danh sách liên kết kép (DOUBLY LINKED LIST):

+ Danh sách liên kết vòng (CIRCULAR LINKED LIST):

**Câu 3: Tại sao nói STACK và QUEUE là danh sách hạn chế? Cho ví dụ?** Nói Stack và Queue là danh sách hạn chế bởi chúng đều là một danh sách các phần tử được quản lý theo thứ tự.

**Câu 4: Thế nào là LIFO, FIFO? Cho ví dụ.** LIFO (Last in First Out – Vào sau Ra trước): Ở phương pháp này, các hàng hoá được nhập vào kho gần nhất sẽ được xuất ra đầu tiên. Hàng hoá mới được sử dụng trước, dùng ưu tiên hơn hàng hoá cũ.

FIFO (First in First out – Vào trước Ra trước): Với phương pháp này, các lô hàng đầu tiên của hàng hoá nhập vào nhà kho sẽ là hàng hoá đầu tiên được xuất ra khỏi kho – từ đó được gửi vào các cửa hàng hoặc gửi trực tiếp đến khách hàng.

**Câu 6: Theo bạn, danh sách danh sách liên kết có thể ứng dụng xử lý các vấn đề gì trong máy tính?-**Hoạt động chèn: thêm một phần tử vào đầu danh sách liên kết.

-Hoạt động xóa (phần tử đầu): xóa một phần tử tại đầu danh sách liên kết.

-Hiển thị: hiển thị toàn bộ danh sách.

-Hoạt động tìm kiếm: tìm kiếm phần tử bởi sử dụng khóa (key) đã cung cấp.

-Hoạt động xóa (bởi sử dụng khóa): xóa một phần tử bởi sử dụng khóa (key) đã cung cấp.

**Câu 7: Thế nào là cấu trúc dữ liệu động? Cho ví dụ.** Cấu trúc dữ liệu động là những thứ có thể mở rộng hoặc thu lại tùy thuộc vào chương trình, đồng thời các vị trí bộ nhớ liên quan của chúng sẽ có thể thay đổi. Ví dụ: Danh sách liên kết được tạo ra bằng con trỏ.

**CÂU HỎI CHƯƠNG 3**

**Câu 1: Trong các phương pháp xếp thứ tự đã học, phương pháp nào tối ưu nhất, và kém tối ưu nhất? Tại sao?** Trong các phương pháp sắp xếp thứ tự đã học, Phương pháp sắp xếp tối ưu nhất là Buble Sort bởi theo thống kê cho ra thì tốc độ nhanh nhất là O(n) và kém tối ưu nhất là Selection Sort, Selction Sort cho ra tốc dộ chậm hơn trong đa số trường hợp với độ phức tạp là O(n2). Do đó Selection Sort chỉ nên dùng để sắp xếp cho số lượng phần tử không quá nhiều.

**Câu 2: Trong các 2 phương pháp tìm kiếm đã học, trường hợp nào thì cả 02 phương pháp đều như nhau? Giải thích tại sao?** Khi cả 2 phương pháp tìm kiếm là như nhau trong trường hợp tìm kiếm tốt nhất, là khi độ phức tạp bằng 0. Vì với trường hợp tốt nhất, chỉ có 1 lần xét trong cả 2 phương pháp, tìm kiếm tuần tự là ở đầu và tìm kiếm nhị phân là ở giữa.

**Câu 3: Ngoài các phương pháp xếp thứ tự đã học, hãy tìm hiểu thêm một phương pháp xếp thứ tự khác, giới thiệu sơ và giải thích.**

Counting Sort là sắp xếp bằng phương pháp đếm phân phối, có thể rút ngắn thời gian của quá trình thực hiện thuật toán. Được thực hiện bằng việc lập bảng thống kê các giá trị số nguyên.

Có 3 bước thực hiện thuật toán:

Bước 1: Xây dựng mảng dd[i] (dd[i] là số lần xuất hiện của số nguyên i)

Bước 2: Gọi GTMAX là giá trị lớn nhất của số nguyên cần sắp xếp, N là số dương phần tử cần sắp xếp

Bước 3: dd[i] = 0 với i = 0 GTMAX

Bước 4: Duyệt các số cần sắp xếp với mỗi số có giá trị I thực hiện đếm dd[i]=dd[i]+1

Bước 5: Kết thúc và sử dụng theo yêu cầu

**CÂU HỎI CHƯƠNG 4**

**Câu 1: Hãy trình bày các vấn đề sau: Định nghĩa và đặc điểm của cây nhị phân tìm kiếm; Các thao tác thực hiện tốt trong kiểu này; Hạn chế của kiểu CTDL này?**

Cây nhị phân tìm kiếm là cây nhị phân mà giá trị (khóa) của phần tử bên trái của một node có giá trị nhỏ hơn giá trị (khóa) của node, giá trị (khóa) của các phần tử bên phải của một node thì lớn hơn giá trị (khóa) của node đó.

**Đặc điểm:**

- Cấu trúc cây nhị phân tìm kiếm quản lý một tập các phần tử có số lượng khá lớn, được cấp phát rời rạc trong bộ nhớ. - Có khả năng tìm kiếm nhanh, do tính chất, giá trị của một node sẽ lớn hơn các giá trị bên nhánh con bên trái và nhỏ hơn các giá trị của nhánh con bên phải.

**Các thao tác:**

+ Tìm một node trên cây nhị phân tìm kiếm+ Thêm một node mới vào cây+ Duyệt cây nhị phân tìm kiếm+ Xóa một node trên cây

Chiều sâu:

- LNR ( Left - Node - Right)

- LRN ( Left - Right - Node)

- NLR ( Node - Left - Right)

Chiều rộng:

- Level order traversal

**Hạn chế:** Việc thêm/xóa node trong cây khá phức tạp, do phải thực hiện nhiều phép so sánh. Mỗi node của cay chỉ tối đa 2 con

**Câu 2: Hãy so sánh cây nhị phân tìm kiếm và các CTDL cơ bản: danh sách đặc, danh sách liên kết, danh sách hạn chế.**

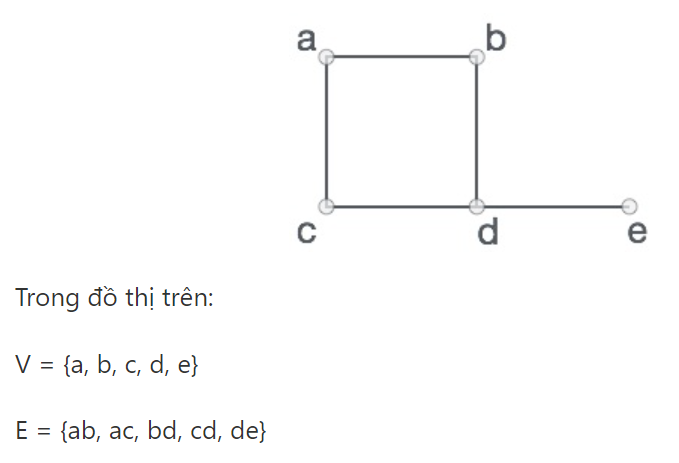
|  |  |  |  |
| --- | --- | --- | --- |
| **Cây Nhị Phân Tìm Kiếm** | **Danh Sách Đặc** | **Danh Sách Liên Kết** | **Danh Sách Hạn Chế** |
| Là một tập các phần tử có số lượng khá lớn | Là một danh sách mà các phần tử trong danh sách có cùng kiểu dữ liệu và bị hạn chế số lượng phần tử. | Là một danh sách mà các phần tử được cấp phát rời rạc nhau, và cố định trong bộ nhớ. | Là một danh sách các phần tử được quản lý theo thứ tự như sau: Phần tử được thêm vào ngăn xếp sau, sẽ được lấy ra (xóa) khỏi ngăn xếp trước |
| Được cấp phát rời rạc trong bộ nhớ. | Được cấp phát liên tục trong bộ nhớ. | Được cấp phát rời rạc nhau, và cố định trong bộ nhớ |  |
| Có khả năng tìm kiếm nhanh | Tìm kiếm chậm hơn | Tìm kiếm chậm hơn | Tìm kiếm chậm hơn |
| Cấu trúc khá phức tạp | Cấu trúc đơn giản | Cấu trúc đơn giản | Cấu trúc đơn giản |

**CÂU HỎI CHƯƠNG 5**

**Câu 1: Đồ thị là gì? Cho ví dụ? Có các loại đồ thị gì? Ngoài những đồ thị đã học, hãy tìm hiểu thêm còn có những loại đồ thị nào?**

Một đồ thị (đồ thị) là một dạng biểu diễn hình ảnh của một tập các đối tượng, trong đó các cặp đối tượng được kết nối bởi các link. Các đối tượng được nối liền nhau được biểu diễn bởi các điểm được gọi là các đỉnh (vertices), và các link mà kết nối các đỉnh với nhau được gọi là các cạnh (edges).

Có 2 loại đồ thị: Đồ thị có trọng số và đồ thị có hướng.

-Một số loại đồ thị khác:

-Đồ thị đơn giản (Simple graph)

-Đa đồ thị (Multiple graph)

-Đồ thị có hướng (Directed Graph)

-Đồ thị có hướng có trọng số (Weighted Directed Graph)

-Đồ thị hỗn hợp (Mixed Graph)

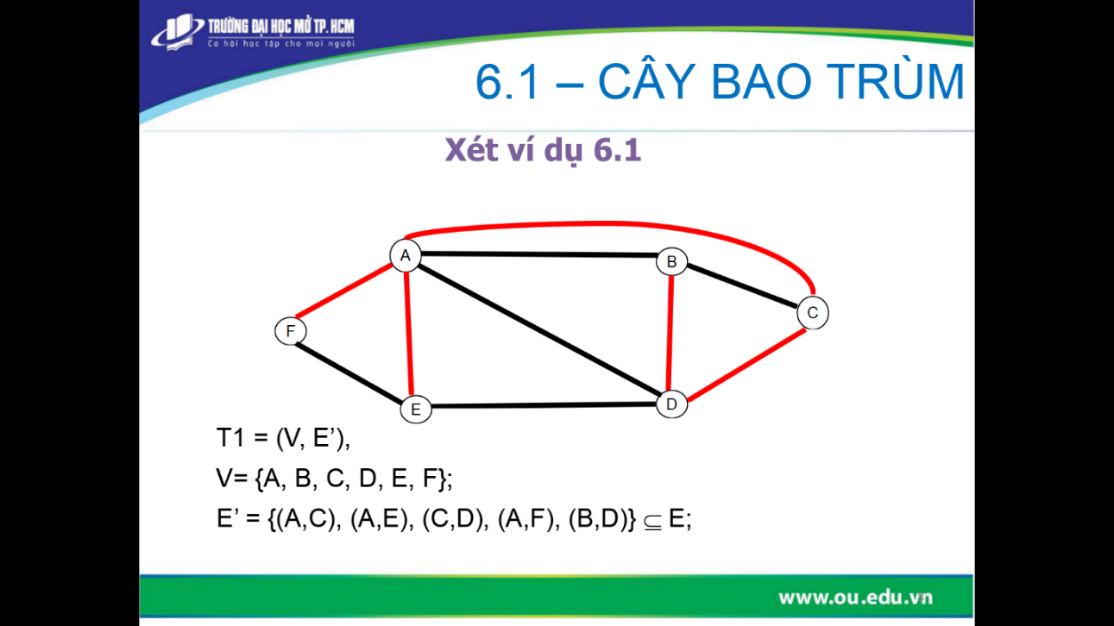
**Câu 2: Để biểu diễn đồ thị trên máy tính, ta có mấy cách? -**Có 2 cách: Ma trận kề và danh sách kề

**CÂU HỎI CHƯƠNG 6**

**Câu 1: Cây bao trùm là gì? Cho ví dụ? Cây bao trùm tối tiểu là gì?**

-Cho đồ thị liên thông G = (V, E), V là tập đỉnh, E là tập cạnh của G. Nếu T = (V, E’), với E’ E, và T là một cây (có nghĩa T không có chu trình)

=> Thì ta nói T là cây bao trùm

Ví dụ:

\*Cây bao trùm tối tiểu là 1 cây bao trùm, có tổng trọng số là tối tiểu trên tập các cây khung Sp(G);

**Câu 2: Đồ thị và cây bao trùm có những điểm giống và khác nhau như thế nào?**

|  |  |  |
| --- | --- | --- |
| **Cơ sở để so sánh** | **Cây** | **Đồ thị** |
| Con đường | Chỉ một giữa hai đỉnh. | Nhiều hơn một đường dẫn được phép. |
| Nút gốc | Nó có chính xác một nút gốc. | Biểu đồ không có nút gốc. |
| Vòng lặp | Không có vòng lặp nào được phép. | Đồ thị có thể có các vòng lặp. |
| Phức tạp | Ít phức tạp hơn | So sánh phức tạp hơn |
| Kỹ thuật truyền tải | Đặt hàng trước, Đặt hàng và Sau khi đặt hàng. | Tìm kiếm theo chiều rộng và tìm kiếm theo chiều sâu. |
| Số cạnh | n-1 (với n là số nút) | Không xác định |
| Loại mô hình | Thứ bậc | Mạng lưới |