



HỒ SĨ ĐÀM (Tổng Chủ biên) – ĐỖ PHAN THUẬN (Chủ biên)
ĐỖ ĐỨC ĐÔNG – NGUYỄN KHÁNH PHƯƠNG

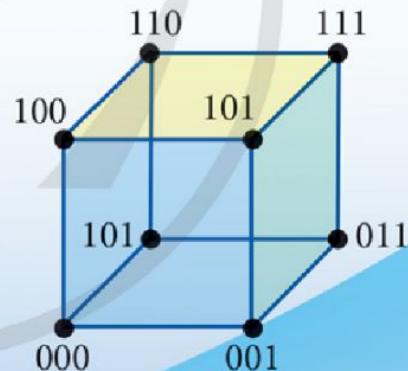
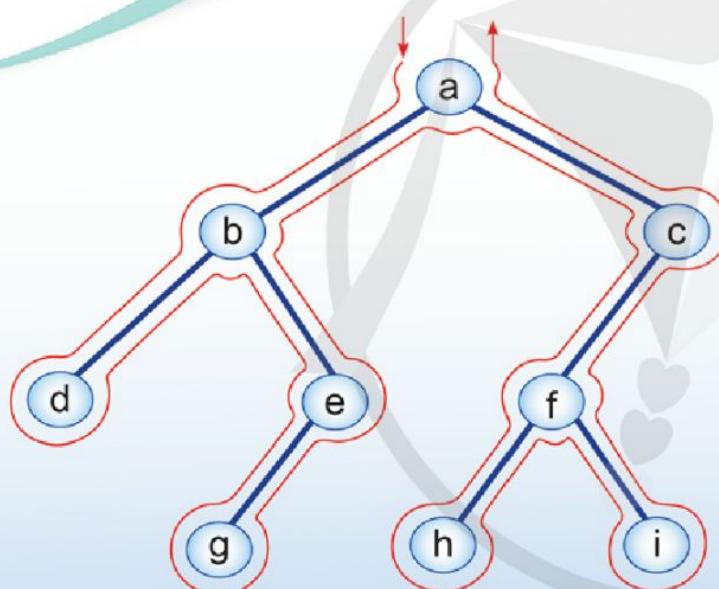
CHUYÊN ĐỀ HỌC TẬP

Tin học

KHOA HỌC
MÁY TÍNH

12

BẢN MẪU



NHÀ XUẤT BẢN ĐẠI HỌC SƯ PHẠM



CÔNG TY CỔ PHẦN ĐẦU TƯ
XUẤT BẢN - THIẾT BỊ GIÁO DỤC VIỆT NAM

Bản in thử

HỒ SĨ ĐÀM (Tổng Chủ biên) – ĐỖ PHAN THUẬN (Chủ biên)
ĐỖ ĐỨC ĐÔNG – NGUYỄN KHÁNH PHƯƠNG

CHUYÊN ĐỀ HỌC TẬP

Tin học

KHOA HỌC
MÁY TÍNH

12



NHÀ XUẤT BẢN ĐẠI HỌC SƯ PHẠM

CÔNG TY CỔ PHẦN ĐẦU TƯ
XUẤT BẢN - THIẾT BỊ GIÁO DỤC VIỆT NAM

Bản in thử



Sách giáo khoa được thẩm định bởi Hội đồng quốc gia thẩm định sách giáo khoa lớp 12
(Theo Quyết định số 1882/QĐ-BGDĐT ngày 29 tháng 6 năm 2023
của Bộ trưởng Bộ Giáo dục và Đào tạo)

CÁC CHUYÊN ĐỀ

Chuyên đề 1

Tìm hiểu một vài kiểu dữ liệu tuyến tính

Chuyên đề 2

Tìm hiểu Cây tìm kiếm nhị phân trong sắp xếp và tìm kiếm

Chuyên đề 3

Tìm hiểu kĩ thuật duyệt Đồ thị và ứng dụng

KÍ HIỆU DÙNG TRONG SÁCH



Khởi động



Hoạt động



Luyện tập



Vận dụng



Câu hỏi tự kiểm tra

Các em giữ gìn sách cẩn thận, không viết vào sách để sử dụng được lâu dài.



LỜI NÓI ĐẦU

Các em học sinh thân mến!

Quyển sách *Chuyên đề học tập Tin học 12 – Khoa học máy tính* sẽ hướng dẫn các em thực hành ứng dụng một số dạng cấu trúc dữ liệu thiết yếu trong Tin học. Điều đó giúp các em phát triển khả năng áp dụng phong phú các kiểu cấu trúc dữ liệu vào giải quyết vấn đề. Cụm chuyên đề với 35 tiết học gồm ba chuyên đề: *Tìm hiểu một vài kiểu dữ liệu tuyến tính*; *Tìm hiểu Cây tìm kiếm nhị phân trong sắp xếp và tìm kiếm*; *Tìm hiểu kỹ thuật duyệt Đồ thị và ứng dụng*. Mỗi chuyên đề đều chú trọng vào việc thực hành và hoàn thiện sản phẩm. Ở cuối mỗi chuyên đề, có một bài làm việc theo nhóm. Cuối Chuyên đề 1 là *Dự án học tập: Xây dựng chương trình sử dụng kiểu dữ liệu hàng đợi và ngăn xếp*. Cuối Chuyên đề 2 là *Thực hành tổng hợp: Ứng dụng cây tìm kiếm nhị phân giải quyết hoàn chỉnh một bài toán thực tế theo hướng dẫn chi tiết*. Cuối Chuyên đề 3 là *Dự án học tập: Tìm hiểu các vấn đề ứng dụng đồ thị*. Các bài làm việc theo nhóm này giúp các em chủ động khám phá và tìm hiểu xây dựng chương trình giải quyết các bài toán thú vị trong thực tế ứng dụng các kiến thức được học trong chuyên đề.

Các hoạt động: *Khởi động*; *Hoạt động* kiến tạo kiến thức, kĩ năng; *Luyện tập*; *Vận dụng* và *Câu hỏi tự kiểm tra* trong quyển sách này được thiết kế kĩ lưỡng và sư phạm. Các chuyên đề góp phần giúp các em có nhiều cơ hội hình thành và phát triển năng lực tin học. Các nội dung thực hành giúp các em rèn luyện kĩ năng làm việc nhóm, khả năng khám phá, giải quyết vấn đề và sáng tạo.

Chúc các em hoàn thành tốt các nội dung học tập và tìm được nhiều điều thú vị trong quyển sách này.

Các tác giả

MỤC LỤC

Nội dung	Trang
CHUYÊN ĐỀ 1. TÌM HIỂU MỘT VÀI KIỂU DỮ LIỆU TUYẾN TÍNH	5
Bài 1. Kiểu dữ liệu hàng đợi	5
Bài 2. Kiểu dữ liệu ngăn xếp	12
Bài 3. Thực hành kiểu dữ liệu hàng đợi và ngăn xếp	18
Bài 4. Dự án học tập: Xây dựng chương trình sử dụng kiểu dữ liệu hang đợi và ngăn xếp	21
CHUYÊN ĐỀ 2. TÌM HIỂU CÂY TÌM KIẾM NHỊ PHÂN TRONG SẮP XẾP VÀ TÌM KIẾM	29
Bài 1. Giới thiệu cây nhị phân	29
Bài 2. Thực hành duyệt cây nhị phân	37
Bài 3. Cây tìm kiếm nhị phân	43
Bài 4. Thực hành tổng hợp: Ứng dụng cây tìm kiếm nhị phân	49
CHUYÊN ĐỀ 3. TÌM HIỂU KỸ THUẬT DUYỆT ĐỒ THỊ VÀ ỨNG DỤNG	53
Bài 1. Đồ thị, phân loại đồ thị	53
Bài 2. Biểu diễn đồ thị trên máy tính	57
Bài 3. Thực hành các thao tác cơ bản với đồ thị trên máy tính	60
Bài 4. Duyệt đồ thị	62
Bài 5. Thực hành duyệt đồ thị	69
Bài 6. Dự án học tập: Tìm hiểu các vấn đề ứng dụng đồ thị	72
Bảng giải thích thuật ngữ	75

TÌM HIỂU MỘT VÀI KIỂU DỮ LIỆU TUYẾN TÍNH

Bài 1

KIỂU DỮ LIỆU HÀNG ĐỢI

Học xong bài này, em sẽ:

- Biết được kiểu dữ liệu hàng đợi là kiểu dữ liệu tuyến tính.
- Mô tả được khái niệm hàng đợi và cơ chế hoạt động của nó.
- Biểu diễn được hàng đợi bằng mảng một chiều và viết được chương trình con thực hiện các phép toán cơ bản trên hàng đợi.
- Nêu được một số ứng dụng của hàng đợi.



Một phòng máy thực hành có 50 máy tính nối mạng với một máy in duy nhất (*Hình 1*). Có nhiều người dùng trong phòng đều có nhu cầu in tệp dữ liệu của họ. Theo em, các tệp dữ liệu đó sẽ được in theo thứ tự thế nào và hệ thống phải sắp xếp các tệp dữ liệu này như thế nào để làm được điều đó?



Hình 1. Một ví dụ về nhiều máy tính cùng kết nối đến một máy in

1) Một số ví dụ về hàng đợi và cơ chế hoạt động

Xếp hàng đợi đến lượt là điều diễn ra hằng ngày xung quanh chúng ta. Ví dụ: xếp hàng đợi thanh toán ở siêu thị (*Hình 2*), ở quầy bán vé xem phim; xếp hàng đợi để lên máy bay, được bác sĩ thăm khám,... Ở một số điểm giao dịch tại ngân hàng hay văn phòng công chứng, việc xếp hàng thông qua lấy số thứ tự, ai đến trước nhận số thứ tự được phục vụ trước.



Hình 2. Ví dụ xếp hàng đợi thanh toán ở siêu thị

Việc tạo ra hàng đợi trong các ví dụ ở trên là để đảm bảo công bằng, tránh tình trạng người đến trước lại bị phục vụ sau và ngược lại. Do đó, cơ chế hoạt động của hàng đợi tuân theo nguyên tắc người đến trước xếp hàng trước, người đến sau nối tiếp theo sau một cách lần lượt. Người đang đứng đầu hàng sẽ được phục vụ đầu tiên, tức là sẽ được ra khỏi hàng đợi đầu tiên. Vì vậy, cơ chế hoạt động của hàng đợi là vào trước ra trước hay còn gọi là FIFO (viết tắt của *First In First Out* trong tiếng Anh).

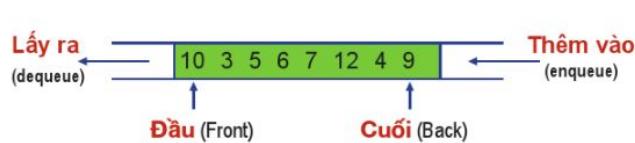
Trong tin học cũng có nhiều ứng dụng sử dụng hàng đợi. Ví dụ: Khi dùng phần mềm nhắn tin trên mạng, nếu tại thời điểm ta gửi tin nhắn mà người nhận đang không kết nối Internet thì những tin nhắn này sẽ được “xếp hàng” vào một hàng đợi, tin nào được gửi trước thì sẽ vào hàng đợi trước. Khi người nhận kết nối Internet, những tin nhắn này sẽ lần lượt được lấy ra khỏi hàng đợi và gửi đến người nhận. Bằng cách sử dụng hàng đợi như vậy sẽ đảm bảo bên nhận nhìn thấy các tin nhắn này theo đúng thứ tự mà chúng đã được gửi đến.

2) Kiểu dữ liệu hàng đợi và các phép toán cơ bản trên hàng đợi

Trong tin học, kiểu dữ liệu hàng đợi (gọi là *Queue*) được xây dựng để mô phỏng cơ chế hoạt động FIFO của các hàng đợi trong thực tế. Hàng đợi thuộc kiểu dữ liệu tuyến tính, các phần tử được sắp xếp một cách lần lượt từ đầu đến cuối, hết phần tử này đến phần tử khác. Phép toán thêm vào (*enqueue*) thực hiện chỉ ở một phía gọi là *phía cuối* (*Back* hay *Rear*), phép toán lấy ra (*dequeue*) chỉ thực hiện ở phía còn lại gọi là *phía đầu* (*Front* hay *Head*).

Hình 3a mô phỏng một hàng đợi gồm 8 số nguyên lần lượt là 10, 3, 5, 6, -7, 12, 4, 9, trong đó số 10 đứng đầu hàng đợi, số 9 đứng cuối hàng đợi. Khi dùng mảng một chiều, kí hiệu là Q , để lưu các phần tử này của hàng đợi, ta sẽ lưu như ở *Hình 3b*. Số 10 đứng đầu hàng đợi được lưu ở $Q[Front]$ với $Front = 0$,

tiếp theo là số 3 được lưu ở $Q[Front + 1]$, số 5 được lưu ở $Q[Front + 2]$,... và số 9 đứng cuối hàng đợi được lưu ở $Q[Back]$. Trong trường hợp này, $Back = 7$.



Hình 3a. Hàng đợi chứa các phần tử là các số nguyên



Hình 3b. Biểu diễn hàng đợi ở Hình 3a
bởi mảng một chiều



Hình 4a. Hàng đợi thu được từ Hình 3a sau khi thực hiện một thao tác lấy ra



Hình 4b. Biểu diễn hàng đợi ở Hình 4a
bởi mảng một chiều

Nếu tiến hành thực hiện một thao tác lấy ra, số 10 do đang đứng đầu hàng đợi sẽ được đưa ra khỏi hàng đợi, lúc này trong hàng đợi chỉ còn 7 số nguyên xếp hàng theo thứ tự 3, 5, 6, -7, 12, 4, 9 như trong *Hình 4a*. Tương ứng trong mảng một chiều Q (*Hình 4b*), sau khi thực hiện thao tác lấy ra, ta cần tăng giá trị biến *Front* lên 1 đơn vị, để $Q[Front]$ luôn lưu trữ phần tử đứng đầu hàng đợi và lúc này chính là số 3 ở vị trí $Q[1]$.



Em h̄y:

- a) Cho biết những thao tác nào cần được thực hiện để có thể lấy số 6 ra khỏi hàng đợi ở *Hình 4a* và vẽ hàng đợi biểu diễn bởi mảng một chiều tại thời điểm lấy xong số 6. Giá trị biến *Front* thay đổi thế nào so với trạng thái ban đầu ở *Hình 4b*.

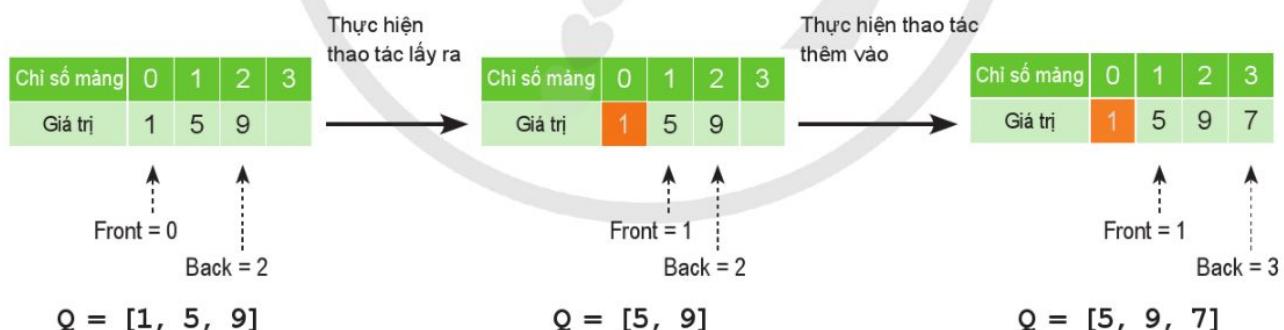
b) Vẽ hàng đợi thu được khi tiếp tục thực hiện một thao tác thêm vào số 8. Giá trị biến *Back* thay đổi thế nào so với *Hình 4b*.

c) Tiếp tục thực hiện các thao tác lấy ra cho đến khi hàng đợi rỗng, và cho biết mối quan hệ giữa giá trị biến *Front* và *Back* khi hàng đợi rỗng.

Trong một hàng đợi, các phần tử phải được lấy ra theo thứ tự từ đầu đến cuối một cách lần lượt (tức là, lấy ra theo thứ tự chúng được thêm vào hàng đợi), không được phép lấy ra phần tử bất kì. Các phần tử trong hàng đợi không truy cập được một cách trực tiếp như ở kiểu dữ liệu mảng (ví dụ: $a[i]$ dùng để truy cập phần tử thứ i của mảng a), mà chỉ được truy cập lần lượt từng phần tử một, theo thứ tự từ đầu đến cuối thông qua thực hiện các thao tác lấy ra.

Khi biểu diễn hàng đợi bởi mảng một chiều, kí hiệu là Q , ta cần hai biến $Front$ và $Back$ để lưu lần lượt chỉ số trong mảng của phần tử đứng đầu hàng đợi và đứng cuối hàng đợi. Trong hàng đợi sẽ chỉ bao gồm các phần tử từ $Q[Front]$, $Q[Front + 1], \dots, Q[Back]$. Để đảm bảo điều này, sau khi thực hiện một thao tác lấy ra, ta cần tăng giá trị biến $Front$ lên 1 đơn vị; còn khi thực hiện một thao tác thêm vào thì trước tiên giá trị biến $Back$ tăng lên 1 đơn vị, sau đó gán $Q[Back]$ bằng phần tử cần thêm vào hàng đợi. Nhược điểm của cách làm này là các phần tử từ $Q[0]$ đến $Q[Front - 1]$ không được dùng để lưu những phần tử mới được thêm vào hàng đợi. Vì vậy có thể xảy ra tình huống mảng không còn đủ vị trí trống thích hợp để lưu các phần tử trong hàng đợi mặc dù số lượng phần tử trong hàng đợi chưa vượt quá kích thước mảng.

Hình 5 là một ví dụ minh họa về điều này. Ban đầu, mảng Q lưu trữ ba phần tử của hàng đợi theo thứ tự từ đầu đến cuối là $Q[0] = 1, Q[1] = 5, Q[2] = 9$, với $Front = 0, Back = 2$. Khi thực hiện một thao tác lấy ra, biến $Front$ được tăng lên 1 đơn vị và có giá trị bằng 1, tương ứng phần tử đứng đầu hàng đợi lúc này sẽ là $Q[1] = 5$ và $Q[0]$ không còn được dùng để lưu trữ phần tử trong hàng đợi. Tiếp theo, một thao tác thêm vào được thực hiện để thêm số 7 vào hàng đợi. Vì vậy, biến $Back$ được tăng lên 1 đơn vị và có giá trị bằng 3, sau đó $Q[Back]$ được gán bằng 7. Dù mảng có thể lưu trữ được tối đa bốn phần tử, nhưng tại bước cuối cùng, ta chỉ dùng các phần tử $Q[1], Q[2], Q[3]$ để lưu hàng đợi. Do vậy, không thể tiếp tục thực hiện một thao tác thêm vào vì mảng không có vị trí $Q[4]$ để lưu và $Q[0]$ dù đang là vị trí còn trống nhưng lại không được dùng để lưu phần tử thuộc hàng đợi.



Hình 5. Một ví dụ minh họa về biểu diễn hàng đợi bởi mảng một chiều

Để khắc phục được nhược điểm này, người ta đề xuất một số phương pháp khác. Tuy nhiên do giới hạn trong bài học sẽ không trình bày.

3) Cài đặt hàng đợi

Nhận thấy rằng, khi các phần tử trong hàng đợi liên tục được cập nhật bởi một số lượng lớn các thao tác thêm vào và lấy ra, ta cần phải có sự trợ giúp của máy tính.

Phần này sẽ tìm hiểu cách cài đặt một số hàm thực hiện các phép toán cơ bản trên hàng đợi. *Bảng 1* là danh sách một số hàm cơ bản thường được cài đặt khi làm việc với hàng đợi.

Bảng 1. Một số hàm cơ bản trên hàng đợi

HÀM	Ý NGHĨA
createQueue ()	Khởi tạo hàng đợi rỗng.
front (Q)	Hàm trả về phần tử đang đứng đầu hàng đợi Q nhưng không lấy nó ra khỏi Q.
enqueue (Q, data)	Thêm phần tử data vào cuối hàng đợi Q.
dequeue (Q)	Lấy ra khỏi hàng đợi Q phần tử đang đứng đầu hàng đợi và trả về phần tử này.
isEmptyQueue (Q)	Hàm trả về giá trị True nếu hàng đợi Q đang rỗng, ngược lại trả về giá trị False.

Để đơn giản, ta cài đặt hàng đợi Q bằng mảng một chiều sử dụng kiểu dữ liệu danh sách (kiểu *list* của Python). Khi đó, đầu *Front* ở phần tử đầu tiên (biến $Front = 0$) và đầu *Back* ở phần tử cuối cùng của danh sách (biến $Back = \text{len}(Q) - 1$). Do đó, ta không cần có biến *Front* và biến *Back* khi cài đặt.

Hàm *createQueue()* trong *Hình 6* trả về một danh sách rỗng để tạo hàng đợi rỗng.

Hàm *front(Q)* trong *Hình 7* thực hiện trả về phần tử đang đứng đầu hàng đợi Q nhưng không lấy nó ra khỏi Q .

Hàm *enqueue(Q, data)* trong *Hình 8* thêm *data* vào hàng đợi Q , nghĩa là thêm *data* vào cuối danh sách.

```
File Edit Format Run Options Window Help
1 def createQueue():
2     return []
```

Hình 6. Hàm khởi tạo hàng đợi rỗng

```
File Edit Format Run Options Window Help
1 def front(Q):
2     return Q[0]
```

Hình 7. Hàm trả về phần tử đầu hàng đợi

```
File Edit Format Run Options Window Help
1 def enqueue(Q, data):
2     Q.append(data)
```

Hình 8. Hàm thêm một phần tử vào hàng đợi

4 ➤ Thực hành

Em hãy:

- Đọc hàm `dequeue(Q)` ở *Hình 9* và cho biết dấu `(?)` cần được thay bằng gì.
- Đọc chương trình ở *Hình 10* và cho biết hàng đợi `queue` sẽ chứa những phần tử nào và kết quả in trên màn hình là gì khi chạy chương trình.
- Bổ sung ba câu lệnh `temp = dequeue(queue)` giống nhau vào cuối đoạn chương trình ở *Hình 10* và cho biết kết quả in trên màn hình là gì khi chạy chương trình.
- Viết hàm `isEmptyQueue(Q)` với tham số truyền vào là hàng đợi `Q`. Hàm trả về giá trị `True` nếu hàng đợi `Q` đang rỗng không chứa phần tử nào, ngược lại hàm trả về giá trị `False`.

Gợi ý: Hàm `len(Q)` trả về số phần tử của hàng đợi `Q`.

- Sửa lại chương trình thu được khi thực hiện xong câu c) như sau: Thay mỗi câu lệnh `temp = dequeue(queue)` thành đoạn chương trình ở *Hình 11*. Cho biết kết quả in trên màn hình là gì khi chạy chương trình.

```
File Edit Format Run Options Window Help
1 def dequeue(Q):
2     return Q.pop(?)
```

Hình 9. Hàm lấy ra một phần tử của hàng đợi

```
File Edit Format Run Options Window Help
1 queue = createQueue()
2 enqueue(queue,2)
3 enqueue(queue,4)
4 enqueue(queue,6)
5 temp = dequeue(queue)
6 print("Phần tử đang đứng đầu hàng đợi = ", front(queue))
```

Hình 10. Đoạn chương trình kiểm thử

```
File Edit Format Run Options Window Help
1 if not isEmptyQueue(queue):
2     temp = dequeue(queue)
3     print("Phần tử", temp, "đã được lấy ra khỏi hàng đợi")
4 else:
5     print("Hàng đợi rỗng, không thực hiện được thao tác dequeue")
```

Hình 11. Kiểm tra hàng đợi rỗng trước khi thực hiện thao tác `dequeue`

Chú ý: Python cung cấp lớp *Queue* trong thư viện có sẵn *queue* để người lập trình có thể sử dụng mà không cần phải cài đặt các thao tác cơ bản trên cấu trúc dữ liệu hàng đợi. Để có thể sử dụng các hàm có sẵn của thư viện này, ta cần bổ sung thêm câu lệnh **from queue import Queue** vào chương trình.



Tại căng tin, các bạn học sinh đang xếp thành hai hàng để chờ đến lượt nhận suất ăn trưa. Để công bằng, bác đầu bếp sẽ phát suất ăn trưa cho lần lượt từng bạn theo quy tắc: bạn đầu hàng của hàng thứ nhất, rồi đến bạn đầu hàng của hàng thứ hai. Thứ tự này lặp đi lặp lại cho đến khi một trong hai hàng không còn học sinh. Khi đó, những bạn ở hàng còn lại sẽ được tiếp tục phát theo thứ tự lần lượt từ đầu đến cuối hàng cho đến khi tất cả các học sinh đều nhận được suất ăn của mình. Em hãy viết hàm *nhan_suat_an(Q1, Q2)* với tham số đầu vào là hai hàng đợi *Q1, Q2*. Hàm thực hiện in ra màn hình tên của các học sinh một cách lần lượt theo đúng thứ tự nhận suất ăn trưa.

Ví dụ: Các bạn Thái, Trà, Mai theo thứ tự đang xếp ở hàng đợi *Q1*. Các bạn Đức, Hùng, Cường, Bình, Phước theo thứ tự đang xếp ở hàng đợi *Q2*. Hàm *nhan_suat_an(Q1, Q2)* sẽ in ra màn hình tên của các học sinh theo thứ tự như sau: Thái, Đức, Trà, Hùng, Mai, Cường, Bình, Phước.



Trong các câu sau đây, những câu nào đúng khi nói về hàng đợi?

- Cơ chế hoạt động của hàng đợi là vào trước ra trước.
- Khác với kiểu dữ liệu mảng, các phần tử trong hàng đợi không được truy cập một cách trực tiếp.
- Khi thêm một phần tử vào hàng đợi, phần tử này sẽ được đứng đầu hàng đợi.
- Có thể lấy một phần tử bất kì ra khỏi hàng đợi.
- Có thể thêm một phần tử mới vào vị trí bất kì trong hàng đợi.

Tóm tắt bài học

- ✓ Hàng đợi thuộc kiểu dữ liệu tuyến tính, hoạt động theo cơ chế vào trước ra trước.
- ✓ Hai thao tác cơ bản trong hàng đợi là thêm vào và lấy ra.
- ✓ Các phần tử trong hàng đợi không được truy cập một cách trực tiếp, mà phải truy cập lần lượt từng phần tử một theo thứ tự từ đầu đến cuối thông qua các thao tác lấy ra.

Bài 2

KIỂU DỮ LIỆU NGĂN XẾP

Học xong bài này, em sẽ:

- Biết được kiểu dữ liệu ngăn xếp là kiểu dữ liệu tuyến tính.
- Mô tả được khái niệm ngăn xếp và cơ chế hoạt động của nó.
- Biểu diễn được ngăn xếp bằng mảng một chiều và viết được chương trình con thực hiện các phép toán cơ bản trên ngăn xếp.
- Nêu được một số ứng dụng của ngăn xếp.



Lan xếp các đĩa CD thành một cọc (*Hình 1*). Mỗi lần lấy đĩa ra khỏi cọc, Lan sẽ lấy lần lượt từng đĩa một từ trên xuống. Mỗi lần bổ sung, Lan cũng lần lượt xếp từng đĩa mới vào cọc.

Em hãy:

- Cho biết với đĩa nằm ở đáy và đĩa nằm ở đỉnh cọc, đĩa nào được thêm vào cọc trước.
- So sánh quy tắc thực hiện thao tác thêm vào và lấy đĩa ra khỏi cọc với thao tác thêm vào và lấy ra phần tử khỏi hàng đợi đã được học ở bài trước.



Hình 1. Ví dụ một cọc đĩa CD



1 ► Một số ví dụ về ngăn xếp và cơ chế hoạt động

Hình 1 mô tả một cọc gồm rất nhiều đĩa CD được xếp theo thứ tự từ dưới lên trên. Đĩa đầu tiên được thêm vào cọc sẽ nằm ở đáy, đĩa được thêm vào sau cùng sẽ nằm ở đỉnh của cọc. Thao tác thêm đĩa vào và lấy đĩa ra khỏi cọc chỉ được phép thực hiện tại vị trí đỉnh (trên cùng) của cọc. Theo quy tắc này, đĩa CD nào được xếp vào cọc sau cùng thì sẽ được lấy ra khỏi cọc trước tiên. Trong tin học, người ta gọi quy tắc này là vào sau ra trước hay còn gọi là LIFO (viết tắt của *Last In First Out* trong tiếng Anh) và xây dựng kiểu dữ liệu ngăn xếp (gọi là *Stack*) để mô phỏng cơ chế hoạt động LIFO.

Hình 2 mô tả một số ví dụ về ngăn xếp diễn ra trong cuộc sống hằng ngày. Trong tin học cũng có nhiều ứng dụng sử dụng ngăn xếp. Ví dụ: Để xây dựng tính năng quay lại trang vừa truy cập ngay trước trong các trình duyệt web, người ta sử dụng một ngăn xếp lưu trữ các đường dẫn liên kết (URL) của các trang web mà người dùng đã và đang truy cập. Mỗi khi người dùng vào một trang web mới, đường dẫn liên kết của trang này sẽ được bổ sung vào ngăn xếp. Khi người dùng bấm nút quay lại trang trước, đường dẫn ở đỉnh ngăn xếp sẽ được loại bỏ khỏi ngăn xếp và lúc này trang web tương ứng với đường dẫn ngay trước đó (đang ở đỉnh ngăn xếp) sẽ được hiển thị lên màn hình người dùng.



Hình 2. Một số ví dụ về ngăn xếp trong thực tế

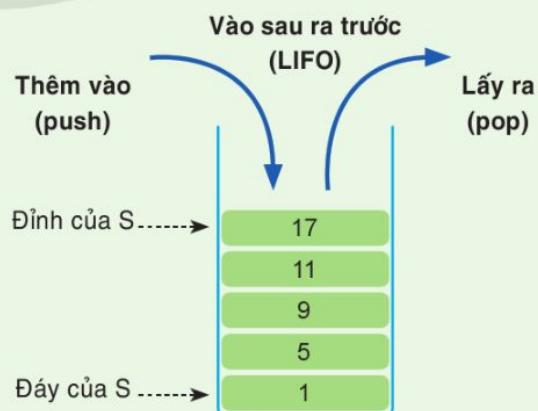
2) Kiểu dữ liệu ngăn xếp và các phép toán cơ bản trên ngăn xếp



1

Cho dãy A gồm 10 số nguyên lẻ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19. Bạn Thái sẽ thực hiện một cách tùy ý các thao tác thêm vào và lấy ra trên ngăn xếp S ban đầu đang không có phần tử nào. Các thao tác thêm vào sẽ lấy ra lần lượt từng số trong dãy A để bổ sung vào ngăn xếp. Em hãy:

- Vẽ ngăn xếp S thu được sau khi Thái thực hiện hai thao tác thêm vào liên tiếp và một thao tác lấy ra.
- Cho biết Thái cần thực hiện những thao tác thêm vào và lấy ra theo thứ tự như thế nào để có thể thu được ngăn xếp S như ở Hình 3.



Hình 3. Ngăn xếp S

Ngăn xếp thuộc kiểu dữ liệu tuyến tính, các phần tử được sắp xếp một cách lần lượt từ dưới đáy lên trên đỉnh, hết phần tử này đến phần tử khác, trong đó phép toán thêm vào (*push*) và lấy ra (*pop*) chỉ được thực hiện ở đỉnh (*Top*) của ngăn xếp.

Tại bất kì thời điểm nào, ta cũng chỉ truy cập được vào một phần tử trên cùng (ở đỉnh) của ngăn xếp. Do đó, trong một ngăn xếp, các phần tử phải được lấy ra theo thứ tự từ đỉnh xuống đáy một cách lần lượt (tức là, phần tử nào vào ngăn xếp sau sẽ được lấy ra khỏi ngăn xếp trước), không được phép lấy ra khỏi ngăn xếp phần tử bất kì. Tương tự, mỗi thao tác thêm vào sẽ chỉ thêm được một phần tử vào vị trí ngay trên đỉnh của ngăn xếp. Các phần tử trong ngăn xếp không được truy cập một cách trực tiếp như ở kiểu dữ liệu mảng. Muốn lấy ra phần tử thứ i tính từ đỉnh ngăn xếp xuống, ta sẽ phải thực hiện liên tiếp ($i - 1$) thao tác lấy ra để phần tử thứ i trở thành phần tử ở đỉnh ngăn xếp.

3 Cài đặt ngăn xếp

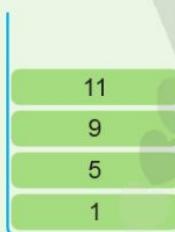


2

Để cài đặt ngăn xếp với hai thao tác thêm vào (*push*) và lấy ra (*pop*), ta có thể dùng mảng một chiều. Khi đó, các phần tử trong mảng sẽ là các phần tử đang có trong ngăn xếp.

Đỉnh của S ----->
(Top)

Đáy của S ----->



Hình 4a. Ngăn xếp S

Chỉ số mảng	0	1	2	3	4	5	6
Giá trị	1	5	9	11			

↑
Top = 3

Hình 4b. Biểu diễn ngăn xếp S ở Hình 4a
bởi mảng một chiều

Ví dụ: Ngăn xếp S ở *Hình 4a* có thể được biểu diễn bởi một mảng một chiều mô phỏng như ở *Hình 4b*. Các phần tử trong mảng theo thứ tự từ đầu đến cuối sẽ tương ứng với các phần tử trong ngăn xếp theo thứ tự từ đáy lên đỉnh. Em hãy vẽ ngăn xếp S và xác định các giá trị tương ứng trong mảng mỗi khi thực hiện xong một thao tác của dãy lần lượt ba thao tác sau:

- Thêm vào ngăn xếp S một phần tử có giá trị 13.
- Thêm vào ngăn xếp S một phần tử có giá trị 15.
- Lấy ra một phần tử khỏi ngăn xếp S.

Khi biểu diễn ngăn xếp bởi mảng một chiều, kí hiệu là S , ta cần một biến Top để lưu chỉ số trong mảng của phần tử đứng ở đỉnh của ngăn xếp. Các phần tử trong ngăn xếp theo thứ tự từ đáy lên đỉnh sẽ được lưu tương ứng trong mảng một chiều gồm các phần tử $S[0], S[1], \dots, S[Top]$. Để đảm bảo tính chất vào sau ra trước của ngăn xếp, sau khi thực hiện một thao tác lấy ra, giá trị biến Top sẽ giảm đi 1 đơn vị. Sau khi thực hiện một thao tác thêm vào, ta tăng biến Top lên 1 đơn vị, sau đó gán $S[Top]$ bằng phần tử cần thêm vào.

Để đơn giản, ta cài đặt ngăn xếp S bằng mảng một chiều sử dụng kiểu dữ liệu danh sách (kiểu *list* của Python). Khi đó, đáy của ngăn xếp là phần tử có chỉ số 0 và đỉnh ngăn xếp ở cuối danh sách (biến $Top = len(S) - 1$). Do đó, ta không cần có biến Top .

Bảng 1 là danh sách một số hàm cơ bản thường được cài đặt khi làm việc với ngăn xếp.

Bảng 1. Một số hàm cơ bản trên ngăn xếp

HÀM	Ý NGHĨA
<code>createStack()</code>	Khởi tạo ngăn xếp rỗng.
<code>top(S)</code>	Hàm trả về phần tử đang đứng ở đỉnh ngăn xếp S nhưng không lấy nó ra khỏi S .
<code>push(S, data)</code>	Thêm phần tử <code>data</code> vào đỉnh ngăn xếp S .
<code>pop(S)</code>	Lấy ra khỏi ngăn xếp S phần tử đang đứng ở đỉnh ngăn xếp và trả về phần tử này cho hàm.
<code>isEmptyStack(S)</code>	Hàm trả về giá trị <code>True</code> nếu ngăn xếp S đang rỗng, ngược lại trả về giá trị <code>False</code> .

Hàm `createStack()` trong *Hình 5* trả về một danh sách rỗng để tạo ngăn xếp rỗng.

```

File Edit Format Run Options Window Help
1 def createStack():
2     return []

```

Hình 5. Hàm khởi tạo ngăn xếp rỗng

```

File Edit Format Run Options Window Help
1 def top(s):
2     return s[len(s)-1]

```

Hình 6. Hàm trả về phần tử ở đỉnh ngăn xếp

Hàm `top(S)` trong *Hình 6* thực hiện trả về phần tử đang đứng ở đỉnh ngăn xếp S nhưng không lấy nó ra khỏi S .

Hàm $push(S, data)$ trong *Hình 7* thực hiện thêm $data$ vào ngăn xếp S , nghĩa là thêm $data$ vào cuối danh sách.

```
File Edit Format Run Options Window Help
1 def push(S,data):
2     S.append(data)
```

Hình 7. Hàm thêm một phần tử vào ngăn xếp

4) Thực hành

Em hãy:

- Đọc hàm $pop(S)$ ở *Hình 8* và cho biết dấu $\textcolor{red}{?}$ cần được thay bằng gì.
- Đọc chương trình ở *Hình 9* và cho biết kết quả thu được khi chạy chương trình.
- Bổ sung hai câu lệnh $temp = pop(S)$ giống nhau vào cuối đoạn chương trình ở *Hình 9* và cho biết kết quả thu được khi chạy chương trình.
- Viết hàm $isEmptyStack(S)$ với tham số truyền vào là ngăn xếp S . Hàm trả về giá trị *True* nếu ngăn xếp S đang rỗng không chứa phần tử nào, ngược lại hàm trả về giá trị *False*.
- Sửa lại chương trình thu được khi thực hiện xong câu c) như sau: Thay mỗi câu lệnh $temp = pop(S)$ thành đoạn chương trình ở *Hình 10*. Cho biết kết quả thu được khi chạy chương trình.

```
File Edit Format Run Options Window Help
1 def pop(S):
2     return ?
```

Hình 8. Hàm lấy ra một phần tử của ngăn xếp

```
File Edit Format Run Options Window Help
1 stack = createStack()
2 push(stack,2)
3 push(stack,4)
4 temp = pop(stack)
5 print("Phần tử đứng đầu stack
6 là",top(stack))
```

Hình 9. Đoạn chương trình kiểm thử

```
File Edit Format Run Options Window Help
1 if not isEmptyStack(stack):
2     temp = pop(stack)
3     print("Phần tử",temp,"đã được lấy ra khỏi ngăn xếp")
4 else:
5     print("Ngăn xếp rỗng, không thực hiện được thao tác pop")
```

*Hình 10. Kiểm tra ngăn xếp rỗng trước khi thực hiện thao tác *pop**

Chú ý: Python cung cấp lớp `LifoQueue` trong thư viện có sẵn `queue` để người lập trình có thể sử dụng mà không cần phải cài đặt các thao tác cơ bản trên cấu trúc dữ liệu ngăn xếp. Để có thể sử dụng các hàm có sẵn của thư viện này, ta cần bổ sung thêm câu lệnh `from queue import LifoQueue` vào chương trình.



Em hãy viết chương trình yêu cầu người sử dụng nhập năm số nguyên dương bất kì từ bàn phím, sau đó in ra màn hình năm số này theo thứ tự đảo ngược của thứ tự nhập vào. Trong chương trình có sử dụng kiểu dữ liệu ngăn xếp và các thao tác đã học trên ngăn xếp.

Ví dụ: Nhập vào năm số: 3, 1, 9, 17, 2. Kết quả in ra: 2, 17, 9, 1, 3.



Trong các câu sau đây, những câu nào đúng khi nói về ngăn xếp?

- a) Cơ chế hoạt động của ngăn xếp là vào trước ra trước.
- b) Khác với kiểu dữ liệu hàng đợi, các phần tử trong ngăn xếp được truy cập một cách trực tiếp.
- c) Khi thêm một phần tử vào ngăn xếp, phần tử này sẽ được đặt ở đáy của ngăn xếp.
- d) Có thể thêm một phần tử mới vào vị trí bất kỳ trong ngăn xếp.
- e) Cơ chế hoạt động của ngăn xếp là vào sau ra trước.
- g) Có thể xoá một phần tử bất kỳ khỏi ngăn xếp.

Tóm tắt bài học

- ✓ Ngăn xếp thuộc kiểu dữ liệu tuyến tính, hoạt động theo cơ chế vào sau ra trước.
- ✓ Hai thao tác cơ bản trong ngăn xếp là thêm vào và lấy ra.
- ✓ Các phần tử trong ngăn xếp không được truy cập một cách trực tiếp, mà phải truy cập lần lượt từng phần tử một theo thứ tự từ đỉnh ngăn xếp đến đáy ngăn xếp thông qua các thao tác lấy ra.
- ✓ Tại bất kỳ thời điểm nào, ta cũng chỉ truy cập được vào phần tử ở đỉnh của ngăn xếp.

Bài 3

THỰC HÀNH KIỂU DỮ LIỆU HÀNG ĐỢI VÀ NGĂN XẾP

Học xong bài này, em sẽ:

- Viết và thực hiện được một vài chương trình sử dụng kiểu dữ liệu hàng đợi và ngăn xếp.

Thực hành 1. Chương trình ở *Hình 1* mô phỏng một ứng dụng của kiểu dữ liệu ngăn xếp trong việc xây dựng tính năng quay lại trang vừa truy cập trên các trình duyệt web. Tìm hiểu chương trình ở *Hình 1* và cho biết kết quả in ra màn hình khi người sử dụng nhập các giá trị số nguyên hoặc xâu kí tự từ bàn phím theo kịch bản lần lượt như sau:

- Ấn phím 2 và phím Enter.
- Ấn phím 1 và phím Enter, sau đó nhập xâu *moet.gov.vn* và ấn phím Enter.
- Ấn phím 1 và phím Enter, sau đó nhập xâu *sachcanhdieu.com* và ấn phím Enter.
- Ấn phím 2 và phím Enter.
- Ấn phím 1 và phím Enter, sau đó nhập xâu *chinhphu.vn* và ấn phím Enter.
- Ấn phím 2 và ấn phím Enter.
- Ấn phím 2 và ấn phím Enter.
- Ấn phím 3 và ấn phím Enter.

Em hãy soạn thảo và chạy chương trình để kiểm tra dự đoán của em.

```
File Edit Format Run Options Window Help
1 def createStack():
2     return []
3 def push(S,data):
4     S.append(data)
5 def pop(S):
6     return S.pop()
7 def top(S):
8     return S[len(S)-1]
9 def isEmptyStack(S):
10    return len(S) == 0
```

```

14 def print_menu():
15     print("----- Mô phỏng nút back trên trình duyệt web -----")
16     print("1 -- Nhập URL để vào trang mới")
17     print("2 -- Bấm nút back để quay lại trang cũ")
18     print("3 -- Kết thúc chương trình")
19     print("-----")
20 def option1(S):
21     URL = input("Nhập URL: ")
22     push(S, URL)
23     print("Trình duyệt đang hiển thị trang web: ", top(S))
24 def option2(S):
25     if isEmptyStack(S):
26         print("Chưa nhập URL nào, chức năng back không thực hiện")
27     elif (len(S) == 1):
28         print(top(S), "là URL đầu tiên được truy cập khi mở trình
29 duyệt, chức năng back không thực hiện")
30     else:
31         pop(S)
32         print("Trình duyệt đang hiển thị trang web: ", top(S))
33 stack = createStack()
34 option = 1
35 while (option < 3):
36     print_menu()
37     option = int(input("Lựa chọn của bạn (ấn số từ 1 đến 3): "))
38     if option == 1:
39         option1(stack)
40     elif option == 2:
41         option2(stack)
42     elif option == 3:
43         print("Cảm ơn đã sử dụng chương trình. Hẹn gặp lại.")
44     else:
45         print("Bạn đã nhập sai. Chỉ nhập số có giá trị từ 1 đến 3.")

```

Hình 1. Chương trình sử dụng kiểu dữ liệu ngăn xếp xây dựng tính năng
quay lại trang vừa truy cập trên các trình duyệt web

Thực hành 2: Tìm hiểu chương trình ở *Hình 2* và thực hiện các yêu cầu sau:

- Cho biết trong hàng đợi queue có những số nào khi người sử dụng nhập $N = 4$, dãy bốn số 1, 3, 2, 4 và $M = 2$.
- Soạn thảo chương trình ở *Hình 2* và bổ sung thêm đoạn chương trình in ra các số đang có trong hàng đợi queue. Chạy chương trình để kiểm tra dự đoán của em.
- Cho biết chương trình thực hiện công việc gì.

```

1 def createQueue():
2     return []
3 def enqueue(Q, data):
4     Q.append(data)
5 def dequeue(Q):
6     return Q.pop(0)
7 queue = createQueue()
8 N = int(input("Nhập số nguyên dương N ="))
9 for i in range(1, N + 1):
10    temp = int(input("Nhập số thứ " + str(i) + ": "))
11    enqueue(queue, temp)
12 M = int(input("Nhập số nguyên dương M ="))
13 for i in range(1, M + 1):
14    temp = dequeue(queue)
15    enqueue(queue, temp)

```

Hình 2. Chương trình sử dụng kiểu dữ liệu hàng đợi



Một xâu kí tự được gọi là có tính chất đối xứng nếu viết từ trái sang phải cũng giống như viết từ phải sang trái (không phân biệt chữ viết hoa và chữ viết thường). *Ví dụ:* “level”, “madam”, “mom”, “civic”, “Able was I ere I saw Elba” là những xâu có tính chất đối xứng. Em hãy thực hiện các yêu cầu sau:

- Dựa trên cơ chế hoạt động vào trước ra trước của hàng đợi và vào sau ra trước của ngăn xếp, em hãy thiết kế thuật toán sử dụng một ngăn xếp và một hàng đợi để kiểm tra xem một xâu kí tự bất kì có tính chất đối xứng hay không.
- Viết hàm *doixung(s)* thực hiện thuật toán xây dựng được ở câu a) để kiểm tra xem xâu *s* có tính chất đối xứng hay không.
- Viết chương trình hoàn thiện, yêu cầu người sử dụng nhập vào một xâu kí tự, rồi gọi hàm *doixung(s)* đã viết được ở câu b) để kiểm tra xâu nhập vào có tính chất đối xứng hay không. Chạy chương trình với các xâu kí tự sau: “123454321”, “112”, “racecar” và cho biết kết quả thu được.

Bài 4

DỰ ÁN HỌC TẬP: XÂY DỰNG CHƯƠNG TRÌNH SỬ DỤNG KIỂU DỮ LIỆU HÀNG ĐỢI VÀ NGĂN XẾP

Trong Tin học, kiểu dữ liệu ngăn xếp và hàng đợi được ứng dụng rất nhiều khi giải quyết các bài toán trong thực tế. Dưới đây giới thiệu thêm một số ứng dụng điển hình của kiểu dữ liệu ngăn xếp và hàng đợi.

1 Mục đích của dự án học tập

Dự án nhằm mục đích giúp học sinh khám phá, tìm hiểu xây dựng chương trình giải quyết một số bài toán ứng dụng sử dụng hàng đợi và ngăn xếp. Sau khi hoàn thành xong dự án, học sinh có khả năng:

- Vận dụng kiến thức đã học để giải quyết vấn đề cụ thể.
- Tìm kiếm, khai thác, học hỏi mở rộng kiến thức từ các nguồn tài nguyên về lập trình trên Internet.
- Làm việc nhóm.

2 Yêu cầu chung

- Lớp chia thành nhiều nhóm, các nhóm độc lập, mỗi nhóm lựa chọn một dự án để thực hiện. Mỗi học sinh tham gia một nhóm, việc tham gia nhóm dự án nào là do học sinh lựa chọn. Sau đó thầy, cô giáo điều chỉnh để đảm bảo cân đối số học sinh tham gia trong mỗi dự án.
- Mỗi nhóm bầu trưởng nhóm và thảo luận chọn bài toán để xây dựng chương trình.
- Các nhóm sẽ thực hiện tạo sản phẩm ngoài giờ lên lớp. Trên lớp, 1 tiết giáo viên hướng dẫn, 2 tiết các nhóm báo cáo kết quả.

3 Một số hướng dẫn và gợi ý thực hiện dự án

a) Lập kế hoạch

Bước 1. Thảo luận, thống nhất chọn một trong ba dự án.

Bước 2. Lập danh sách các đầu công việc cần thực hiện và thời gian hoàn thành.

b) Thực hiện dự án

Bước 1. Thiết kế thuật toán.

Bước 2. Sử dụng các lệnh trong Python để viết chương trình.

Bước 3. Thủ nghiệm, kiểm thử để sửa lỗi, hiệu chỉnh chương trình.

c) Báo cáo kết quả

Trong thời gian 2 tiết, tiến trình thực hiện như trong *Bảng 1*.

Bảng 1. Tiến trình báo cáo kết quả dự án

SỐ THỨ TỰ	NỘI DUNG, NHIỆM VỤ	NGƯỜI THỰC HIỆN	THỜI GIAN VÀ ĐỊA ĐIỂM	CẦN CHUẨN BỊ
1	Các nhóm báo cáo kết quả và tự đánh giá	Tất cả các nhóm	Phòng học bộ môn	Máy tính và máy chiếu
2	Tranh biện giữa các nhóm, đánh giá chéo	Tất cả các nhóm	Phòng học bộ môn	Máy tính và máy chiếu
3	Giáo viên tổng kết, đánh giá, kết luận	Giáo viên và tất cả các nhóm	Phòng học bộ môn	Máy tính và máy chiếu

Khi báo cáo, các nhóm cần trình bày chi tiết các nội dung sau:

- Giới thiệu, demo sản phẩm.
- Trình bày kết quả thực hiện dự án, các chức năng và cách thức triển khai.
- Nhận xét, tự đánh giá về sự đóng góp của từng thành viên trong việc thực hiện dự án.

4) Tiêu chí đánh giá

Việc đánh giá sản phẩm và báo cáo dự án như *Bảng 2*. Điểm của mỗi thành viên dựa trên sự đóng góp trong dự án.

Bảng 2. Tiêu chí đánh giá

TIÊU CHÍ	MÔ TẢ	ĐIỂM
Chương trình	Chạy đúng, ổn định Hiệu quả Tuỳ biến, mở rộng	4
Giao diện	Đẹp, hấp dẫn	2
Báo cáo	Trình bày rõ ràng Trả lời câu hỏi tốt	4

Dự án 1. Xây dựng ứng dụng kiểm tra lỗi cú pháp

Cho một xâu kí tự chỉ gồm các loại dấu mở ngoặc và đóng ngoặc sau: [,] , { , } , (,). Cần kiểm tra xem các cặp ngoặc và thứ tự của chúng trong xâu kí tự có thoả mãn hai điều kiện sau hay không:

- Có dấu mở ngoặc thì phải có dấu đóng ngoặc cùng loại ở phía sau và đồng thời nếu có dấu đóng ngoặc thì phía trước phải có dấu mở ngoặc cùng loại. Ví dụ: Nếu phía trước xâu có dấu mở ngoặc nhọn “{” thì phía sau xâu phải có dấu đóng ngoặc nhọn “}”.
- Nếu dấu mở ngoặc loại này (tạm gọi là loại *a*) đứng trước dấu mở ngoặc của loại khác (tạm gọi là loại *b*), thì dấu đóng ngoặc tương ứng của loại *a* phải đứng sau dấu đóng ngoặc tương ứng của loại *b*.

Xâu kí tự thoả mãn hai điều kiện trên được gọi là xâu cân bằng.

Một số ví dụ về xâu cân bằng: {}, [{}], [](){}, {{}}{}, {[{}]}[()]. *Bảng 3* đưa ra một số ví dụ về xâu không cân bằng.

Bảng 3. Một số ví dụ về xâu không cân bằng

XÂU	LÝ DO KHÔNG CÂN BẰNG
{	Thiếu mất dấu đóng ngoặc nhọn “}”. Cách sửa lại cho đúng { }
{)	Có mở ngoặc nhọn “{”, nhưng sau đó lại không có đóng ngoặc nhọn “}”. Có dấu đóng ngoặc tròn “)”, nhưng trước đó lại không có dấu mở ngoặc tròn “(”. Cách sửa lại cho đúng () hoặc { }
{()}	Mặc dù có đủ mở ngoặc nhọn và đóng ngoặc nhọn { }, mở ngoặc tròn và đóng ngoặc tròn (), nhưng thứ tự xuất hiện lại không đúng: mở ngoặc nhọn xuất hiện trước mở ngoặc tròn, nhưng đóng ngoặc nhọn lại đứng trước đóng ngoặc tròn. Một cách sửa lại cho đúng: {())}
{()}{)}	Có mở ngoặc vuông “[” nhưng lại không có đóng ngoặc vuông “]”. Dấu đóng ngoặc tròn “)” cuối cùng không đi cặp với dấu mở ngoặc tròn “(” nào trước nó

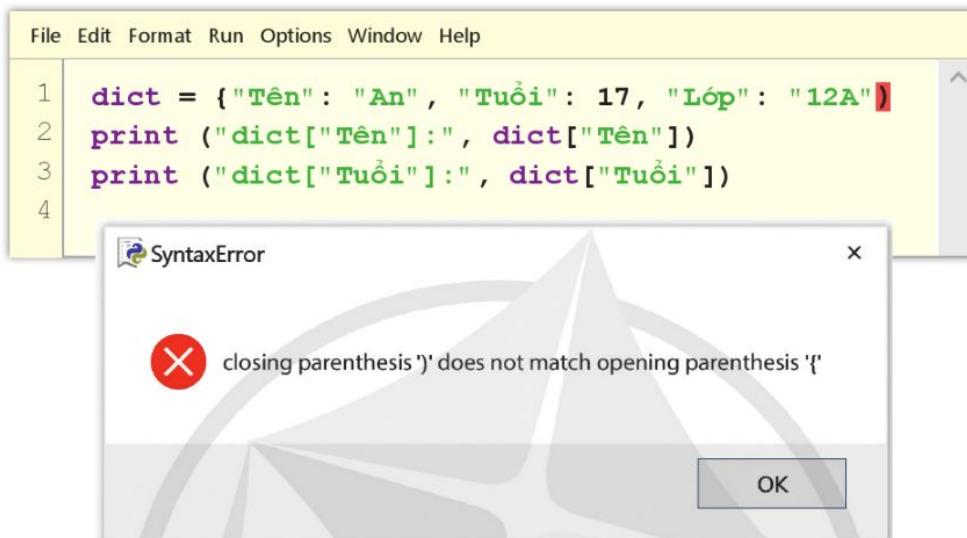
Bài tập trên có thể ứng dụng để kiểm tra lỗi cú pháp về thứ tự xuất hiện của các cặp ngoặc trong một ngôn ngữ lập trình. Ví dụ khi chạy chương trình viết bằng ngôn ngữ Python ở *Hình 1*, ta sẽ nhận được thông báo về lỗi cú pháp ở *Hình 2* với nội dung: dấu đóng ngoặc “)” không cùng một cặp với dấu mở ngoặc “{”; và dấu đóng ngoặc tròn “)” cuối cùng ở dòng 1 của chương trình bị bôi đỏ. Chương trình sẽ không còn lỗi cú pháp nếu dấu đóng ngoặc tròn “)” ở dòng 1 được thay bằng dấu đóng ngoặc nhọn “}”.

```

File Edit Format Run Options Window Help
1 dict = {"Tên": "An", "Tuổi": 17, "Lớp": "12A"}
2 print ("dict["Tên"]:", dict["Tên"])
3 print ("dict["Tuổi"]:", dict["Tuổi"])

```

Hình 1. Chương trình trên Python sai lỗi cú pháp



Hình 2. Báo lỗi khi chạy chương trình ở Hình 1

Yêu cầu:

a) Em hãy viết chương trình nhập vào một xâu kí tự chỉ gồm các dấu mở ngoặc và đóng ngoặc sau: [,] , { , } , (,) . Sau đó, kiểm tra xem xâu kí tự đó có cân bằng hay không. In ra màn hình thông báo “Cân bằng” nếu xâu đó là cân bằng, ngược lại in ra “Không cân bằng”.

b) Mở rộng chương trình ở câu a): đọc tệp có tên *test.py* chứa chương trình viết bằng ngôn ngữ Python, và kiểm tra xem chương trình có lỗi cú pháp về thứ tự xuất hiện của các cặp ngoặc [,] , { , } , (,) hay không. In ra màn hình thông báo “Không có lỗi” nếu chương trình không có lỗi về các cặp ngoặc này, ngược lại in ra “Có lỗi cú pháp”.

Gợi ý: Dùng một ngăn xếp *S* chỉ chứa các dấu mở ngoặc, khởi tạo ban đầu bằng rỗng. Duyệt lần lượt từng kí tự trong xâu từ trái sang phải:

- Nếu kí tự hiện tại trong xâu là dấu mở ngoặc: “(” hoặc “{” hoặc “[”, thì thực hiện thao tác *push* đẩy kí tự đó vào ngăn xếp.

- Nếu kí tự hiện tại trong xâu là dấu đóng ngoặc: “)” hoặc “}” hoặc “]”, thì thực hiện thao tác *pop* để lấy ra một dấu mở ngoặc ở ngăn xếp. So sánh dấu đóng ngoặc và dấu mở ngoặc với nhau, nếu chúng không phải là một cặp thì thuật toán kết thúc, in ra thông báo “Không cân bằng”, ngược lại nếu chúng

cùng một cặp thì tiếp tục duyệt kí tự tiếp theo trong xâu. Trong tình huống ngăn xếp đã rỗng, không thực hiện được thao tác *pop*, có nghĩa là số lượng dấu mở ngoặc đang ít hơn số lượng dấu đóng ngoặc, thì thuật toán kết thúc, in ra thông báo “Không cân bằng”.

Khi tất cả các kí tự trong xâu đã được duyệt xong mà trong ngăn xếp *S* vẫn chưa rỗng, có nghĩa là số lượng dấu mở ngoặc nhiều hơn số lượng dấu đóng ngoặc, ta cần in ra thông báo “Không cân bằng”. Ngược lại, in ra thông báo “Cân bằng”.

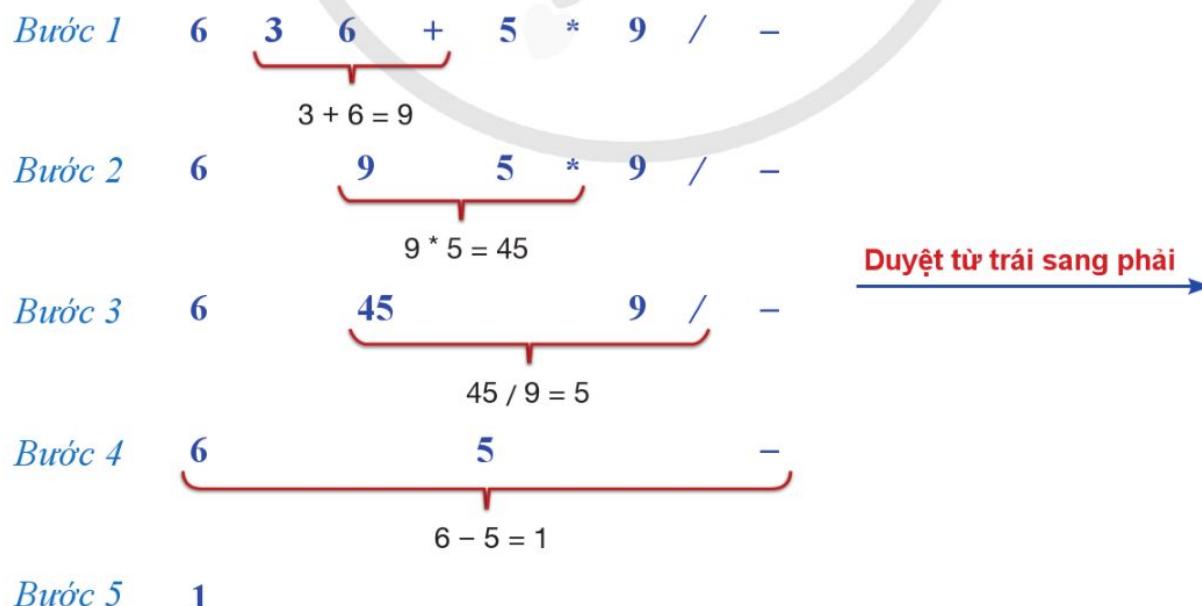
Dự án 2. Kí pháp nghịch đảo Ba Lan và phương pháp tính giá trị của biểu thức toán học

Kí pháp nghịch đảo Ba Lan (tiếng Anh là Reverse Polish Notation hay Postfix Notation) là cách viết một biểu thức toán học mà không cần phải dùng các dấu ngoặc để biết thứ tự ưu tiên thực hiện của các phép toán có trong biểu thức đó.

Ví dụ: Biểu thức toán học “ $A - B / (C + D)$ ” được biểu diễn dưới dạng kí pháp nghịch đảo Ba Lan sẽ là “ $A B C D + / -$ ”.

Ngoài ưu điểm làm cho biểu thức toán học ngắn gọn hơn, cách viết này còn giúp máy tính có thể tính được giá trị của biểu thức một cách dễ dàng hơn, không cần phải tách chuỗi các dấu mở đóng ngoặc trong quá trình tính toán. Việc tính giá trị biểu thức lúc này sẽ dựa trên thứ tự các kí tự từ trái sang phải có trong biểu thức, kết hợp với thứ tự ưu tiên của các phép toán.

Ví dụ: Biểu thức toán học “ $6 - [(3 + 6) * 5] / 9$ ” được biểu diễn dưới dạng kí pháp nghịch đảo Ba Lan sẽ là “ $6 3 6 + 5 * 9 / -$ ”. Quá trình tính toán giá trị biểu thức này được biểu diễn ở *Hình 3*, bao gồm các bước sau:



Hình 3. Cách tính giá trị biểu thức biểu diễn bởi kí pháp nghịch đảo Ba Lan

Bước 1. Duyệt kí pháp từ trái sang phải, gấp phép toán đầu tiên là phép cộng, lấy hai số hạng trước nó là 3 và 6 rồi thực hiện phép cộng trên hai số hạng này: $3 + 6 = 9$, kết quả 9 được đẩy lại vào kí pháp, ta thu được kí pháp mới là: $6 \ 9 \ 5 * 9 / -$.

Bước 2. Duyệt kí pháp mới từ trái sang phải, gấp phép toán đầu tiên là phép nhân, lấy hai số hạng trước nó là 9 và 5 rồi thực hiện phép nhân trên hai số hạng này: $9 * 5 = 45$, kết quả 45 được đẩy lại vào kí pháp, ta thu được kí pháp mới là: $6 \ 45 \ 9 / -$.

Bước 3. Duyệt kí pháp mới từ trái sang phải, gấp phép toán đầu tiên là phép chia, lấy hai số hạng trước nó là 45 và 9 rồi thực hiện phép chia trên hai số hạng này: $45 / 9 = 5$, kết quả 5 được đẩy lại vào kí pháp, ta thu được kí pháp mới là: $6 \ 5 -$.

Bước 4. Duyệt kí pháp mới từ trái sang phải, gấp phép toán đầu tiên là phép trừ, lấy hai số hạng trước nó là 6 và 5 rồi thực hiện phép trừ trên hai số hạng này: $6 - 5 = 1$, kết quả 1 được đẩy lại vào kí pháp, ta thu được kí pháp mới là: 1.

Bước 5. Chỉ còn 1 là giá trị duy nhất trong kí pháp mới, do đó 1 cũng chính là giá trị của biểu thức biểu diễn bởi kí pháp: $6 \ 3 \ 6 + 5 * 9 / -$.

Yêu cầu: Hãy viết chương trình yêu cầu nhập vào một kí pháp nghịch đảo Ba Lan dưới dạng một xâu kí tự chỉ bao gồm: năm toán tử cộng, trừ, nhân, chia, luỹ thừa ($+, -, *, /, ^$), các toán hạng đều chỉ là các số nguyên có một chữ số từ 0 đến 9, không có bất kì khoảng trắng nào giữa các kí tự. Sau đó, tính giá trị của kí pháp này và in kết quả ra màn hình.

Bảng 4 là một số bộ dữ liệu thử nghiệm cho bài toán này.

Bảng 4. Một số bộ dữ liệu thử nghiệm cho bài toán tính giá trị kí pháp nghịch đảo Ba Lan

SỐ THỨ TỰ	DỮ LIỆU VÀO	KẾT QUẢ RA
1	$3 \ 2 * 5 / 1 -$	0.2
2	$6 \ 3 \ 6 + 5 * 9 / -$	1
3	$5 \ 1 \ 2 + 4 * 3 + +$	20
4	$7 \ 2 + 9 / 2 * 3 + 7 +$	12
5	$2 \ 7 + 3 / 2 \ 3 + * 5 +$	20

Gợi ý: Dùng một ngăn xếp S chỉ chứa các toán hạng của kí pháp trong quá trình thực hiện thuật toán, khởi tạo ban đầu bằng rỗng. Duyệt lần lượt từng kí tự trong kí pháp từ trái sang phải:

- Nếu kí tự hiện tại trong kí pháp là toán hạng (con số), thì thực hiện thao tác *push* đẩy kí tự đó vào ngăn xếp.

- Nếu kí tự hiện tại trong kí pháp là toán tử ($+$, $-$, $*$, $/$, $^$), tạm kí hiệu là op : thực hiện hai lần thao tác pop để lấy ra hai toán hạng ở ngăn xếp, kí hiệu lần lượt là A và B tương ứng với lần thực hiện thao tác pop thứ nhất và thứ hai. Thực hiện phép tính: $B \ op \ A$, sau đó thực hiện thao tác $push$ để đẩy kết quả thu được vào ngăn xếp. Ví dụ: Kí tự hiện tại là toán tử lũy thừa ($op = ^$), B là 3, A là 2; khi đó ta thực hiện $B \ op \ A = 3^2 = 9$.

Khi tất cả các kí tự trong kí pháp đã được duyệt qua, trong ngăn xếp S chỉ còn duy nhất một giá trị. Giá trị đó chính là kết quả của biểu thức biểu diễn bởi kí pháp nghịch đảo Ba Lan đã cho.

Dự án 3. Đổi màu một vùng bức tranh



Hình 4a. Bức tranh gốc



Hình 4b. Bức tranh đổi màu đầu con gà từ xanh lá cây sang xanh da trời

Bé An đang tập vẽ tranh. Khi xem lại các bức tranh của mình, có một số bức tranh An muốn đổi màu một số vùng trên bức tranh (xem ví dụ *Hình 4a* và *Hình 4b*). Em hãy giúp An làm được điều này. Biết rằng, mỗi bức tranh của An được biểu diễn dưới dạng một lưới ô vuông kích thước $M \times N$, mỗi ô vuông tương ứng với một điểm ảnh (pixel) của bức tranh. Mỗi ô vuông sẽ chứa một số nguyên dương biểu diễn màu của điểm ảnh tương ứng. Hai điểm ảnh được gọi là *liền kề* nếu chúng có chung một cạnh. Tức là, ô ở tọa độ dòng i , cột j (kí hiệu (i, j)) sẽ liền kề với 4 ô khác ở các tọa độ là: $(i + 1, j)$; $(i - 1, j)$; $(i, j - 1)$; $(i, j + 1)$.

Hai điểm ảnh (i, j) và (k, h) được gọi là thuộc cùng dãy liền kề:

- Nếu hai điểm ảnh (i, j) và (k, h) liền kề và có cùng màu.
- Nếu tồn tại điểm ảnh (x, y) liền kề với cả hai điểm ảnh (i, j) ; (k, h) và cả ba điểm ảnh này cùng màu.
- Nếu tồn tại điểm ảnh (x, y) thuộc cùng dãy liền kề với cả hai điểm ảnh (i, j) và (k, h) .

An cho em thông tin về tọa độ dòng và cột của một điểm ảnh trong lưới ô vuông và giá trị màu mới, nhiệm vụ của em là thay thế màu của điểm ảnh đã cho và tất cả các điểm ảnh thuộc cùng dãy liền kề với nó thành màu mới.

Ví dụ: Hình 5 thể hiện bức tranh của An được biểu diễn bởi lưới ô vuông kích thước 4×4 , An muốn đổi màu của điểm ảnh ở ô có tọa độ $(2, 3)$ đang có màu 3 thành màu 8 và đồng thời tất cả các điểm ảnh thuộc cùng dãy liền kề với điểm ảnh ở ô đó cũng được đổi sang màu 8.

1	1	7	5
1	3	3	3
3	5	5	3
2	2	3	3

1	1	7	5
1	8	8	8
3	5	5	8
2	2	8	8

Hình 5. Ví dụ minh họa lưới ô vuông trước và sau khi đổi màu

Yêu cầu: Em hãy viết chương trình yêu cầu nhập vào lần lượt các thông tin sau:

- Hai giá trị M và N tương ứng với số dòng và số cột của lưới ô vuông.
- Các giá trị màu ở mỗi ô trong lưới ô vuông kích thước $M \times N$ theo thứ tự hết dòng này đến dòng khác.
- Ba giá trị r, c, m lần lượt là tọa độ dòng, tọa độ cột và màu mới của ô muốn đổi màu.

In ra màn hình lưới ô vuông sau khi đã đổi màu các ô theo đúng yêu cầu của An.

Gợi ý: Để duyệt qua được hết tất cả các ô cần đổi màu xuất phát từ ô (i, j) cho trước, ta lần lượt xét các ô liền kề với nó và đánh dấu nếu ô đó cùng màu, tiếp theo với những ô vừa đánh dấu thực hiện lặp lại cho đến khi không còn ô nào để đánh dấu.

Chú ý:

- Chỉ xét những ô chưa đánh dấu (đảm bảo mỗi ô chỉ xét đúng một lần, tránh lặp vô hạn).
- Mỗi ô đánh dấu có thể phát sinh tối đa 4 ô liền kề cùng màu, cần lưu các ô phát sinh này vào hàng đợi và lần lượt lấy từng ô trong hàng đợi ra để xử lí.

Bảng 5 là một ví dụ về bộ dữ liệu mẫu và kết quả ra tương ứng

Bảng 5. Một ví dụ về bộ dữ liệu mẫu và kết quả ra tương ứng

DỮ LIỆU VÀO	KẾT QUẢ RA
4 4	1 1 7 5
1 1 7 5	1 8 8 8
1 3 3 3	3 5 5 8
3 5 5 3	2 2 8 8
2 2 3 3	
2 3 8	

TÌM HIỂU CÂY TÌM KIẾM NHỊ PHÂN TRONG SẮP XẾP VÀ TÌM KIẾM

Bài 1

GIỚI THIỆU CÂY NHỊ PHÂN

Học xong bài này, em sẽ:

- Nêu được khái niệm cây, cây nhị phân và biểu diễn được cây nhị phân bằng mảng một chiều.
- Trình bày và mô phỏng được các phép toán duyệt trước, duyệt giữa và duyệt sau cây nhị phân được cho bằng biểu diễn trực quan.



Em hãy quan sát một nhánh cây phả hệ ở Hình 1 và cho biết Bình phải xưng hô với An như thế nào?

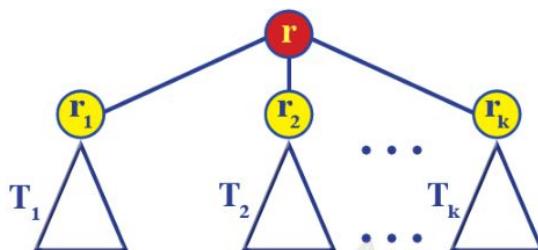


Hình 1. Một nhánh cây phả hệ bên họ nội của An và Bình

1 Khái niệm cây

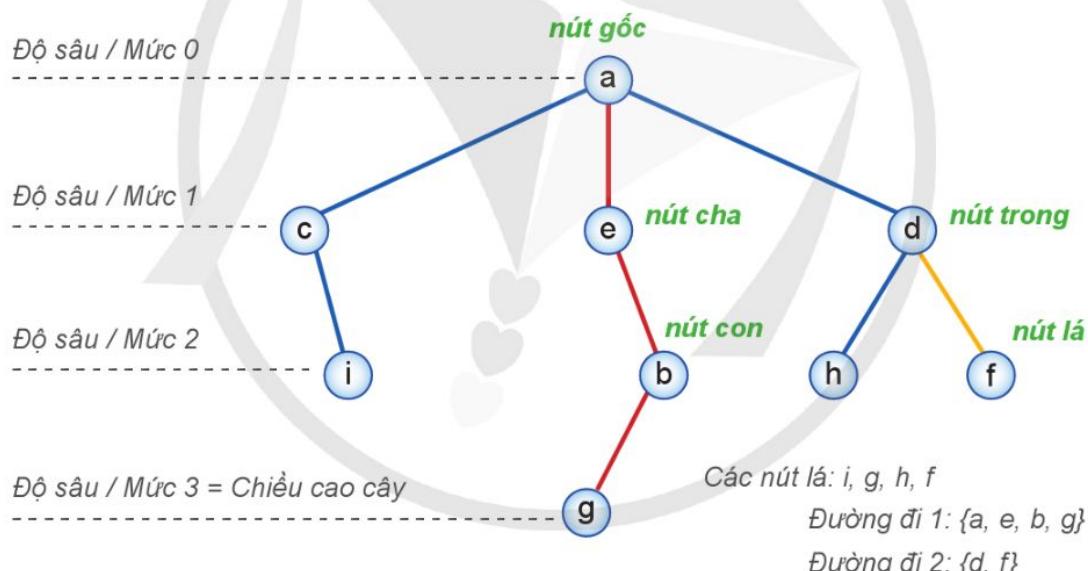
Cây phả hệ trong hoạt động khởi động ở trên được biểu diễn bởi cấu trúc dữ liệu cây trong tin học. Cấu trúc dữ liệu cây là một cấu trúc phân cấp gồm một tập hợp hữu hạn các nút, giữa các nút có mối quan hệ phân cấp gọi là quan hệ cha-con. Trên cây có một nút đặc biệt gọi là *nút gốc*. Mỗi nút có thể không có hoặc có ít nhất một *nút con* nhưng chỉ có duy nhất một *nút cha*, ngoại trừ nút gốc là không có nút cha. Cây không chứa nút nào được gọi là *cây rỗng*. Cây không rỗng có thể được định nghĩa đê quy như sau:

- **Bước cơ sở:** Cây có duy nhất một nút r và r được gọi là *nút gốc* của cây này.
- **Bước quy nạp:** Giả sử T_1, T_2, \dots, T_k là các cây với nút gốc lần lượt là r_1, r_2, \dots, r_k . Ta có thể xây dựng cây mới bằng cách đặt r làm *nút cha* của các nút r_1, r_2, \dots, r_k . Trong cây này, r là nút gốc và T_1, T_2, \dots, T_k là các *cây con* của nút gốc r . Các nút r_1, r_2, \dots, r_k được gọi là *nút con* của nút r ; mỗi nút con này được nối với r bởi một cạnh. *Hình 2* thể hiện cấu trúc đệ quy của cây.



Hình 2. Cấu trúc đệ quy của cây

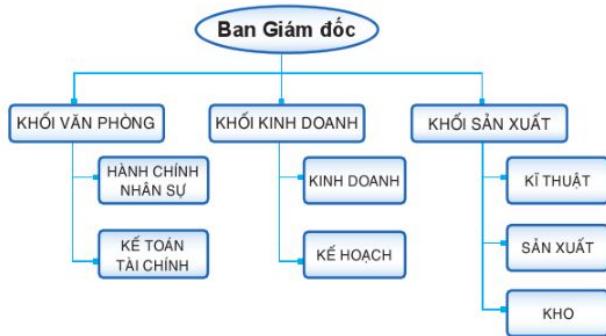
Trên cây có *nút trong* và *nút lá*. Nút trong là nút có ít nhất một nút con, nút lá là nút không có nút con nào (*Hình 3*). Thông thường mỗi nút của cây được gán một *khoá* là một *giá trị* liên quan đến thông tin cần được cất giữ trong nút đó.



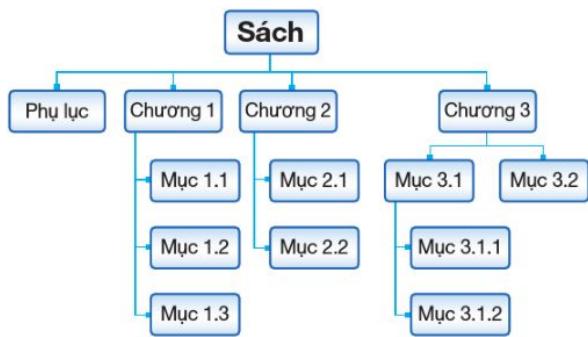
Hình 3. Minh họa các thuật ngữ trên cây

Đường đi là một dãy các cạnh liên tiếp nhau trên cây. Nếu n_1, n_2, \dots, n_k là dãy nút đối nhau trên cây sao cho nút n_i là cha của nút n_{i+1} với $1 \leq i < k$, thì dãy này biểu diễn đường đi duy nhất từ nút n_1 tới nút n_k . *Độ sâu* hay *mức* của một nút là số lượng cạnh trên đường đi duy nhất từ gốc r đến nút đó. *Chiều cao* của cây là độ sâu của nút lá có độ sâu lớn nhất.

Cây có rất nhiều ứng dụng thực tế như: cây phả hệ, sơ đồ lịch thi đấu, sơ đồ phân cấp quản lí (*Hình 4*), mục lục của một quyển sách (*Hình 5*).



Hình 4. Một cây phân cấp quản lý



Hình 5. Ví dụ cây mục lục một quyển sách

Thực hành: Em hãy vẽ cây biểu diễn phân cấp một phần các thư mục trong máy tính của mình để mô tả cấu trúc tổ chức lưu trữ thư mục mẹ, thư mục con và các tệp trong máy tính. Em hãy tính chiều cao của cây và chỉ ra nút nào là nút lá, nút trong, nút gốc.

2 Khái niệm cây nhị phân



Đội tuyển Argentina đã giành chức vô địch World Cup 2022. Dựa vào hình minh họa ở *Hình 6*, em hãy vẽ cây trong tin học biểu diễn kết quả thi đấu World Cup 2022 các trận đấu từ vòng đấu loại 1:16 đến hết trận chung kết với cấu trúc như sau:

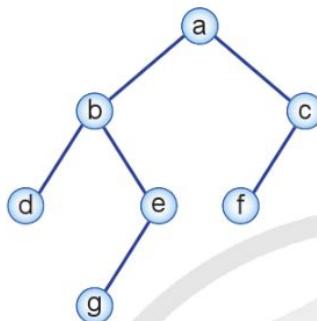
- Gốc của cây là đội vô địch.
- Mỗi nút có đúng hai nút con tương ứng với hai đội tham gia trận đấu loại trực tiếp. Khoá của các nút trong là tên đội bóng giành chiến thắng. Khoá của các nút lá là tên các đội bóng ghép đấu với nhau ở vòng 1:16.



Hình 6.
Lịch các vòng
đấu loại

Hình 7 là một ví dụ các nút biểu diễn trận đấu giữa Brasil và Hàn Quốc mà Brasil là đội giành chiến thắng; sau đó là trận đấu giữa Brasil và Croatia mà Croatia là đội giành chiến thắng.

Cây nhị phân là cây mà mỗi nút có nhiều nhất là hai nút con (Hình 8).



Hình 8. Một số ví dụ cây nhị phân

Vì mỗi nút chỉ có không quá hai nút con, ta gọi tương ứng chúng là *nút con trái* và *nút con phải* (Hình 9) của nút đó. Nút con trái là nút gốc của cây con trái và nút con phải là nút gốc của cây con phải. Như vậy, mỗi nút của cây nhị phân hoặc là không có nút con, hoặc chỉ có nút con trái, hoặc chỉ có nút con phải, hoặc có cả nút con trái và nút con phải. Ta coi một nút không có nút con trái là nút có cây con trái là cây rỗng, một nút không có nút con phải là nút có cây con phải là cây rỗng. Nút lá là nút có hai cây con trái và phải đều rỗng.

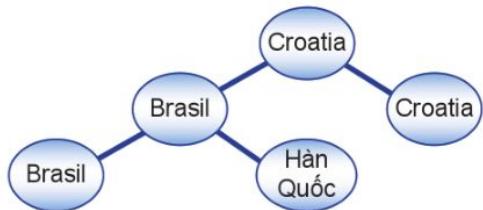
Cây nhị phân hoàn chỉnh (Hình 10) là cây nhị phân thoả mãn:

- Tất cả các mức, ngoại trừ mức cuối cùng, đều phải lấp đầy hoàn toàn các nút.
- Tất cả các nút ở mức cuối cùng phải nằm lệch sang trái nhất có thể được.

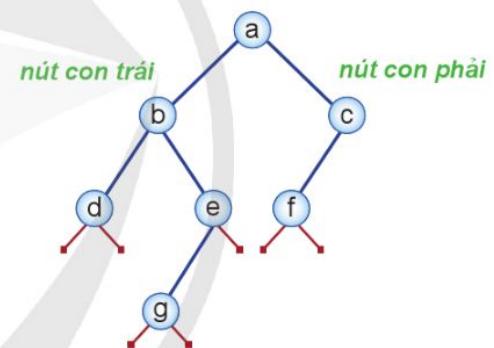
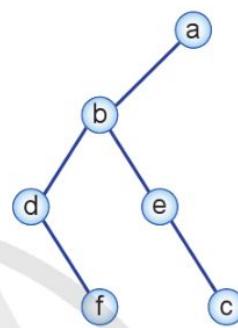
Cây nhị phân hoàn hảo là cây nhị phân thoả mãn:

- Mỗi nút trong có đúng 2 nút con.
- Tất cả các nút lá ở cùng một độ sâu.

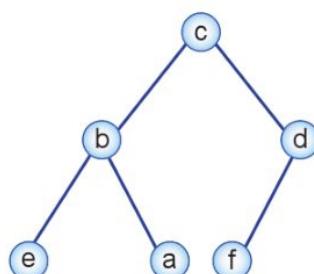
Như vậy, một cây nhị phân hoàn hảo chiều cao h sẽ có đúng $2^{h+1} - 1$ nút. Hình 11 là một ví dụ về cây nhị phân hoàn hảo.



Hình 7. Minh họa các nút cây nhị phân
biểu diễn trận đấu Brasil – Hàn Quốc
và trận đấu Brasil – Croatia



Hình 9. Ví dụ nút con trái và
nút con phải trong cây nhị phân



Hình 10. Ví dụ một cây
nhị phân hoàn chỉnh

3 Biểu diễn cây nhị phân bằng mảng một chiều

Xét cây nhị phân hoàn hảo T có n nút, trong đó mỗi nút chứa một giá trị. Gán nhãn cho các nút của cây T từ trên xuống dưới và từ trái qua phải bằng các chỉ số $0, 1, \dots, n - 1$. Đặt tương ứng cây T với mảng một chiều $A[0, \dots, n - 1]$, trong đó phần tử $A[i]$ ($0 \leq i \leq n - 1$) là giá trị cát giữ trong nút có nhãn là i trên cây T (Hình 11).

Do cây nhị phân có mối quan hệ cha-con nên với mỗi nút có nhãn là i ta cần xác định được nhãn của nút cha, nút con trái, nút con phải của nó. Từ cách gán nhãn nêu trên, với mỗi nút có nhãn i dễ dàng xác định được:

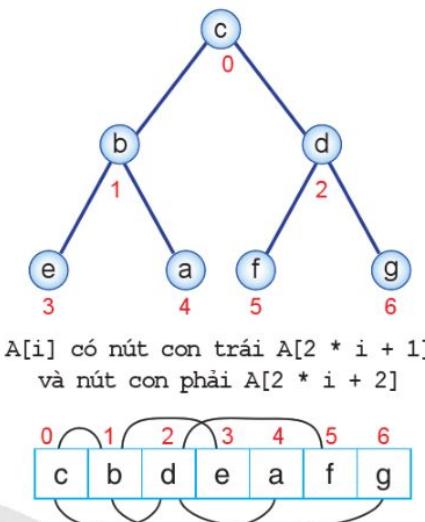
- Nhãn nút cha là $[(i - 1) / 2]$ với $i \geq 1$.
- Nhãn nút con trái là $2 * i + 1$ với $2 * i + 1 < n$.
- Nhãn nút con phải là $2 * i + 2$ với $2 * i + 2 < n$.

Ví dụ, với cây ở Hình 11, nút với nhãn là 1 có nút cha là $[(1 - 1) / 2] = 0$, nút con trái có nhãn là $2 * 1 + 1 = 3$, nút con phải có nhãn là $2 * 1 + 2 = 4$.

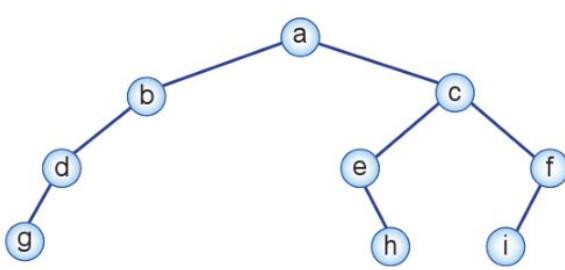
Một cây nhị phân tổng quát có chiều cao h được biến đổi thành cây nhị phân hoàn chỉnh bằng cách thêm các nút giả vào mỗi mức để mỗi mức có đầy đủ các nút; ngoại trừ mức h không được có các nút giả cuối cùng (các nút sát phải).

Do đó, ta có thể sử dụng mảng một chiều để biểu diễn cây nhị phân. Lưu ý rằng số lượng nút giả được bổ sung có thể rất lớn. Trong trường hợp tồi nhất là cây có n nút mà các nút trong chỉ có một nút con phải, nghĩa là cây suy biến (lệch phải), chiều cao của cây là $h = n - 1$. Sau khi biến đổi cây này thành cây nhị phân hoàn chỉnh bằng cách thêm vào các nút giả; cây này có n nút thật dùng để lưu trữ dữ liệu và có $2^n - 1 - n$ nút giả (không lưu trữ dữ liệu).

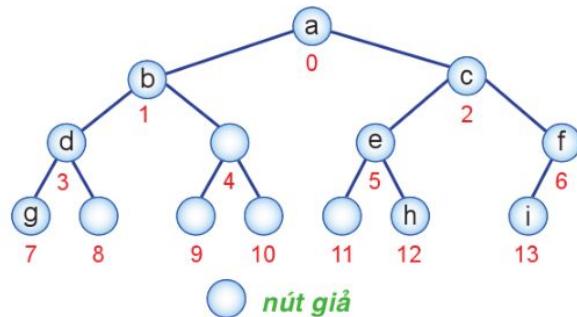
Hình 12a là ví dụ một cây nhị phân và Hình 12b là cây nhị phân sau khi thêm các nút giả vào cây ở Hình 12a để trở thành cây nhị phân hoàn chỉnh. Mảng một chiều biểu diễn cây nhị phân hoàn chỉnh đó gồm các phần tử có giá trị lần lượt là: [a, b, c, d, None, e, f, g, None, None, None, h, i]. Nút giả có giá trị khoá là None.



Hình 11. Ví dụ biểu diễn một cây nhị phân hoàn hảo bởi mảng



Hình 12a. Ví dụ một cây nhị phân



Hình 12b. Cây nhị phân sau khi bổ sung mít giả cho cây ở Hình 12a

④ Duyệt cây nhị phân

Các nút con của một nút được xếp thứ tự *từ trái sang phải*. Như vậy, hai cây trong *Hình 13* sau đây là *khác nhau*, bởi vì hai con của nút *a* xuất hiện trong hai cây theo thứ tự khác nhau.

Duyệt cây là cách thăm tất cả các nút, mỗi nút đúng một lần theo một thứ tự nhất định. Ta có thể duyệt thứ tự các nút của cây theo nhiều cách. Có ba thứ tự quan trọng nhất, đó là: *thứ tự trước*, *thứ tự sau* và *thứ tự giữa*.

Xét một cây nhị phân *T* có nút gốc *r* với cây con trái *T₁* và cây con phải *T₂* (*Hình 14*). Tuỳ thứ tự duyệt nút gốc trước, sau và giữa, cách duyệt cây *T* theo ba thứ tự trên được mô tả một cách đê quy như sau:

a) Duyệt theo thứ tự trước

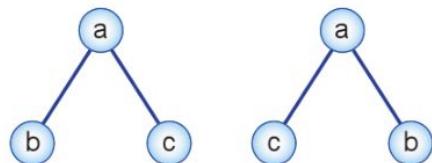
Thứ tự trước cho các nút của *T* là:

- Nút gốc *r* của *T*.
- Tiếp theo là các nút của *T₁* theo thứ tự trước.
- Cuối cùng là các nút của *T₂* theo thứ tự trước.

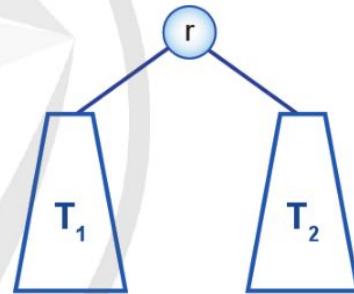
Quá trình mô phỏng duyệt cây nhị phân trong *Hình 15* như sau:

Bắt đầu từ duyệt cây gốc *a*:

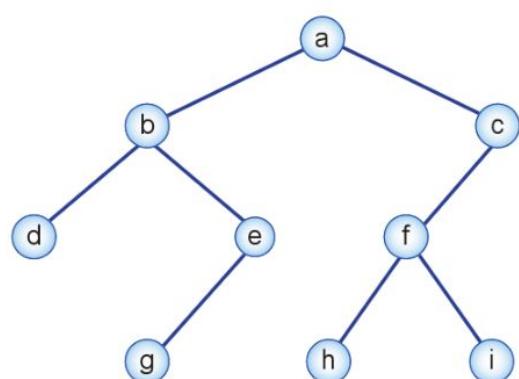
- Đưa ra giá trị khoá nút gốc *a*.
- Duyệt cây con gốc *b* theo thứ tự trước:
 - Đưa ra giá trị khoá nút gốc *b*.
 - Duyệt cây con gốc *d*:
 - Đưa ra giá trị khoá nút gốc *d*.
 - Duyệt cây con gốc *e*:
 - Đưa ra giá trị khoá nút gốc *e*.



Hình 13. Ví dụ hai cây khác nhau



Hình 14. Cấu trúc đê quy cây nhị phân



Hình 15. Ví dụ duyệt cây nhị phân

- Duyệt cây con gốc g :
 - Đưa ra giá trị khoá nút gốc g .
- Duyệt cây con gốc c :
 - Đưa ra giá trị khoá nút gốc c .
 - Duyệt cây con gốc f :
 - Đưa ra giá trị khoá nút gốc f .
 - Duyệt cây con gốc h :
 - Đưa ra giá trị khoá nút gốc h .
 - Duyệt cây con gốc i :
 - Đưa ra giá trị khoá nút gốc i .

Dãy các nút được liệt kê thứ tự trước của cây trong *Hình 15* là:

$a, b, d, e, g, c, f, h, i$.

b) Duyệt theo thứ tự sau

Thứ tự sau cho các nút của T là:

- Các nút của T_1 theo thứ tự sau.
- Tiếp theo là các nút của T_2 theo thứ tự sau.
- Cuối cùng là nút gốc r của T .

Thực hành: Dựa vào cách mô phỏng trong phần duyệt theo thứ tự trước, em hãy mô phỏng lại quá trình duyệt cây nhị phân trong *Hình 15* theo thứ tự sau. Dãy các nút sau khi được liệt kê thứ tự sau của cây trong *Hình 15* sẽ là: $d, g, e, b, h, i, f, c, a$.

c) Duyệt theo thứ tự giữa

Thứ tự giữa cho các nút của T là:

- Các nút của T_1 theo thứ tự giữa.
- Tiếp theo là nút gốc r của T .
- Cuối cùng là các nút của T_2 theo thứ tự giữa.

Thực hành: Dựa vào cách mô phỏng trong phần duyệt theo thứ tự trước, em hãy mô phỏng lại quá trình duyệt cây nhị phân trong *Hình 15* theo thứ tự giữa. Dãy các nút sau khi được liệt kê thứ tự giữa của cây trong *Hình 15* là: $d, b, g, e, a, h, f, i, c$.

Lưu ý: Nếu cây T là rỗng thì danh sách rỗng, hoặc nếu cây T chỉ có một nút thì nút đó chính là danh sách các nút được duyệt theo bất kì thứ tự trước hoặc sau hoặc giữa của cây T .

Để nhớ cách đưa ra các nút theo ba thứ tự vừa trình bày, hãy hình dung là ta đi vòng quanh bên ngoài cây bắt đầu từ gốc, ngược chiều kim đồng hồ và sát theo cây nhất. Chẳng hạn, đường đi đó đối với cây trong *Hình 15* được mô phỏng như trong *Hình 16*. Khi đó:

- *Đối với thứ tự trước, đưa ra nút mỗi khi đi qua nút đó.*

- Đối với thứ tự sau, đưa ra nút khi qua nó ở lần cuối trước khi quay về cha của nút đó.

- Đối với thứ tự giữa, đưa ra nút lá ngay khi đi qua nút đó, còn những nút trong được đưa ra khi lần thứ hai được đi qua.



Em hãy đưa ra danh sách các thành viên gia đình có mối quan hệ được biểu diễn dưới dạng cây trong *Hình 17* (còn được gọi là cây phả hệ ngược) theo các thứ tự trước, sau và giữa.

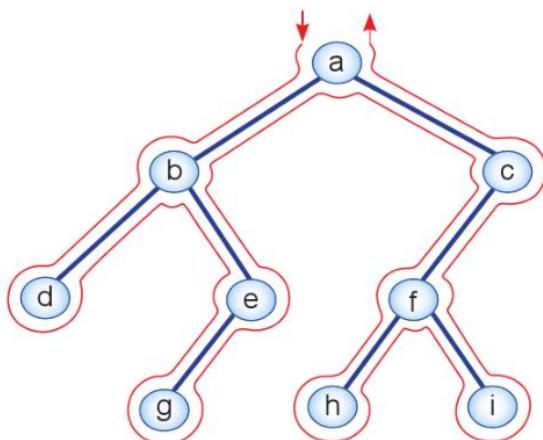


Câu 1. Trong các câu sau, những câu nào SAI?

- Nút trong của cây có ít nhất một nút con.
- Nút lá của cây là nút không có nút con.
- Sử dụng các cách duyệt cây khác nhau trên cùng một cây sẽ cho cùng một dãy kết quả.
- Số lượng nút giả cần bổ sung thêm trong trường hợp tồi nhất khi dùng mảng một chiều để biểu diễn cây nhị phân có chiều cao h là $2^{h+1} - h - 2$ nút.

Câu 2. Tổng số nút của một cây nhị phân hoàn hảo chiều cao 4 là bao nhiêu?

- A. 7. B. 8. C. 15. D. 16.



Hình 16. Một mô phỏng quá trình duyệt cây nhị phân



Hình 17. Mối quan hệ giữa các thành viên gia đình bé Tùn theo dạng cây

Tóm tắt bài học

- ✓ Cấu trúc dữ liệu cây là một cấu trúc phân cấp gồm một tập hợp hữu hạn các nút, giữa các nút có mối quan hệ phân cấp gọi là quan hệ cha-con. Trên cây có một nút đặc biệt gọi là nút gốc.
- ✓ Cây nhị phân là cây mà mỗi nút có nhiều nhất là hai nút con.
- ✓ Duyệt cây là cách thăm tất cả các nút, mỗi nút đúng một lần theo một thứ tự nhất định. Có ba thứ tự duyệt hay sử dụng nhất, đó là: duyệt theo thứ tự trước, duyệt theo thứ tự sau và duyệt theo thứ tự giữa.

Bài 2

THỰC HÀNH DUYỆT CÂY NHỊ PHÂN

Học xong bài này, em sẽ:

- Viết được chương trình biểu diễn cây nhị phân bằng mảng một chiều.
- Viết được chương trình duyệt cây nhị phân theo thứ tự trước, sau và thứ tự giữa sử dụng mảng một chiều.

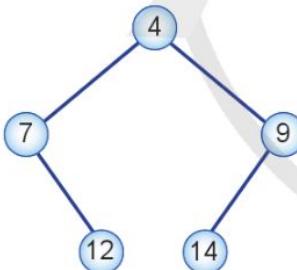
Yêu cầu: Sử dụng mảng một chiều để xây dựng cây nhị phân và duyệt cây theo các thứ tự trước, thứ tự sau và thứ tự giữa.

Trong các bài thực hành sau đây, em hãy thay ví dụ cây nhị phân và thông tin khoá các nút của cây để thực hành viết chương trình trên Python theo hướng dẫn.

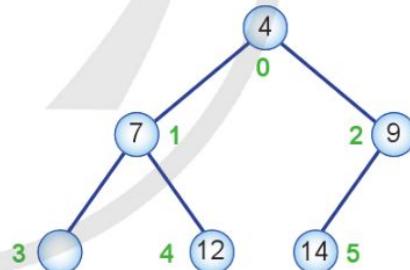
Thực hành 1: Xây dựng cây nhị phân từ một danh sách các phần tử cho trước
Hướng dẫn:

Bước 1. Bổ sung các nút giả vào cây để trở thành cây nhị phân hoàn chỉnh.

Bước 2. Biểu diễn cây nhị phân hoàn chỉnh bằng kiểu dữ liệu mảng một chiều *list*.



Hình 1a. Một ví dụ cây nhị phân với 5 nút



Hình 1b. Thực hiện Bước 1 trên cây ở Hình 1a

Hình 1b biểu diễn cây nhị phân hoàn chỉnh thu được sau khi tiến hành bổ sung các nút giả vào cây nhị phân ở Hình 1a. Để biểu diễn cây nhị phân hoàn chỉnh ở Hình 1b bằng mảng một chiều *list*, ta đánh chỉ số các nút trên cây như sau:

- Nút gốc có chỉ số 0.
- Đối với nút có chỉ số i , nút con trái có chỉ số $2 * i + 1$ và nút con phải có chỉ số $2 * i + 2$.

Em hãy tham khảo chương trình trong Hình 2 sử dụng *list* để biểu diễn cây nhị phân trong Hình 1b:

```

File Edit Format Run Options Window Help
1 # Bổ sung các nút giả với giá trị khoá None vào cây nhị phân
2 # ban đầu để thành cây nhị phân hoàn chỉnh:
3 tree = [4, 7, 9, None, 12, 14]
4 # Đưa ra danh sách giá trị khoá các nút từ cây nhị phân
5 print("\nDanh sách giá trị khoá các nút từ cây nhị phân:", tree)

```

Hình 2. Chương trình in ra danh sách nút

Màn hình kết quả:

Danh sách giá trị khoá các nút từ cây nhị phân: [4, 7, 9, None, 12, 14]

Thực hành 2: Sử dụng kiểu dữ liệu list trong Python xây dựng cây nhị phân bằng cách bổ sung từng nút vào cây

Hướng dẫn:

- Tạo cây nhị phân hoàn chỉnh từ việc nhập vào một dãy số từ bàn phím.
- Giả sử nút thật có giá trị là số dương và nút giả có giá trị là 0. Cây rỗng tương ứng với `dsNut[0] = 0`.
- Dữ liệu nhập để tạo cây nhị phân tổng quát ở *Hình 1a* là dãy các số 4, 7, 9, 0, 12, 14. Chương trình sau đây sẽ tạo cây nhị phân hoàn chỉnh từ cây nhị phân tổng quát sau khi thêm vào các nút giả.

Em hãy tham khảo chương trình trong *Hình 3* sau đây:

```

File Edit Format Run Options Window Help
1 # Tạo cây nhị phân hoàn chỉnh từ mảng dsNut không có các số 0 ở
2 # cuối mảng
3 def createTree(dsNut):
4     if dsNut == [] or dsNut[0] == 0:
5         return []           # Trả về cây rỗng
6     # Tạo cây từ mảng dsNut
7     tree = []
8     for i in range(len(dsNut)):
9         if dsNut[i] == 0:
10            tree.append(None)
11        else:
12            tree.append(dsNut[i])
13    return tree
14 print("Nhập danh sách các số để tạo cây: ")
15 dsNut = list(map(int, input().split()))
16 tree = createTree(dsNut)
17 # Đưa ra danh sách giá trị khoá các nút từ cây nhị phân
18 print("\nDanh sách giá trị khoá các nút từ cây nhị phân:", tree)

```

Hình 3. Chương trình tạo cây nhị phân hoàn chỉnh từ mảng dsNut

Màn hình kết quả:

Danh sách giá trị khoá các nút từ cây nhị phân: [4, 7, 9, None, 12, 14]

Thực hành 3: Duyệt cây theo thứ tự trước, thứ tự sau và thứ tự giữa

Hướng dẫn:

Các bước thuật toán và hàm đệ quy dưới đây thực hiện các phép duyệt cây nhị phân theo thứ tự trước, sau và giữa theo phương pháp được mô tả trong Mục 4, Bài 1.

a) Thứ tự trước

Mã giả mô tả thuật toán đệ quy duyệt theo thứ tự trước (gốc – trái – phải) cây nhị phân T trong *Hình 4*:

```
hàm ThuTuTruoc(cây T gốc r):
    nếu cây T gốc r khác rỗng thì:
        Thăm nút gốc r # gốc
        ThuTuTruoc(cây con trái của nút r) # trái
        ThuTuTruoc(cây con phải của nút r) # phải
```

Hình 4. Mã giả mô tả thuật toán đệ quy duyệt theo thứ tự trước

Em hãy tham khảo chương trình trong *Hình 5* sau đây:

```
File Edit Format Run Options Window Help
1 # Hàm duyệt cây theo thứ tự trước;
2 # i là chỉ số trong tree, là nút gốc của cây con đang duyệt
3 def ThuTuTruoc(tree, i):
4     if i<len(tree) and tree[i] != None: # cây khác rỗng
5         # chỉ dẫn end = " " để tránh xuống dòng sau mỗi lệnh in giá
6         # trị khoá
7         print(tree[i],end = " ") # gốc
8         ThuTuTruoc(tree,2*i+1) # trái
9         ThuTuTruoc(tree,2*i+2) # phải
10    tree = [4, 7, 9, None, 12, 14]
11    print("Danh sách giá trị khoá các nút cây nhị phân theo thứ tự trước:")
12    ThuTuTruoc(tree,0)
```

Hình 5. Chương trình duyệt cây theo thứ tự trước

Màn hình kết quả:

Danh sách giá trị khoá các nút cây nhị phân theo thứ tự trước: 4 7 12 9 14

b) Thứ tự sau

Mã giả mô tả thuật toán đệ quy duyệt theo thứ tự sau (trái – phải – gốc) cây nhị phân T trong *Hình 6*:

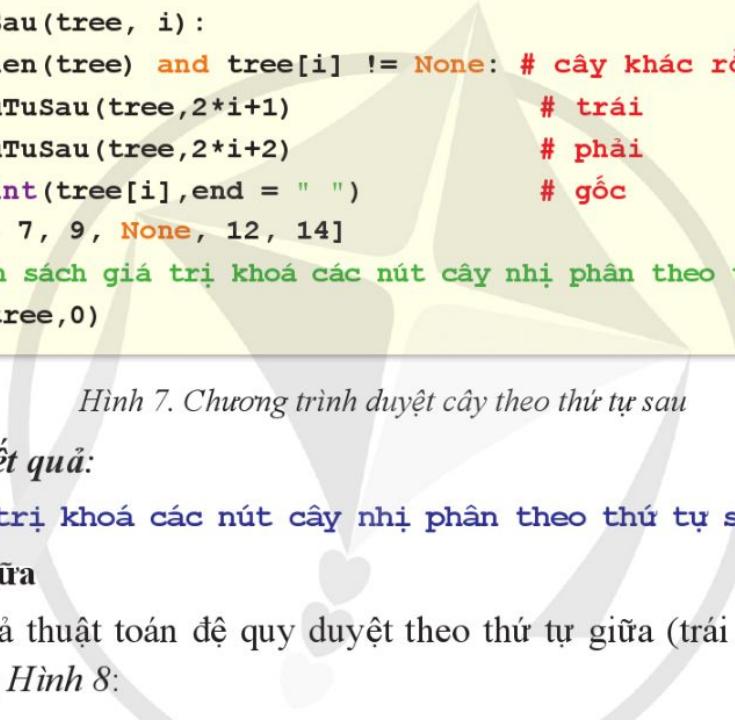
```

hàm ThuTuSau(cây T gốc r):
    nếu cây T gốc r khác rỗng thì:
        ThuTuSau(cây con trái của nút r) # trái
        ThuTuSau(cây con phải của nút r) # phải
        Thăm nút gốc r                      # gốc

```

Hình 6. Mã giả mô tả thuật toán để quy duyệt theo thứ tự sau

Em hãy tham khảo chương trình trong *Hình 7*:



```

File Edit Format Run Options Window Help
1 # Hàm duyệt cây theo thứ tự sau;
2 # i là chỉ số trong tree, là nút gốc của cây con đang duyệt
3 def ThuTuSau(tree, i):
4     if i<len(tree) and tree[i] != None: # cây khác rỗng
5         ThuTuSau(tree,2*i+1)           # trái
6         ThuTuSau(tree,2*i+2)           # phải
7         print(tree[i],end = " ")
8 tree = [4, 7, 9, None, 12, 14]
9 print("Danh sách giá trị khoá các nút cây nhị phân theo thứ tự sau:")
10 ThuTuSau(tree,0)

```

Hình 7. Chương trình duyệt cây theo thứ tự sau

Màn hình kết quả:

Danh sách giá trị khoá các nút cây nhị phân theo thứ tự sau: 12 7 14 9 4

c) Thứ tự giữa

Mã giả mô tả thuật toán để quy duyệt theo thứ tự giữa (trái – gốc – phải) cây nhị phân *T* trong *Hình 8*:

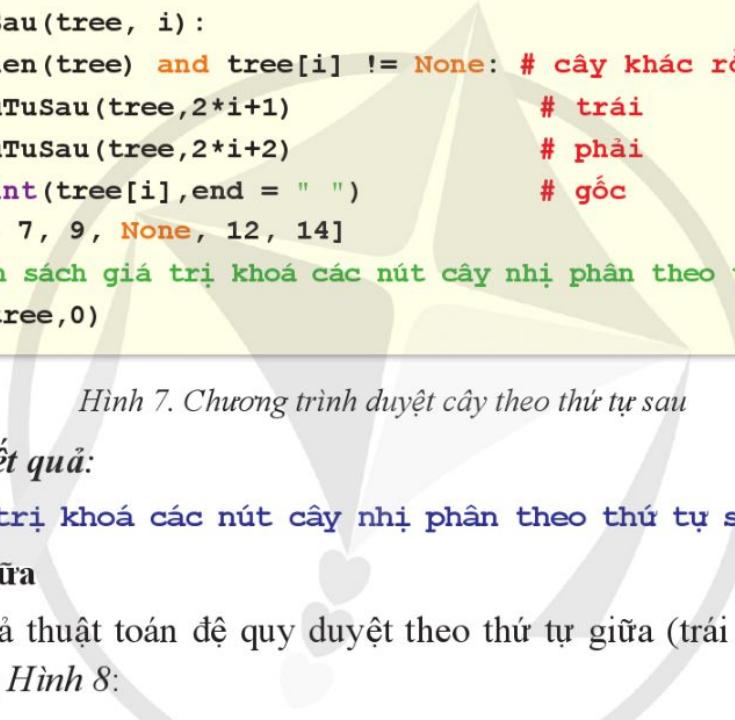
```

hàm ThuTuGiua(cây T gốc r):
    nếu cây T gốc r khác rỗng thì:
        ThuTuGiua(cây con trái của nút r) # trái
        thăm nút gốc r                  # gốc
        ThuTuGiua(cây con phải của nút r) # phải

```

Hình 8. Mã giả mô tả thuật toán để quy duyệt theo thứ tự giữa

Em hãy tham khảo chương trình trong *Hình 9*:



```

File Edit Format Run Options Window Help
1 # Hàm duyệt cây theo thứ tự giữa;
2 # i là chỉ số trong tree, là nút gốc của cây con đang duyệt
3 def ThuTuGiua(tree, i):
4     if i<len(tree) and tree[i] != None: # cây khác rỗng

```

```

5     ThuTuGiua(tree,2*i+1)           # trái
6     print(tree[i],end = " ")        # gốc
7     ThuTuGiua(tree,2*i+2)           # phải
8 tree = [4, 7, 9, None, 12, 14]
9 print("Danh sách giá trị khoá các nút cây nhị phân theo thứ tự giữa:")
10 ThuTuGiua(tree,0)

```

Hình 9. Chương trình duyệt cây theo thứ tự giữa

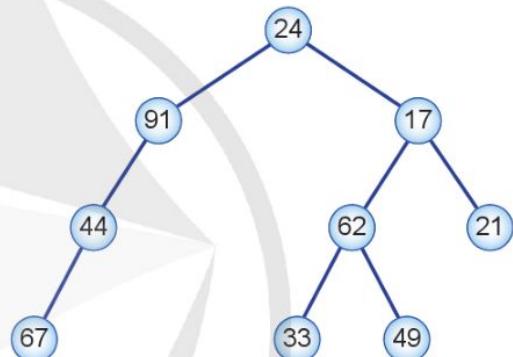
Màn hình kết quả:

Danh sách giá trị khoá các nút cây nhị phân theo thứ tự giữa: 7 12 4 14 9



Từ ví dụ cây nhị phân tổng quát trong *Hình 10* dưới đây, em hãy lập trình thực hiện các nhiệm vụ sau:

- Bổ sung các nút giả và xây dựng cây nhị phân hoàn chỉnh bằng cách thêm từng nút giả vào cây. Khoá các nút nhập vào lần lượt từ bàn phím, nút giả có giá trị là 0.
- Đưa ra danh sách các nút từ cây nhị phân vừa xây dựng theo các thứ tự trước, thứ tự sau và thứ tự giữa.



Hình 10. Ví dụ cây nhị phân có 4 mức

BÀI TÌM HIỂU THÊM

THƯ VIỆN BINARYTREE

Binarytree là một thư viện của Python cho phép xây dựng cây nhị phân, hiển thị trực quan, kiểm tra và điều khiển các thao tác trên cây nhị phân. Học sinh có thể sử dụng thư viện này như là một công cụ để kiểm tra lại kết quả các bài thực hành về cây nhị phân. Những hướng dẫn sau đây tham khảo tại nguồn <https://www.geeksforgeeks.org/binarytree-module-in-python/>.

Để sử dụng thư viện này, ở đầu chương trình cần đặt định hướng khai báo thư viện theo cách thức cây nhị phân cần xây dựng, ví dụ:

```
from binarytree import Node, tree, build
```

Nếu trên máy tính chưa cài đặt thư viện binarytree thì cần tiến hành cài đặt trước khi sử dụng, ví dụ sử dụng câu lệnh sau trong cửa sổ nhận lệnh Terminal:

```
pip install binarytree
```

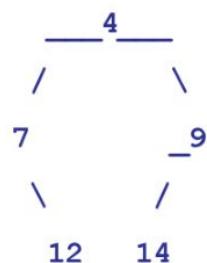
Chương trình mẫu trong Hình 11 xây dựng cây nhị phân từ một danh sách các phần tử cho trước và duyệt cây theo thứ tự trước, thứ tự sau và thứ tự giữa.

```
File Edit Format Run Options Window Help
1 from binarytree import build
2 # Danh sách giá trị khoá các nút trong mảng một chiều biểu diễn cây
3 # nhị phân hoàn chỉnh,
4 # các giá trị khoá None là nút giả thêm vào cây nhị phân ban đầu
5 # để thành cây nhị phân hoàn chỉnh, ví dụ:
6 tree = [4, 7, 9, None, 12, 14]
7 # Xây dựng cây nhị phân từ danh sách tree,
8 # kết quả trả về nút gốc của cây được xây dựng
9 nut_goc = build(tree)
10 print("Cây nhị phân từ danh sách giá trị khoá các nút:\n", nut_goc)
11 # Đưa ra danh sách giá trị khoá các nút cây nhị phân
12 print("\nDanh sách giá trị khoá các nút cây nhị phân:", nut_goc.values)
13 # Duyệt cây theo thứ tự trước
14 print("\nDanh sách giá trị khoá các nút theo thứ tự trước:", nut_goc.preorder)
15 # Duyệt cây theo thứ tự sau
16 print("\nDanh sách giá trị khoá các nút theo thứ tự sau:", nut_goc.postorder)
17 # Duyệt cây theo thứ tự giữa
18 print("\nDanh sách giá trị khoá các nút theo thứ tự giữa:", nut_goc.inorder)
```

Hình 11. Chương trình mẫu xây dựng cây nhị phân và duyệt cây

Màn hình kết quả:

Cây nhị phân từ danh sách giá trị khoá các nút:



Danh sách giá trị khoá các nút cây nhị phân: [4, 7, 9, None, 12, 14]

Danh sách giá trị khoá các nút theo thứ tự trước: [Node(4), Node(7), Node(12), Node(9), Node(14)]

Danh sách giá trị khoá các nút theo thứ tự sau: [Node(12), Node(7), Node(14), Node(9), Node(4)]

Danh sách giá trị khoá các nút theo thứ tự giữa: [Node(7), Node(12), Node(4), Node(14), Node(9)]

Bài 3

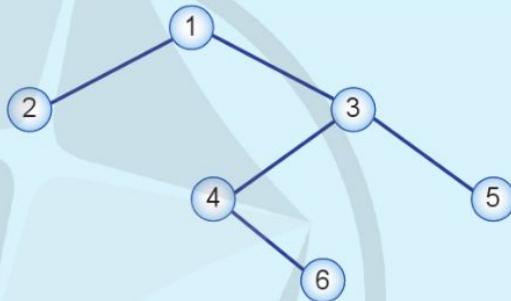
CÂY TÌM KIẾM NHỊ PHÂN

Học xong bài này, em sẽ:

- Trình bày được khái niệm Cây tìm kiếm nhị phân.
- Mô phỏng được thuật toán tạo Cây tìm kiếm nhị phân biểu diễn một tập số nguyên dương và thuật toán xác định một giá trị đã cho có thuộc tập hợp đó hay không.



Em hãy đưa ra danh sách giá trị khoá các nút ở cây nhị phân *Hình 1* trong phép duyệt cây theo thứ tự giữa và nhận xét đặc điểm của cây nhị phân cùng danh sách được đưa ra này.



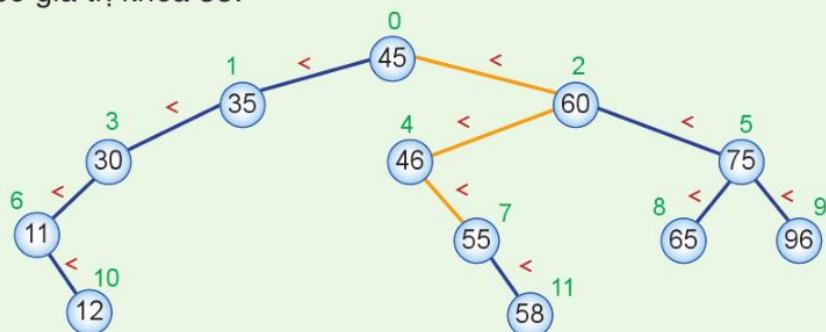
Hình 1. Ví dụ cây nhị phân

1 Khái niệm cây tìm kiếm nhị phân



1

An và Hoà cùng tham gia một hoạt động về bài toán trên dãy số. Giả sử các phần tử 45, 35, 60, 30, 46, 75, 11, 55, 65, 96, 12, 58 của tập số An giữ được biểu diễn dưới dạng cây nhị phân trong *Hình 2*, các nút được đánh chỉ số từ 0 đến 11. Em hãy quan sát đặc điểm của cây và các giá trị khoá từng nút của cây này, từ đó đưa ra một cách giúp Hoà đặt ít câu hỏi nhất có thể để tìm ra vị trí nút có giá trị khoá 55.



Hình 2. Ví dụ cây tìm kiếm nhị phân

Quan sát cây nhị phân ở *Hình 2*, ta có nhận xét: Giá trị khoá mỗi nút trong lớn hơn giá trị khoá của các nút thuộc cây con trái của nó (nếu có) và bé hơn giá trị khoá của các nút thuộc cây con phải của nó (nếu có). Như vậy, Hoà chỉ cần đặt bốn câu hỏi là xác định được vị trí nút có giá trị khoá 55:

Câu hỏi 1: “*Nút thứ 0 có giá trị khoá bằng, nhỏ hơn hay lớn hơn 55?*”. Trả lời: “45 nhỏ hơn 55”. Do đó, 55 chắc chắn sẽ nằm trong cây con phải của nút thứ 0.

Câu hỏi 2: “*Nút thứ 2 có giá trị khoá bằng, nhỏ hơn hay lớn hơn 55?*”. Trả lời: “60 lớn hơn 55”. Do đó 55 chắc chắn sẽ nằm trong cây con trái của nút thứ 2.

Câu hỏi 3: “*Nút thứ 4 có giá trị khoá bằng, nhỏ hơn hay lớn hơn 55?*”. Trả lời: “46 nhỏ hơn 55”. Do đó 55 chắc chắn sẽ nằm trong cây con phải của nút thứ 4.

Câu hỏi 4: “*Nút thứ 7 có giá trị khoá bằng, nhỏ hơn hay lớn hơn 55?*”. Trả lời: “55 bằng 55”. Do đó 55 là nút thứ 7.

Cây nhị phân trên là cây tìm kiếm nhị phân. Cây tìm kiếm nhị phân gồm các nút có giá trị khoá đôi một khác nhau có thể được định nghĩa đệ quy như sau:

- 1) Cây rỗng là một cây tìm kiếm nhị phân.
- 2) Mọi nút thuộc cây con trái của nút gốc có khoá nhỏ hơn khoá nút gốc.
- 3) Mọi nút thuộc cây con phải của nút gốc có khoá lớn hơn khoá nút gốc.
- 4) Cả cây con trái và cây con phải của nút gốc cũng là cây tìm kiếm nhị phân.

2 Xây dựng cây tìm kiếm nhị phân

Với một dãy A gồm n số nguyên dương, cây tìm kiếm nhị phân được xây dựng từ dãy A bằng cách chèn từng nút có khoá là giá trị từng phần tử của A vào cây bắt đầu từ cây rỗng.

Thuật toán đệ quy chèn một nút có khoá x vào cây tìm kiếm nhị phân như sau:

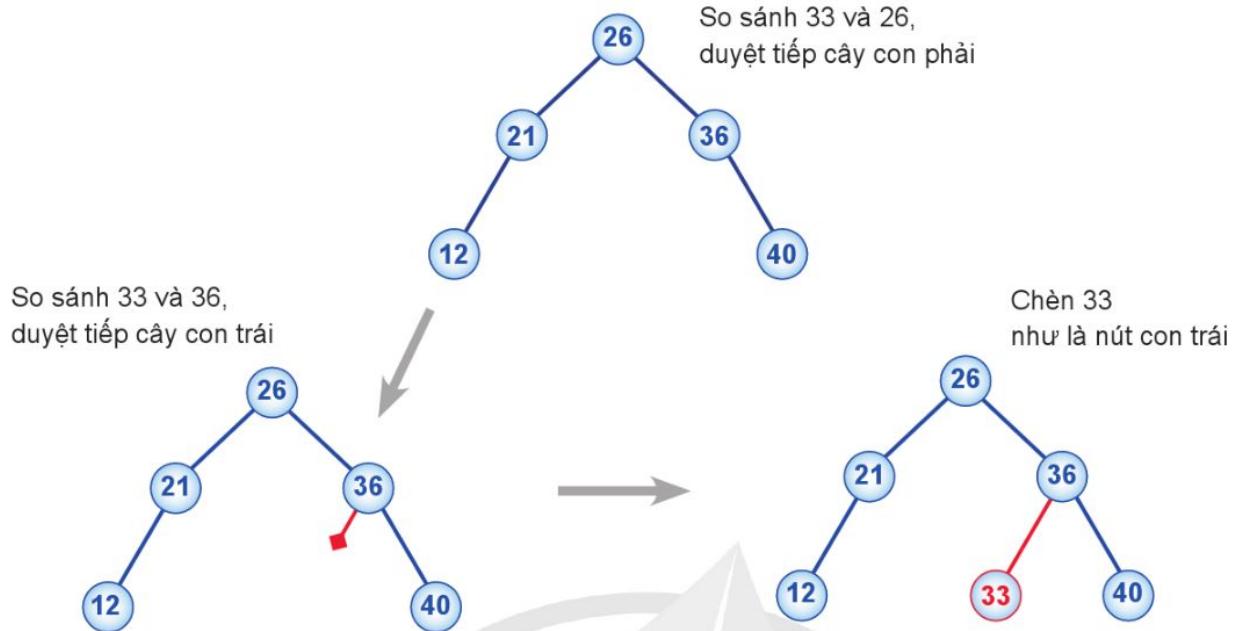
- Nếu cây rỗng, tạo nút mới là nút gốc của cây có khoá là x . Kết thúc quá trình chèn. Nút mới chèn luôn là nút lá.

- Trái lại, gọi khoá nút gốc là *khoa_nut_goc*:

- Nếu $x < \text{khoa_nut_goc}$, chèn x sang cây con trái của nút gốc.
- Nếu $x > \text{khoa_nut_goc}$, chèn x sang cây con phải của nút gốc.

Thực hiện lặp lại n lần thuật toán trên, mỗi lần với một phần tử dãy A lần lượt bắt đầu từ phần tử thứ nhất sẽ nhận được một cây tìm kiếm nhị phân.

Ví dụ: *Hình 3* sau mô tả các bước chèn nút mới có giá trị khoá 33 vào cây tìm kiếm nhị phân đang biểu diễn tập 5 số nguyên dương 26, 21, 36, 12, 40.



Hình 3. Minh họa chèn nút mới có giá trị khoá 33 vào cây

Lưu ý là ta có thể chọn bất kì thứ tự lấy các phần tử trong mảng A để chèn vào cây tìm kiếm nhị phân. Vì vậy các cách chọn thứ tự lấy phần tử trong mảng để chèn có thể tạo thành nhiều cây tìm kiếm nhị phân khác nhau. Thông thường, để dễ kiểm soát, người ta hay chọn lần lượt theo thứ tự phần tử từ đầu mảng đến cuối mảng, mỗi phần tử lấy đúng một lần.

Hình 4 là mã giả mô tả thuật toán chèn một nút có khóa x vào cây tìm kiếm nhị phân là cây *nut_goc*. Nếu cây *nut_goc* khác rỗng thì nút gốc của cây này là *nut_goc*.

```

hàm chèn(cây nut_goc, x):
    nếu cây nut_goc là rỗng thì:
        Tạo nút nut_goc có khóa là x
        trả về nút nut_goc
    trái lại
        nếu x < khóa của nút nut_goc thì:
            # Chèn x sang cây con trái của nút nut_goc
            trả về chèn(cây con trái của nút nut_goc, x)
        trái lại
        nếu x > khóa của nút nut_goc thì:
            # Chèn x sang cây con phải của nút nut_goc
            trả về chèn(cây con phải của nút nut_goc, x)
    
```

Hình 4. Mã giả mô tả thuật toán chèn

Thực hành: Từ cây tìm kiếm nhị phân trong Hình 3 sau khi đã chèn thêm nút có giá trị khoá 33, em hãy mô tả từng bước chèn một nút mới có giá trị khoá bằng 28 vào cây.

3) TÌM KIẾM NÚT TRÊN CÂY TÌM KIẾM NHỊ PHÂN

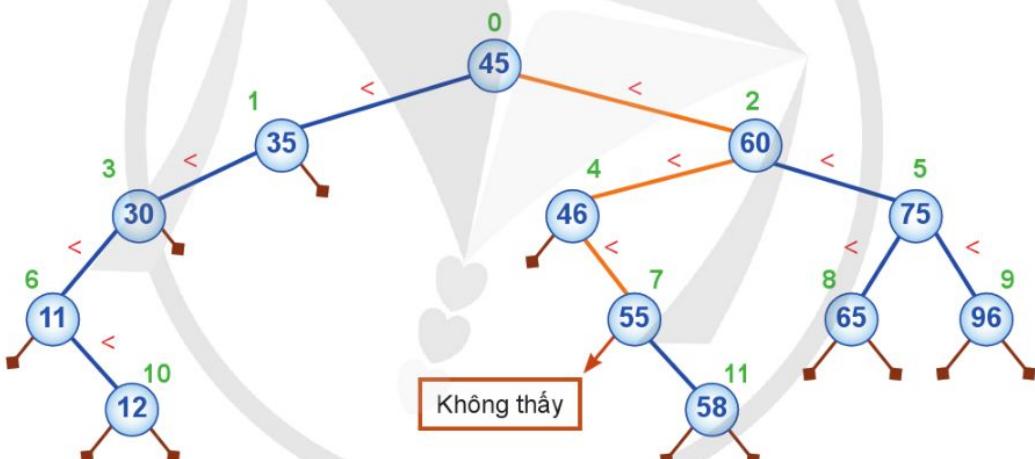
Thuật toán để quy tìm kiếm một nút có khoá x trên cây tìm kiếm nhị phân như sau:

- Nếu cây rỗng, kết thúc quá trình tìm kiếm, cây không chứa nút có khoá x cần tìm.
- Trái lại:

- Nếu $x = \text{khoa_nut_goc}$, nút gốc hiện tại là nút có khoá x cần tìm trong cây ban đầu. Kết thúc quá trình tìm kiếm.
- Nếu $x < \text{khoa_nut_goc}$, tìm kiếm x sang cây con trái của nút gốc.
- Nếu $x > \text{khoa_nut_goc}$, tìm kiếm x sang cây con phải của nút gốc.

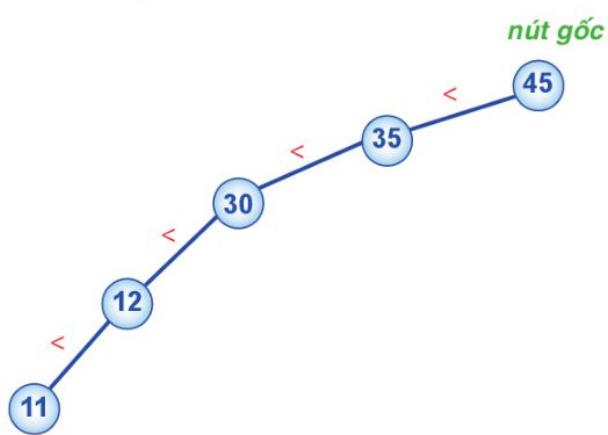
Như vậy, sau mỗi bước so sánh, quá trình tìm kiếm xét đến độ sâu tăng thêm 1. Do đó, số bước so sánh để tìm kiếm một nút trên cây tìm kiếm nhị phân không vượt quá chiều cao của cây cộng 1.

Ví dụ: Để tìm nút có giá trị khoá 55 trên cây tìm kiếm nhị phân *Hình 2*, ta cần bốn bước so sánh theo đường tô màu vàng như trong *Hình 5*. *Hình 5* cũng minh họa việc không tìm thấy giá trị khoá 50 trên một cây tìm kiếm nhị phân sau bốn bước so sánh theo đường tô màu vàng và gặp cây rỗng.



Hình 5. Minh họa việc tìm kiếm nút có giá trị khoá 55 và việc không tìm thấy nút có giá trị khoá 50 trên một cây tìm kiếm nhị phân

Trong trường hợp tồi nhất, cây tìm kiếm nhị phân suy biến thành một đường thẳng n nút (cây lệch trái hoặc cây lệch phải) như trong *Hình 6* và có chiều cao bằng số nút trừ 1, thuật toán tìm kiếm một giá trị khoá trong cây tìm kiếm nhị phân sẽ có độ phức tạp tương đương với thuật toán tìm kiếm tuần tự.



Hình 6. Ví dụ cây tìm kiếm nhị phân suy biến

Tuy nhiên, thông thường chiều cao của cây tìm kiếm nhị phân nhỏ hơn đáng kể so với số lượng nút của cây.

Lưu ý: Với một tập số nguyên dương, mỗi thứ tự chèn các phần tử vào cây tìm kiếm nhị phân có thể cho một cây tìm kiếm nhị phân khác nhau.

Hình 7 là mã giả mô tả thuật toán tìm kiếm một nút có khoá x trên cây tìm kiếm nhị phân là cây *nut_goc*. Nếu cây *nut_goc* khác rỗng thì nút gốc của cây này là *nut_goc*.

```
hàm tim_kiem(cây nut_goc, x):  
    nếu cây nut_goc là rỗng thì:  
        trả về giá trị rỗng (None)  
    trái lại  
        nếu x = khóa của nút nut_goc thì:  
            trả về nút nut_goc  
        nếu x < khóa của nút nut_goc thì:  
            # Tìm kiếm x sang cây con trái của nút nut_goc  
            trả về tim_kiem(cây con trái của nút nut_goc, x)  
        trái lại  
            # Tìm kiếm x sang cây con phải của nút nut_goc  
            trả về tim_kiem(cây con phải của nút nut_goc, x)
```

Hình 7. Mã giả mô tả thuật toán tìm kiếm

Thực hành: Từ cây tìm kiếm nhị phân trong *Hình 5*, em hãy mô tả từng bước tìm kiếm một nút có giá trị khoá bằng 65 và một nút có giá trị khoá bằng 70 trên cây.

4) Sắp xếp dựa trên cây tìm kiếm nhị phân

Nhắc lại, trên cây tìm kiếm nhị phân, đối với một nút X , các nút thuộc cây con bên trái X có khoá nhỏ hơn khoá của X và các nút thuộc cây con bên phải X có khoá lớn hơn khoá của X . Do thứ tự đưa ra các nút của phép duyệt cây theo thứ tự giữa, đối với một nút X , các nút thuộc cây con bên trái X sẽ được đưa ra trước, rồi đến X , rồi đến các nút thuộc cây con bên phải X được đưa ra. Vì vậy, thực hiện phép duyệt cây tìm kiếm nhị phân theo thứ tự giữa sẽ cho ra danh sách các giá trị khoá của các nút được sắp xếp theo thứ tự tăng dần của giá trị.

Từ nhận xét trên, ta có phương pháp sắp xếp một dãy số A dựa trên tư tưởng duyệt theo thứ tự giữa của cây tìm kiếm nhị phân như sau:

Hàm *sort* (dãy A) thực hiện các bước sau:

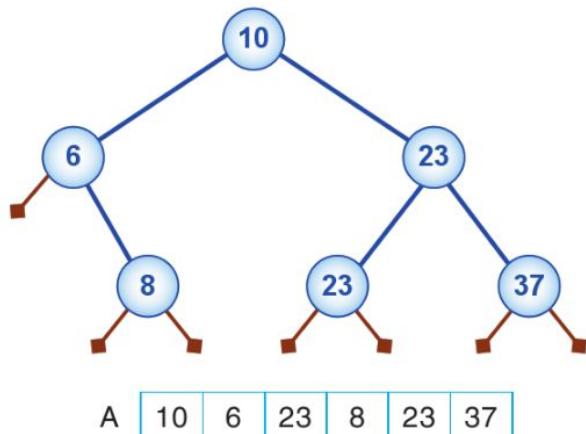
① Tạo cây T chứa các giá trị của dãy A bất kì (một giá trị có thể xuất hiện nhiều lần trong dãy A).

- Nếu giá trị $x \leq$ khoá của nút r : thêm x vào cây con trái của nút r .
- Nếu giá trị $x >$ khoá của nút r : thêm x vào cây con phải của nút r .

Lưu ý: Cây T không phải là cây tìm kiếm nhị phân như định nghĩa ở Mục 1. Cây này chỉ dùng để sắp xếp dãy A .

- 2 Xoá dãy A thành dãy rỗng.
- 3 Duyệt giữa (trái – gốc – phải) cây T và lưu các đỉnh được duyệt vào cuối dãy A .
- 4 Trả về dãy A .

Ví dụ, *Hình 8* minh họa dãy A ban đầu gồm 6 phần tử có giá trị 10, 6, 23, 8, 23, 37 và cây tìm kiếm nhị phân xây dựng từ dãy A . Danh sách các giá trị khoá được đưa ra theo phép duyệt thứ tự giữa trên cây nhị phân là: 6, 8, 10, 23, 23, 37.



Hình 8. Minh họa sắp xếp dựa trên tư tưởng duyệt cây tìm kiếm nhị phân theo thứ tự giữa



Từ một dãy có ít nhất 6 số nguyên dương tuỳ ý em hãy vẽ hình minh họa và mô tả từng bước xây dựng một cây tìm kiếm nhị phân tương ứng với dãy đó, bắt đầu từ cây rỗng.



- Câu 1.** Trong các câu sau đây, những câu nào đúng về cây tìm kiếm nhị phân?
- a) Trong cây tìm kiếm nhị phân, giá trị khoá mọi nút của cây con trái và cây con phải đều nhỏ hơn giá trị khoá của nút gốc.
 - b) Từ một dãy số nguyên dương cho trước chỉ tạo được duy nhất một cây tìm kiếm nhị phân.
 - c) Cây tìm kiếm nhị phân giúp quá trình tìm kiếm phần tử trong dãy có giá trị cho trước sẽ nhanh hơn.
 - d) Quá trình tìm kiếm trên cây tìm kiếm nhị phân luôn cho kết quả là tìm được nút có giá trị bằng giá trị khoá cho trước.

Câu 2. Trong cây tìm kiếm nhị phân em vừa xây dựng ở phần vận dụng, với một số cụ thể trong dãy em hãy cho biết cần bao nhiêu bước so sánh để tìm ra nút có giá trị khoá đó trên cây.

Tóm tắt bài học

- ✓ Cây tìm kiếm nhị phân là cây nhị phân mà tại mỗi nút thì giá trị khoá của nút này lớn hơn giá trị khoá của tất cả các nút của cây con trái và nhỏ hơn giá trị khoá của tất cả các nút của cây con phải.
- ✓ Cây tìm kiếm nhị phân biểu diễn một tập các số nguyên dương được xây dựng bằng cách chèn các phần tử của tập theo một thứ tự nhất định, xuất phát từ cây rỗng, và luôn đảm bảo tính chất của cây tìm kiếm nhị phân sau mỗi bước chèn.

Bài 4

THỰC HÀNH TỔNG HỢP: ỨNG DỤNG CÂY TÌM KIẾM NHỊ PHÂN

Học xong bài này, em sẽ:

- Ứng dụng được cây tìm kiếm nhị phân giải bài toán sắp xếp và tìm kiếm.
- Mô phỏng được thuật toán giải một số bài toán ứng dụng cây tìm kiếm nhị phân.

YÊU CẦU

Lớp chia thành các nhóm, sản phẩm của mỗi nhóm qua bài thực hành tổng hợp này bao gồm:

Thực hành 1: Mô tả các bước thực hiện thuật toán cho các yêu cầu trong bài toán.

Thực hành 2: Mô phỏng chi tiết các bước ứng dụng cây tìm kiếm nhị phân giải quyết bài toán đối với một ví dụ cụ thể.

Thực hành 3: Viết chương trình duyệt cây nhị phân theo thứ tự giữa.

Thực hành 4: Kết quả thử nghiệm trên các bộ dữ liệu đầu vào mẫu và tự tạo.

NỘI DUNG CỤ THỂ CẦN THỰC HIỆN

Gia đình Hương Trà làm kinh doanh, thường xuyên nhập về kho các mặt hàng có mã khác nhau. Mỗi lần có một mặt hàng nhập về kho, gia đình Hương Trà thường ghi chép lại bằng tay để quản lý. Sau khi được học về cây tìm kiếm nhị phân, Hương Trà muốn viết chương trình quản lý hàng hóa cho gia đình mình. Hương Trà dự định sử dụng cây tìm kiếm nhị phân để quản lý hàng hóa, mỗi nút trên cây sẽ lưu trữ thông tin của một mặt hàng. Hương Trà hiểu là một mặt hàng có nhiều thông tin cần quản lý như: giá trị, số lượng, ngày xuất/nhập, nguồn gốc,... Do mới học cây tìm kiếm nhị phân với mỗi nút chứa thông tin một khoá, Hương Trà dự kiến xây dựng cây tìm kiếm nhị phân để quản lý một thông tin là mã mặt hàng, rồi sẽ tìm kiếm thêm để mở rộng sang bài toán quản lý nhiều thông tin mặt hàng. Cụ thể, Hương Trà sẽ xây dựng cây tìm kiếm nhị phân, mỗi nút có khoá là mã của một mặt hàng sao cho có thể thuận lợi để thực hiện các yêu cầu sau đây:

- 1 Mỗi lần một mặt hàng được nhập về, Hương Trà sẽ chèn vào cây tìm kiếm nhị phân một nút có khoá là mã của mặt hàng đó.
- 2 Trả lời có tồn tại mặt hàng trong kho có mã bằng x cho trước hay không.
- 3 Dưa ra danh sách mã các món hàng trong kho theo thứ tự tăng dần.
- 4 Dưa ra danh sách các mặt hàng có mã trong khoảng từ min đến max .

Thực hành 1: Mô tả các bước thực hiện thuật toán cho bốn yêu cầu trong bài toán nêu trên

Hướng dẫn: Đầu vào của bài toán là một tập các phần tử có giá trị là mã các mặt hàng được nhập vào lần lượt. Các yêu cầu của bài toán cần được mô tả chi tiết các bước thực hiện thuật toán dựa theo hướng dẫn như sau:

- 1 Đầu tiên, khởi tạo một cây tìm kiếm nhị phân rỗng. Với mỗi mã mặt hàng được nhập vào, tạo một nút mới có khoá là mã mặt hàng vừa nhập và chèn nút vào đúng vị trí trên cây tìm kiếm nhị phân như trong Mục 2 của Bài 3.
- 2 Với một giá trị x nhập vào, thực hiện thao tác tìm kiếm giá trị khoá x trên cây tìm kiếm nhị phân như trong Mục 3 của Bài 3.
- 3 Thực hiện phép duyệt cây tìm kiếm nhị phân theo thứ tự giữa sẽ thu được danh sách các mã mặt hàng được đưa ra theo thứ tự giá trị khoá tăng dần như trong Mục 4 của Bài 3.
- 4 Thực hiện phép duyệt cây tìm kiếm nhị phân theo thứ tự giữa. Trong khi duyệt, nếu gặp nút có giá trị khoá nhỏ hơn min hoặc có giá trị khoá lớn hơn max thì dừng việc duyệt đến các nút con của nó và tiếp tục quay lại quá trình duyệt cho các nhánh khác.

Thực hành 2: Mô phỏng chi tiết các bước ứng dụng cây tìm kiếm nhị phân giải quyết bài toán đối với một ví dụ cụ thể

Hướng dẫn: Mỗi nhóm chọn một ví dụ cụ thể và mô phỏng chi tiết từng bước trên ví dụ đó, bao gồm:

- Khởi tạo cây tìm kiếm nhị phân.
- Chèn từng mã mặt hàng vào cây.
- Tìm kiếm một mã mặt hàng trên cây.
- Duyệt cây theo thứ tự giữa tìm nút có giá trị khoá x .
- Duyệt cây theo thứ tự giữa đưa ra danh sách các nút có giá trị khoá trong khoảng từ min đến max .

Thực hành 3: Viết chương trình duyệt cây nhị phân theo thứ tự giữa

Hướng dẫn: Các nhóm thực hiện theo các bước sau:

Chuẩn bị các bộ dữ liệu đầu vào cho chương trình, mỗi bộ dữ liệu cần lưu trữ trong một mảng một chiều có cấu trúc là một cây nhị phân hoàn chỉnh có tính chất cây tìm kiếm nhị phân. Ví dụ: Mảng A = [26, 21, 36, 12, None, None, 40] biểu diễn một cây tìm kiếm nhị phân hoàn chỉnh.

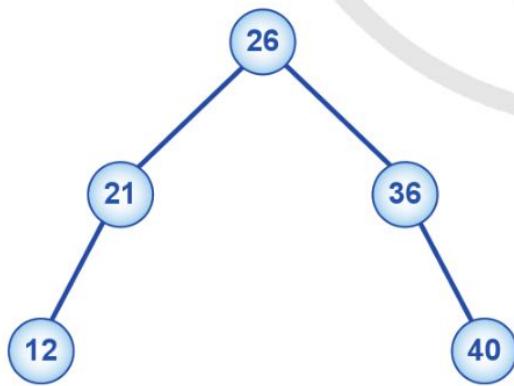
Viết chương trình duyệt cây theo thứ tự giữa sử dụng mảng một chiều và được cài đặt đệ quy.

Thực hành 4: Kết quả thử nghiệm trên các bộ dữ liệu đầu vào mẫu và tự tạo

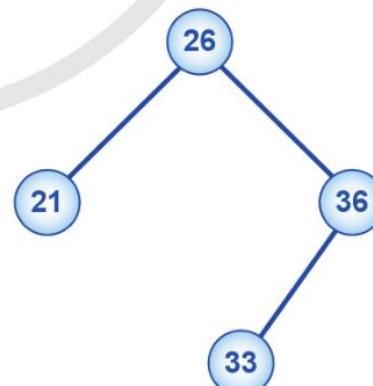
Hướng dẫn: Với hai bộ dữ liệu mẫu trong *Bảng 1* (minh họa tại *Hình 1* và *Hình 2*), kết quả thử nghiệm chương trình của mỗi nhóm cần giống với kết quả đầu ra tương ứng trong bảng:

Bảng 1. Hai bộ dữ liệu thử nghiệm

DỮ LIỆU VÀO	KẾT QUẢ RA
A = [26, 21, 36, 12, None, None, 40]	Danh sách các nút duyệt theo thứ tự giữa: 12 21 26 36 40
A = [26, 21, 36, None, None, 33]	Danh sách các nút duyệt theo thứ tự giữa: 21 26 33 36



Hình 1. Cây tìm kiếm nhị phân tương ứng với bộ dữ liệu mẫu thứ nhất



Hình 2. Cây tìm kiếm nhị phân tương ứng với bộ dữ liệu mẫu thứ hai

Yêu cầu mỗi nhóm cần tự nhập vào thêm nhiều bộ dữ liệu đầu vào khác nhau và kiểm nghiệm kết quả đầu ra.

Khi áp dụng vào thực tiễn cuộc sống, gia đình Hương Trà cần lưu trữ nhiều thông tin hơn về mỗi mặt hàng như: tên mặt hàng, số lượng, xuất xứ, giá trị nhập vào, giá trị bán ra,... Khi đó, Hương Trà cần mở rộng quản lý cây tìm kiếm nhị phân. Trong bài học này, chưa đề cập đến cách giải quyết vấn đề mở rộng này.



Em hãy thay giá trị khoá là mã từng mặt hàng bởi tên công ty sản xuất mặt hàng đó và thực hiện các yêu cầu của bài toán thực hành ở trên. Giả thiết rằng tên các công ty là đôi một khác nhau.

Lưu ý:

- Tên công ty là một chuỗi kí tự không rỗng, ví dụ: công ty AlphaBee, công ty BetaLight, công ty ZentaHome,...
- Giá trị *min* và *max* trong Thực hành 4 cũng là dạng một chuỗi kí tự.

Cây tìm kiếm nhị phân cần được xây dựng dựa trên việc so sánh chuỗi là giá trị khoá của mỗi nút. Việc so sánh hai chuỗi cần theo quy luật thứ tự từ điển của chuỗi kí tự dựa trên thuật toán như sau:

- ① Khởi tạo biến $i = 0$. Các kí tự trong chuỗi được đánh số từ 0 đến độ dài của chuỗi trừ 1.
- ② Bước lặp:
 - a) So sánh kí tự thứ i của hai chuỗi với nhau.
 - b) Kí tự nào lớn hơn thì chuỗi chứa kí tự đó sẽ lớn hơn chuỗi còn lại theo thứ tự từ điển. Kết thúc quá trình so sánh.
 - c) $i = i + 1$.
 - d) Bước lặp kết thúc khi i bằng độ dài của một trong hai chuỗi. Nếu độ dài cả hai chuỗi đều bằng i thì hai chuỗi bằng nhau theo thứ tự từ điển. Trái lại, chuỗi nào có độ dài lớn hơn sẽ lớn hơn chuỗi còn lại theo thứ tự từ điển.

Ví dụ, “Alphabet” > “Alphabee”, “BetaLight” = “BetaLight”, “ZentaHome” < “ZentaHomeStay” theo thứ tự từ điển.

Sau khi đưa ra kết quả bài Thực hành 3, em hãy nhận xét dãy tên các công ty đưa ra theo phép duyệt thứ tự giữa trên cây tìm kiếm nhị phân có đặc điểm gì?

TÌM HIỂU KĨ THUẬT DUYỆT ĐỒ THỊ VÀ ỨNG DỤNG

Bài 1

ĐỒ THỊ, PHÂN LOẠI ĐỒ THỊ

Học xong bài này, em sẽ:

- Trình bày được khái niệm đồ thị.
- Nhận biết được đơn đồ thị vô hướng và có hướng.
- Biết được một số thuật ngữ và tính chất của đồ thị.
- Tìm hiểu được một số ứng dụng.



Để mô tả các cặp tỉnh có địa giới giáp ranh nhau, ta có thể nêu bằng lời, ví dụ, quan sát lược đồ bốn tỉnh Bạc Liêu, Cà Mau, Sóc Trăng và Kiên Giang trong *Hình 1*, ta nói “Các cặp tỉnh: (Bạc Liêu, Cà Mau), (Bạc Liêu, Sóc Trăng), (Bạc Liêu, Kiên Giang), (Kiên Giang, Cà Mau) giáp ranh nhau; còn (Kiên Giang, Sóc Trăng) và (Sóc Trăng, Cà Mau) không giáp ranh nhau”. Em có cách mô tả nào khác về mối quan hệ giáp ranh giữa các tỉnh đó không?



Hình 1. Lược đồ phần đất liền một số tỉnh miền Tây Nam Bộ

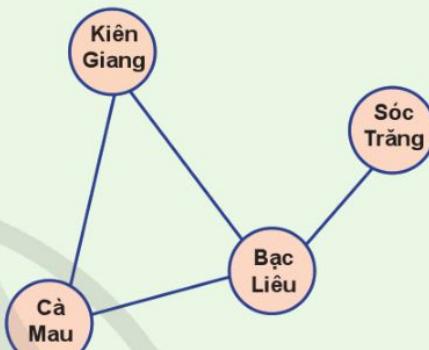
1 Khái niệm đồ thị

Để mô tả (biểu diễn) các đối tượng và mối quan hệ giữa các đối tượng có thể dùng mô hình đồ thị. Đồ thị G là một cấu trúc gồm hai tập: tập đỉnh V (mỗi đỉnh tương ứng với một đối tượng) và tập cạnh E (mỗi cạnh cho biết mối quan hệ giữa hai đối tượng).



1

Việc mô tả mối quan hệ giáp ranh của bốn tỉnh Kiên Giang, Cà Mau, Bạc Liêu, Sóc Trăng trong phần khởi động có thể biểu diễn bằng bốn đỉnh (tương ứng với bốn vòng tròn nét liền) và bốn cạnh (tương ứng với bốn đường nối giữa hai vòng tròn cho biết cặp tỉnh giáp ranh) như trong *Hình 2*. Em hãy vẽ *Hình 2* vào vở và vẽ thêm hai vòng tròn mô tả hai tỉnh Cần Thơ, Hậu Giang cùng với các cạnh để mô tả mối quan hệ giáp ranh của sáu tỉnh.



Hình 2. Đồ thị mô tả mối quan hệ giáp ranh của bốn tỉnh Kiên Giang, Cà Mau, Bạc Liêu, Sóc Trăng

Đồ thị được dùng để biểu diễn, giải nhiều bài toán trong nhiều lĩnh vực, như mô tả các hệ thống: mạng điện, mạng lưới giao thông, mạng máy tính, mối quan hệ dinh dưỡng giữa các loài,...

2 Phân loại đồ thị

Một cạnh của đồ thị có thể có các tính chất: có hướng hay vô hướng; có trọng số hay không có trọng số,... Người ta dựa vào các tính chất đó để phân loại đồ thị. Trong bài học này, chúng ta sẽ tìm hiểu hai loại đồ thị: đơn đồ thị vô hướng và đơn đồ thị có hướng.

a) Đơn đồ thị vô hướng

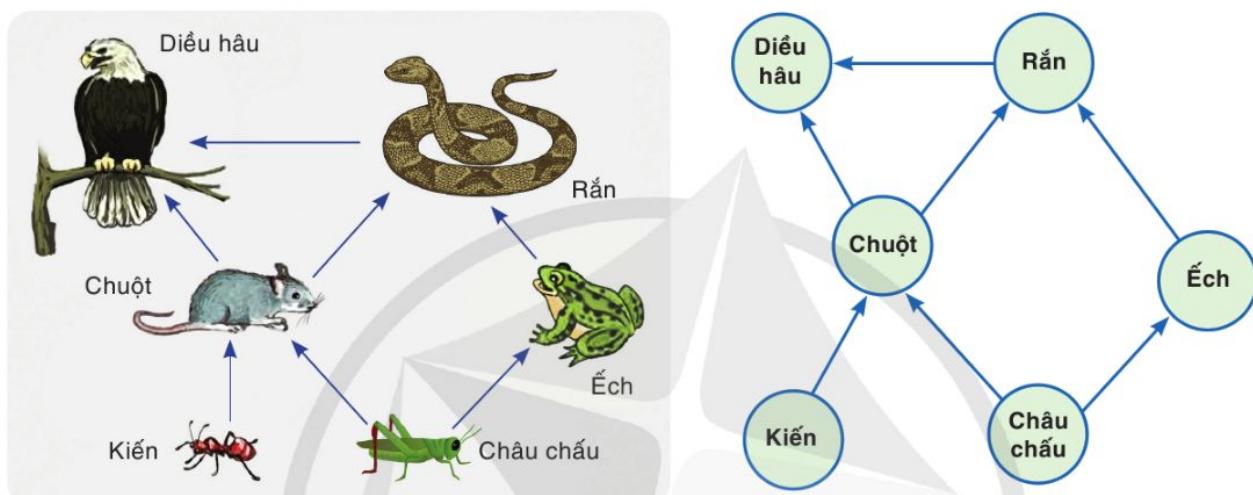
Đơn đồ thị vô hướng là đồ thị $G = (V, E)$ gồm một tập không rỗng V mà các phần tử là các đỉnh và một tập E mà các phần tử là các cạnh, mỗi cạnh tương ứng với một cặp đỉnh phân biệt *không có thứ tự*. Một cạnh nối giữa hai đỉnh phân biệt u, v kí hiệu là $\{u, v\}$ hoặc $\{v, u\}$.

Đồ thị trong *Hình 2* là đơn đồ thị vô hướng, ví dụ nói tỉnh Cà Mau giáp ranh với tỉnh Kiên Giang là đồng nghĩa với nói tỉnh Kiên Giang là tỉnh giáp ranh với tỉnh Cà Mau. Cặp đỉnh này không phân biệt thứ tự.

b) Đơn đồ thị có hướng

Đơn đồ thị có hướng là đồ thị $G = (V, E)$ gồm một tập đỉnh không rỗng V và một tập cạnh E , mỗi cạnh tương ứng với một cặp đỉnh phân biệt *có thứ tự*. Một cạnh nối từ đỉnh u tới đỉnh v kí hiệu là (u, v) .

Ví dụ: Hình 3 mô tả mối quan hệ dinh dưỡng giữa sáu loài, mỗi quan hệ giữa các đỉnh là có thứ tự, chẳng hạn Chuột là thức ăn của Rắn nhưng Rắn không phải là thức ăn của Chuột mà Rắn là thức ăn của Diều hâu.



Hình 3. Mô tả mối quan hệ dinh dưỡng giữa các loài

3) Một số thuật ngữ và tính chất trên đồ thị

a) Độ của đỉnh trong đơn đồ thị vô hướng

Trong đơn đồ thị vô hướng $G = (V, E)$ có cạnh $\{u, v\}$. Khi đó: cạnh $\{u, v\}$ được gọi là cạnh liên thuộc với hai đỉnh u, v ; đỉnh u kề với đỉnh v ; đỉnh v kề với đỉnh u ; u, v là hai đầu mút.

Độ của đỉnh u trong đơn đồ thị vô hướng là số cạnh liên thuộc với đỉnh u .

Nhận xét: Trong đơn đồ thị vô hướng thì tổng số độ của tất cả các đỉnh bằng hai lần số cạnh.

Ví dụ: Trong Hình 2, hai đỉnh tương ứng với tỉnh Kiên Giang, Cà Mau có độ bằng 2, đỉnh tương ứng với tỉnh Bạc Liêu có độ là 3.

b) Độ của đỉnh trong đơn đồ thị có hướng

Trong đơn đồ thị có hướng $G = (V, E)$ có cạnh (u, v) . Khi đó: đỉnh u được gọi là nối tới đỉnh v ; đỉnh v được nối từ đỉnh u ; đỉnh v kề với đỉnh u ; u được gọi là đỉnh đầu, v được gọi là đỉnh cuối.

Độ ra của đỉnh u trong đơn đồ thị có hướng là số cạnh có đỉnh đầu là u . Độ vào của đỉnh v trong đơn đồ thị có hướng là số cạnh có đỉnh cuối là v .

Nhận xét: Trong đơn đồ thị có hướng thì tổng số bậc ra của tất cả các đỉnh bằng tổng số bậc vào của tất cả các đỉnh.

Ví dụ: Trong *Hình 3*, đỉnh tương ứng với Rắn có bậc ra là 1 và bậc vào là 2.



Câu 1. Em hãy nêu ví dụ thực tế có thể sử dụng mô hình đồ thị để mô tả.

Câu 2. Xét đơn đồ thị có hướng gồm 6 đỉnh, các đỉnh được đánh chỉ số tương ứng từ 0 đến 5.

- Em hãy xây dựng đồ thị, biết rằng với hai đỉnh được đánh chỉ số u, v ($0 \leq u, v \leq 5$), nếu $u > v$ và $u - v$ là một số nguyên tố thì đỉnh có chỉ số u có cạnh tới được đỉnh có chỉ số v .
- Xác định bậc ra, bậc vào của từng đỉnh.



Em hãy chọn ra 6 quận/huyện trong tỉnh/thành phố nơi em ở và xây dựng đồ thị vô hướng thể hiện mối quan hệ giáp ranh của các quận/huyện đó.



Trong các câu sau đây, những câu nào đúng khi nói về đồ thị?

- Trong đồ thị, mỗi cạnh thể hiện mối quan hệ giữa hai đối tượng (hai đỉnh).
- Bậc của một đỉnh trong đơn đồ thị vô hướng có thể lẻ.
- Trong đồ thị có hướng tổng bậc ra của tất cả các đỉnh bằng tổng bậc vào của tất cả các đỉnh.
- Tuỳ theo mối quan hệ của hai đỉnh u, v bất kì mà cạnh nối hai đỉnh đó có thứ tự hay không và tương ứng là đồ thị có hướng hay đồ thị vô hướng.

Tóm tắt bài học

- ✓ Trong đồ thị $G = (V, E)$, tập đỉnh V tương ứng với tập các đối tượng và tập cạnh E cho biết các mối quan hệ giữa hai đối tượng trong tập V .
- ✓ Đồ thị $G = (V, E)$ là đơn đồ thị vô hướng nếu mỗi cạnh thể hiện mối quan hệ không có thứ tự giữa hai đỉnh, ngược lại là đơn đồ thị có hướng nếu mỗi cạnh thể hiện mối quan hệ có thứ tự giữa hai đỉnh .
- ✓ Trong đơn đồ thị vô hướng, bậc của một đỉnh là số cạnh liên thuộc với đỉnh đó.
- ✓ Trong đơn đồ thị có hướng, bậc ra của một đỉnh là số cạnh có đỉnh đầu là đỉnh đó, bậc vào của một đỉnh là số cạnh có đỉnh cuối là đỉnh đó.



Bài 2

BIỂU DIỄN ĐỒ THỊ TRÊN MÁY TÍNH

Học xong bài này, em sẽ:

- Biểu diễn được đồ thị bằng ma trận kề.
- Biểu diễn được đồ thị bằng danh sách kề.



Nam thu thập thông tin về tuyến xe buýt giữa các địa điểm và kí hiệu như trong *Bảng 1*. Ví dụ, trên hàng bắt đầu bằng kí tự A cho biết từ địa điểm A có hai tuyến xe buýt, tuyến thứ nhất từ A tới B và tuyến thứ hai từ A tới D. Với thông tin Nam thu thập được, em hãy cho biết: Từ địa điểm B có tuyến xe buýt nào tới địa điểm D không? Từ địa điểm D có những tuyến xe buýt tới các địa điểm nào?

Bảng 1. Thông tin về các tuyến xe buýt

	A	B	C	D
A		↗		↗
B	↗		↗	
C		↗		↗
D	↗	↗	↗	

1 Biểu diễn đồ thị bằng ma trận kề

Để giải quyết bài toán bằng máy tính sử dụng mô hình đồ thị, trước tiên cần phải biểu diễn được đồ thị trên máy tính. Một cách biểu diễn đơn giản, trực quan là biểu diễn bằng *ma trận kề*. Cụ thể, đối với đơn đồ thị có hướng có thể sử dụng một mảng hai chiều *g* kích thước $n \times n$ (với n là số đỉnh của đồ thị), trong đó phần tử $g[i][j]$ nhận giá trị bằng 1 (hoặc 0) tương ứng có (hoặc không) cạnh nối từ i tới j .



1

Nếu coi các địa điểm A, B, C, D trong *Bảng 1* tương ứng là các đỉnh 0, 1, 2, 3 của đồ thị thì mảng hai chiều *g* trong *Hình 1* biểu diễn đồ thị mô tả tuyến xe buýt giữa các địa điểm. Nếu Nam bổ sung thêm thông tin có một tuyến xe buýt từ B đến D, thì mảng *g* biểu diễn đồ thị thay đổi như thế nào? Em có nhận xét gì về các tuyến xe buýt mảng *g*?

	0	1	2	3
0	0	1	0	1
1	1	0	1	0
2	0	1	0	1
3	1	1	1	0

Hình 1. Mảng *g* kích thước 4×4 biểu diễn đồ thị mô tả các tuyến xe buýt

Để biểu diễn đơn đồ thị vô hướng trên máy tính, chúng ta vẫn sử dụng một mảng hai chiều g như đối với đơn đồ thị có hướng. Trong trường hợp đó, mảng g sẽ là một mảng đối xứng qua đường chéo chính, nghĩa là phần tử $g[i][j]$ có giá trị bằng $g[j][i]$.

2 Biểu diễn đồ thị bằng danh sách kề



2

Quan sát Hình 2, em hãy xây dựng mảng g biểu diễn đồ thị cho mối quan hệ giáp ranh giữa 8 tỉnh (các tỉnh được đánh số từ 0 đến 7): Sơn La (0), Điện Biên (1), Lai Châu (2), Lào Cai (3), Hà Giang (4), Cao Bằng (5), Lạng Sơn (6), Quảng Ninh (7). Mảng g có kích thước như thế nào? Em có nhận xét gì về số lượng số 0 và số 1 trong mảng g ?



Hình 2. Lược đồ một số tỉnh Bắc Bộ

Trong trường hợp đồ thị có nhiều đỉnh nhưng ít cạnh (đồ thị thưa) thì việc biểu diễn đồ thị bằng ma trận kề sẽ rất lãng phí bộ nhớ vì phần lớn các phần tử của mảng đều nhận giá trị bằng 0. Khi đó, chúng ta nên sử dụng biểu diễn đồ thị bằng danh sách kề, cụ thể, với mỗi đỉnh ta có một danh sách các đỉnh kề với đỉnh đó.

Ví dụ: Trong Bảng 2, với mỗi tỉnh ta liệt kê danh sách các tỉnh kề với tỉnh đó.

Bảng 2. Danh sách các tỉnh kề trong Hoạt động 2

TỈNH	DANH SÁCH CÁC TỈNH KỀ
Sơn La (0)	Điện Biên (1), Lai Châu (2)
Điện Biên (1)	Sơn La (0), Lai Châu (2)
Lai Châu (2)	Sơn La (0), Điện Biên (1), Lào Cai (3)
Lào Cai (3)	Lai Châu (2), Hà Giang (4)
Hà Giang (4)	Lào Cai (3), Cao Bằng (5)
Cao Bằng (5)	Hà Giang (4), Lạng Sơn (6)
Lạng Sơn (6)	Cao Bằng (5), Quảng Ninh (7)
Quảng Ninh (7)	Lạng Sơn (6)

Khi đó, ta sử dụng mảng g kích thước n (với n là số đỉnh của đồ thị), trong đó $g[i]$ là một mảng một chiều chứa các đỉnh kề của đỉnh i . Mảng g tương ứng với *Bảng 2* có giá trị như *Bảng 3*.

Cách biểu diễn này dùng được cho cả đơn đồ thị có hướng và đơn đồ thị vô hướng. Trong trường hợp đơn đồ thị vô hướng, nếu j nằm trong danh sách $g[i]$ thì i cũng sẽ nằm trong danh sách $g[j]$.



Một đồ thị gồm 4 đỉnh, các đỉnh được đánh số từ 0 đến 3, được biểu diễn bằng ma trận kề như *Hình 3*. Theo ma trận kề cho thấy từ đỉnh 1 đến được đỉnh 0 và đỉnh 2.

- Em hãy cho biết những đỉnh nào đến được đỉnh 2.
- Em hãy biểu diễn đồ thị bằng danh sách kề.



Đồ thị khối Q_3 (*Hình 4*) là đồ thị có 8 đỉnh, mỗi đỉnh là một dãy bit độ dài 3, hai đỉnh có cạnh nối nếu hai dãy bit sai khác nhau đúng một bit. Em hãy biểu diễn đồ thị bằng ma trận kề và danh sách kề.



Trong các câu sau đây, những câu nào đúng?

- Chỉ có đơn đồ thị có hướng mới biểu diễn được bằng ma trận kề.
- Cách biểu diễn đồ thị bằng ma trận kề đơn giản nhưng lãng phí bộ nhớ trong trường hợp đồ thị có nhiều đỉnh nhưng có ít cạnh.
- Chỉ có đơn đồ thị vô hướng mới biểu diễn được bằng danh sách kề.
- Cách biểu diễn bằng danh sách kề sẽ phù hợp khi đồ thị có nhiều đỉnh nhưng có ít cạnh.

Tóm tắt bài học

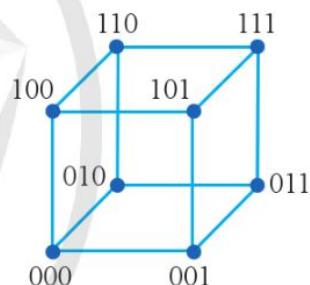
- ✓ Biểu diễn đồ thị trên máy tính bằng ma trận kề là cách sử dụng một mảng hai chiều, mỗi phần tử là 0 hoặc 1, trong đó phần tử ở hàng i cột j bằng 1 nếu có cạnh nối từ đỉnh i tới đỉnh j .
- ✓ Khi biểu diễn đồ thị có nhiều đỉnh nhưng ít cạnh trên máy tính, chúng ta nên sử dụng danh sách kề, với mỗi đỉnh lưu trữ một danh sách các đỉnh kề với đỉnh đó.

Bảng 3. Giá trị của mảng g

PHẦN TỬ	GIÁ TRỊ
$g[0]$	[1, 2]
$g[1]$	[0, 2]
$g[2]$	[0, 1, 3]
$g[3]$	[2, 4]
$g[4]$	[3, 5]
$g[5]$	[4, 6]
$g[6]$	[5, 7]
$g[7]$	[6]

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Hình 3. Một ma trận kề kích thước 4×4



Hình 4. Đồ thị khối Q_3

Bài 3

THỰC HÀNH CÁC THAO TÁC CƠ BẢN VỚI ĐỒ THỊ TRÊN MÁY TÍNH

Học xong bài này, em sẽ:

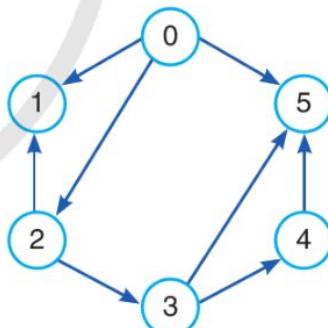
- Thực hiện được các thao tác cơ bản với đồ thị được biểu diễn bằng ma trận kề và danh sách kề. Biết được ưu, nhược điểm của hai cách biểu diễn.
- Đọc hiểu được một số đoạn chương trình sử dụng cách biểu diễn ma trận kề, danh sách kề.
- Biết được ưu, nhược điểm của hai cách biểu diễn.

Bài toán Giao hữu bóng đá

Có 6 đội bóng (các đội được đánh số từ 0 đến 5) tham gia đá giao hữu, theo thể thức mỗi cặp đội thi đấu với nhau không quá một trận và với mỗi trận sẽ luôn phân định thắng thua, không có hòa. *Hình 1* là một đồ thị biểu diễn thông tin các trận đấu đã diễn ra tính đến thời điểm hiện tại, trong đó mỗi đội tương ứng với một đỉnh của đồ thị, nếu có một cạnh nối từ đỉnh i tới đỉnh j thì cho biết hai đội i, j đã đấu với nhau và đội i giành chiến thắng ($0 \leq i, j \leq 5$).

a) Em hãy cho biết đến thời điểm hiện tại có bao nhiêu trận đấu đã diễn ra; liệt kê tất cả các trận, mỗi trận được mô tả bằng cặp số i, j có ý nghĩa là đội i đấu với đội j và đội i giành chiến thắng.

b) Xây dựng ma trận kề biểu diễn đồ thị.
c) Xây dựng danh sách kề biểu diễn đồ thị.
d) Em hãy cho biết đội số 2 đã thi đấu bao nhiêu trận, trong đó có bao nhiêu trận thắng, bao nhiêu trận thua.



Hình 1. Đồ thị biểu diễn thông tin các trận đấu đã diễn ra ở thời điểm hiện tại

Hướng dẫn:

- Xét lần lượt từng đội, với mỗi đội i ($0 \leq i \leq 5$) liệt kê các trận thắng của đội đó.
- Coi mỗi đội là một đỉnh của đồ thị, mỗi trận đấu là một cạnh của đồ thị. Xây dựng ma trận gồm 6 hàng và 6 cột (các hàng đánh số từ 0 đến 5 và từ trên xuống dưới; các cột đánh số từ 0 đến 5 và từ trái sang phải). Điền lần lượt cho từng hàng; Nếu đội i thắng đội j thì trên hàng i , cột j ghi số 1; ngược lại ghi số 0.

- c) Dựa vào ma trận đã xây dựng ở yêu cầu b) để liệt kê các đỉnh kề của mỗi đỉnh.
d) Số trận thắng của đội i ($0 \leq i \leq 5$) là số các số 1 trên hàng i của ma trận đã xây dựng ở yêu cầu b), còn số trận thua của đội i ($0 \leq i \leq 5$) là số các số 1 trên cột i của ma trận.



Tìm hiểu chương trình giúp giải quyết các yêu cầu b), c), d) của bài toán *Giao hữu bóng đá*.

a) Tìm hiểu chương trình ở *Hình 2*, dữ liệu được nhập vào như sau:

- Nhập hai số nguyên dương n, m (trong đó n là số đội bóng tham gia, m là số trận đấu đã diễn ra).
- Nhập m cặp số i_k, j_k ($0 \leq i_k, j_k < n; i_k \neq j_k$ với $k = 1, 2, \dots, m$), trong đó mỗi cặp số tương ứng với một cạnh của đồ thị nối từ đỉnh i_k đến đỉnh j_k . Cho biết đội i_k đã đấu với đội j_k và đội i_k giành chiến thắng.

```

File Edit Format Run Options Window Help
1 n, m = map(int, input().split())
2 g = [[0 for j in range(n)] for i in range(n)]
3 for k in range(m):
4     i, j = map(int, input().split())
5     g[i][j] = 1

```

Hình 2. Đoạn chương trình đọc dữ liệu và lưu trữ đồ thị bằng ma trận kề

b) Em hãy cho biết $\textcircled{?}$ trong *Hình 3* được thay bằng đại lượng thích hợp nào, sau đó ghép với đoạn chương trình ở *Hình 2* để hiển thị ma trận kề của đồ thị.

c) Em hãy cho biết $\textcircled{?}$ trong *Hình 4* được thay bằng đại lượng thích hợp nào, sau đó ghép với đoạn chương trình ở *Hình 2* để hiển thị bậc ra của các đỉnh $0, 1, \dots, n - 1$.

d) Thử nghiệm với dữ liệu đã tạo ở yêu cầu a) trong bài toán *Giao hữu bóng đá*.

```

File Edit Format Run Options Window Help
1 for i in range(n):
2     for j in range(n):
3         print(g[i][\textcircled{?}], end = " ")
4     print()

```

Hình 3. Đoạn chương trình hiển thị ma trận kề

```

File Edit Format Run Options Window Help
1 for i in range(n):
2     cnt = 0
3     for j in range(n):
4         if g[i][j] == 1:
5             cnt = cnt + \textcircled{?}
6     print(i, ":", cnt)

```

Hình 4. Đoạn chương trình tính bậc ra của các đỉnh

Bài 4

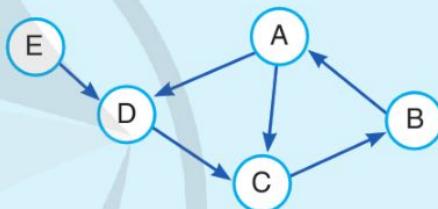
DUYỆT ĐỒ THỊ

Học xong bài này, em sẽ:

- Hiểu được cách duyệt đồ thị theo chiều sâu và chiều rộng.
- Mô phỏng được thuật toán duyệt theo chiều rộng và theo chiều sâu một đồ thị cụ thể.



Có 5 bạn A, B, C, D và E , biết rằng A có số điện thoại của C và D , do đó A có thể liên lạc với C, D ; tương tự B có số điện thoại của A ; C có số điện thoại của B ; D có số điện thoại của C ; E có số điện thoại của D . Nếu biểu diễn A, B, C, D, E là các đỉnh của đồ thị và xét mối quan hệ có số điện thoại (có thể liên lạc), ta có đồ thị như *Hình 1*. Em hãy cho biết nếu A cần thông báo một thông tin thì những ai có thể nhận được thông tin đó. Câu hỏi tương tự nếu người cần thông báo thông tin là E .



Hình 1. Một đồ thị mô tả mối quan hệ có thể liên lạc

1 Duyệt đồ thị

Duyệt đồ thị là đi thăm các đỉnh của đồ thị theo quy tắc ưu tiên định trước. Có hai cách duyệt chính là duyệt theo chiều rộng và duyệt theo chiều sâu. Trong quá trình duyệt, mỗi đỉnh được đánh dấu để đảm bảo không có đỉnh nào được thăm quá một lần. Bài toán được đề cập ở phần khởi động có thể giải quyết bằng cách duyệt đồ thị xuất phát từ đỉnh tương ứng là người cần thông báo tin, đi thăm các đỉnh của đồ thị có thể tới được.

Duyệt đồ thị có nhiều ứng dụng như kiểm tra tính kết nối của đồ thị, tìm kiếm đường đi giữa các đỉnh, hoặc xác định các thuộc tính khác nhau của đồ thị.

2 Duyệt đồ thị theo chiều rộng

Duyệt đồ thị theo chiều rộng (Breadth First Search – viết tắt là BFS) bắt đầu từ một đỉnh là thực hiện thăm các đỉnh theo từng tầng bắt đầu từ đỉnh xuất phát đó. Các đỉnh đến được từ đỉnh xuất phát là các đỉnh thuộc tầng một, các đỉnh chưa đến được

nhưng đến được từ các đỉnh thuộc tầng một là các đỉnh thuộc tầng hai,... Để duyệt đồ thị theo cách như vậy thì thứ tự đỉnh được lựa chọn để mở rộng đi tiếp là đỉnh được thăm sớm nhất trong các đỉnh chờ để mở rộng. Các bước duyệt đồ thị theo chiều rộng bắt đầu từ đỉnh $start$ sử dụng hàng đợi như sau:

Bước 1. Khởi tạo hàng đợi rỗng. Thêm đỉnh $start$ vào hàng đợi. Thăm và đánh dấu đỉnh $start$ đã thăm.

Bước 2. Gọi u là đỉnh ở đầu hàng đợi. Xét lần lượt từng đỉnh v thuộc tập đỉnh kề của u : Nếu đỉnh v chưa được thăm thì thêm đỉnh v vào hàng đợi, thăm v và đánh dấu đỉnh v đã thăm. Nếu đã xét hết các đỉnh kề của u thì xoá (lấy ra) u khỏi hàng đợi.

Bước 3. Nếu hàng đợi khác rỗng thì quay lại **Bước 2**. Ngược lại, kết thúc quá trình duyệt.

Ví dụ: Mô phỏng thuật toán BFS trên đồ thị trong *Hình 1* bắt đầu từ đỉnh A .

- Khởi tạo hàng đợi Q rỗng, thêm đỉnh A vào hàng đợi Q , thăm A và đánh dấu đỉnh A đã thăm.



- Hiện tại, đỉnh A là đỉnh ở đầu hàng đợi Q , xét đỉnh C là đỉnh kề của đỉnh A , vì đỉnh C chưa được thăm nên thêm đỉnh C vào hàng đợi Q , thăm C và đánh dấu đỉnh C đã thăm.



- Tiếp tục xét đỉnh D là đỉnh kề của đỉnh A , vì đỉnh D chưa được thăm nên thêm đỉnh D vào hàng đợi Q , thăm D và đánh dấu đỉnh D đã thăm.



- Đỉnh A đã xét hết các đỉnh kề, xoá đỉnh A khỏi hàng đợi Q .



- Hiện tại, đỉnh C đứng ở đầu hàng đợi Q , xét đỉnh B là đỉnh kề của đỉnh C vì đỉnh B chưa được thăm nên thêm đỉnh B vào hàng đợi Q , thăm B và đánh dấu đỉnh B đã thăm.



- Đỉnh C đã xét hết các đỉnh kề, xoá đỉnh C khỏi hàng đợi Q .



- 7 Hiện tại, đỉnh D đứng ở đầu hàng đợi Q , chỉ có đỉnh C là đỉnh kề của đỉnh D , nhưng đỉnh C đã thăm nên không được thêm vào hàng đợi Q . Khi đó, đỉnh D đã xét hết các đỉnh kề, xoá đỉnh D khỏi hàng đợi Q .



- 8 Hiện tại, đỉnh B đứng ở đầu hàng đợi Q , chỉ có đỉnh A là đỉnh kề của đỉnh B , nhưng đỉnh A đã thăm nên không được thêm vào hàng đợi Q . Khi đó, đỉnh B đã xét hết các đỉnh kề, xoá đỉnh B khỏi hàng đợi Q , hàng đợi Q rỗng, thuật toán kết thúc.



Sau khi thực hiện duyệt đồ thị theo chiều rộng bắt đầu từ đỉnh A , ta thấy từ đỉnh A có thể đi đến được các đỉnh C, D, B . Như vậy, câu trả lời cho phần khởi động là: Nếu A muốn thông tin cho các bạn thì các bạn B, C, D sẽ nhận được thông tin.

Khi duyệt đồ thị theo chiều rộng bắt đầu từ một đỉnh, chúng ta có thể chưa thăm được hết toàn bộ các đỉnh của đồ thị. Chẳng hạn, trong ví dụ trên đỉnh E chưa được thăm. Để duyệt toàn bộ đồ thị theo chiều rộng, ta cần xét lần lượt từng đỉnh s của đồ thị, nếu đỉnh s chưa được thăm thì gọi hàm BFS bắt đầu từ đỉnh s . Trong *Hình 2* là mã giả mô tả thuật toán duyệt đồ thị theo chiều rộng bắt đầu từ đỉnh start và duyệt toàn bộ đồ thị.

```
# Duyệt đồ thị theo chiều rộng bắt đầu từ đỉnh start
hàm BFS(đỉnh start):
```

```
Khởi tạo hàng đợi queue rỗng
Thêm đỉnh start vào hàng đợi queue
Thăm start và đánh dấu đỉnh start đã thăm
```

lặp:

```
    u là đỉnh ở đầu hàng đợi queue
    Xét mỗi đỉnh v thuộc tập đỉnh kề của u:
        nếu đỉnh v chưa được thăm thì:
            Thêm đỉnh v vào cuối hàng đợi queue
            Thăm v và đánh dấu đỉnh v đã thăm
        Xoá đỉnh u khỏi hàng đợi queue
    cho đến khi hàng đợi queue rỗng
```

```
# Duyệt toàn bộ đồ thị theo chiều rộng
hàm bfsTraversal():
```

```
Đánh dấu tất cả các đỉnh chưa được thăm
Xét lần lượt từng đỉnh s của đồ thị:
    nếu đỉnh s chưa được thăm thì:
        BFS(đỉnh s)
```

Hình 2. Mô tả thuật toán duyệt đồ thị theo chiều rộng

Thực hành 1: Mô phỏng thuật toán BFS trên đồ thị trong *Hình 1* bắt đầu từ đỉnh E , em hãy cho biết những đỉnh nào có thể đến được từ đỉnh E .

Gợi ý: Thực hiện các bước của thuật toán duyệt đồ thị theo chiều rộng bắt đầu từ đỉnh E , sau khi duyệt xong hàng đợi Q sẽ như sau:



3) Duyệt đồ thị theo chiều sâu

Duyệt đồ thị theo chiều sâu (Depth First Search – viết tắt là DFS) bắt đầu từ một đỉnh là thực hiện thăm tất cả các đỉnh trong nhánh của đỉnh xuất phát (nhánh của một đỉnh là tập hợp tất cả các đỉnh có thể đến được từ đỉnh đó). Việc thăm một nhánh được thực hiện bằng cách đi sâu theo cạnh của đồ thị vào nhánh con chưa được thăm để thăm tất cả các đỉnh trong nhánh đó, rồi quay lại và đi nhánh con tiếp theo. Các bước duyệt đồ thị theo chiều sâu bắt đầu từ đỉnh start sử dụng ngăn xếp như sau (*Hình 3* là mã giả mô tả thuật toán):

Bước 1. Khởi tạo ngăn xếp rỗng. Thêm đỉnh $start$ vào ngăn xếp.

Bước 2. Gọi u là đỉnh ở đầu ngăn xếp. Thăm u và đánh dấu đỉnh u đã thăm. Xoá đỉnh u khỏi ngăn xếp. Xét lần lượt từng đỉnh v thuộc tập đỉnh kề của u : Nếu đỉnh v chưa được thăm thì thêm đỉnh v vào đầu ngăn xếp.

Bước 3. Nếu ngăn xếp khác rỗng thì quay lại *Bước 2*. Ngược lại, kết thúc quá trình duyệt.

```
# Duyệt đồ thị theo chiều sâu bắt đầu từ đỉnh start
hàm DFS(đỉnh start):
```

```
    Khởi tạo ngăn xếp stack rỗng
```

```
    Thêm đỉnh start vào ngăn xếp stack
```

```
    lặp:
```

```
        u là đỉnh ở đầu ngăn xếp stack
```

```
        Thăm u và đánh dấu đỉnh u đã thăm
```

```
        Xoá đỉnh u khỏi ngăn xếp stack
```

```
        Xét mỗi đỉnh v thuộc tập đỉnh kề của u:
```

```
            nếu đỉnh v chưa được thăm thì:
```

```
                Thêm đỉnh v vào đầu ngăn xếp stack
```

```
    cho đến khi ngăn xếp stack rỗng
```

Hình 3. Mã giả mô tả thuật toán duyệt đồ thị theo chiều sâu từ một đỉnh sử dụng ngăn xếp

Chúng ta cũng có thể duyệt đồ thị theo chiều sâu sử dụng kĩ thuật đệ quy (*Hình 4*). Cách triển khai này đơn giản và dễ hiểu, không cần quản lí cấu trúc dữ liệu bổ sung như ngăn xếp.

Duyệt đồ thị theo chiều sâu bắt đầu từ đỉnh start bằng kĩ thuật đệ quy hàm DFS (đỉnh u) :

Thăm u và đánh dấu đỉnh u đã thăm

Xét mỗi đỉnh v thuộc tập đỉnh kề của u:

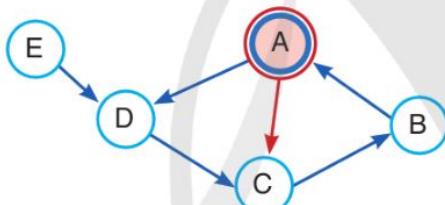
nếu đỉnh v chưa được thăm thì:

DFS (đỉnh v)

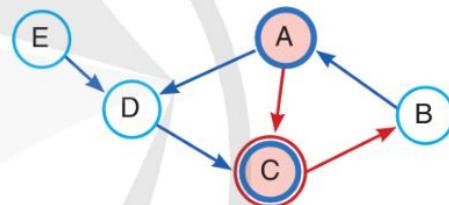
Hình 4. Mô tả thuật toán duyệt đồ thị theo chiều sâu từ một đỉnh sử dụng kĩ thuật đệ quy

Ví dụ, mô phỏng thuật toán duyệt theo chiều sâu sử dụng kĩ thuật đệ quy trên đồ thị trong *Hình 1* bắt đầu từ đỉnh A, bằng cách gọi $DFS(A)$. Từ *Hình 5* đến *Hình 11*: đường tròn đỏ bao quanh một đỉnh thể hiện đang gọi DFS tại đỉnh đó; hình tròn màu hồng chưa một đỉnh thể hiện đỉnh đó đã được thăm; cạnh tô màu đỏ thể hiện vết gọi đệ quy DFS .

- Thực hiện $DFS(A)$, thăm A và đánh dấu đỉnh A đã thăm (*Hình 5*). Có hai đỉnh C, D kề với A và đều chưa được thăm, theo thứ tự $DFS(C)$ được gọi và thực hiện trước.

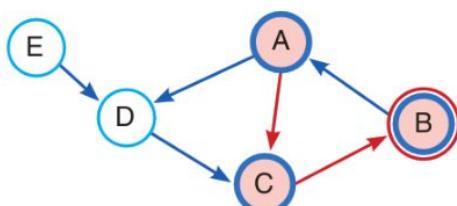


Hình 5. Trạng thái sau bước ①

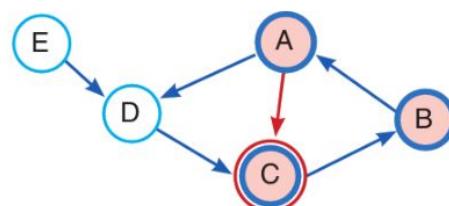


Hình 6. Trạng thái sau bước ②

- Thực hiện $DFS(C)$, thăm C và đánh dấu đỉnh C đã thăm (*Hình 6*); chỉ có B là đỉnh kề của đỉnh C và đỉnh B chưa được thăm, gọi $DFS(B)$.
- Thực hiện $DFS(B)$, thăm B và đánh dấu đỉnh B đã thăm (*Hình 7*); chỉ có A là đỉnh kề của đỉnh B, nhưng đỉnh A đã thăm nên kết thúc $DFS(B)$.

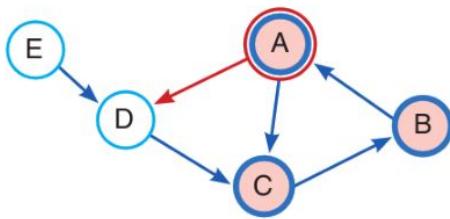


Hình 7. Trạng thái sau bước ③

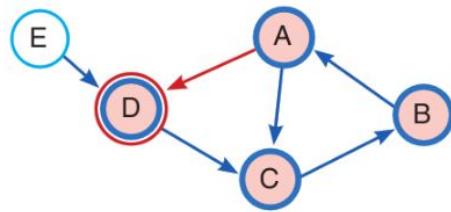


Hình 8. Trạng thái sau bước ④

- Quay về $DFS(C)$, do đã xét hết đỉnh kề của đỉnh C nên $DFS(C)$ kết thúc (*Hình 8*).
- Quay về $DFS(A)$, xét D là đỉnh kề tiếp theo của đỉnh A; vì đỉnh D chưa được thăm nên gọi $DFS(D)$ (*Hình 9*).

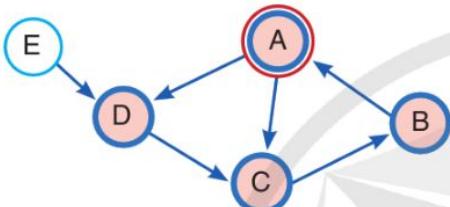


Hình 9. Trạng thái sau bước 5



Hình 10. Trạng thái sau bước 6

- 6 Thực hiện $DFS(D)$, thăm D và đánh dấu đỉnh D đã thăm (Hình 10); chỉ có C là đỉnh kề của đỉnh D nhưng đỉnh C đã thăm nên kết thúc $DFS(D)$.
- 7 Quay về $DFS(A)$, do đã xét hết các đỉnh kề của đỉnh A nên $DFS(A)$ kết thúc (Hình 11), thuật toán dừng.



Hình 11. Trạng thái sau bước 7

Sau khi thực hiện duyệt đồ thị theo chiều sâu, ta thấy từ đỉnh A có thể đi tới được các đỉnh C, B, D . Như vậy, cả hai cách duyệt đồ thị theo chiều sâu và chiều rộng đều có thể trả lời cho câu hỏi trong phần khởi động.

Tương tự như duyệt toàn bộ đồ thị theo chiều rộng, chúng ta cũng có thể duyệt toàn bộ đồ thị theo chiều sâu như trong Hình 12 bằng cách xét lần lượt từng đỉnh s của đồ thị, nếu đỉnh s chưa được thăm thì gọi hàm DFS (trong Hình 3 hoặc Hình 4) bắt đầu từ đỉnh s .

```
# Duyệt toàn bộ đồ thị theo chiều sâu
hàm dfsTraversal():
    Đánh dấu tất cả các đỉnh đều chưa được thăm
    Xét lần lượt từng đỉnh s của đồ thị:
        nếu đỉnh s chưa được thăm thì:
            DFS(đỉnh s)
```

Hình 12. Mô tả thuật toán duyệt toàn bộ đồ thị theo chiều sâu

Thực hành 2: Mô phỏng thuật toán DFS trên đồ thị trong Hình 1 bắt đầu từ đỉnh E .

Gợi ý: Thực hiện các bước của thuật toán duyệt đồ thị theo chiều sâu bắt đầu từ đỉnh E .



Với các thông tin về tuyến xe buýt giữa các địa điểm được biểu diễn bằng ma trận kè như *Hình 13*? Em áp dụng thuật toán duyệt theo chiều rộng hoặc theo chiều sâu để chỉ ra các địa điểm có thể đến được nếu xuất phát địa điểm 0 và chỉ sử dụng các tuyến xe buýt này.

	0	1	2	3
0	0	1	0	1
1	1	0	1	0
2	0	1	0	1
3	1	1	1	0

Hình 13. Ma trận kè biểu diễn các tuyến xe buýt giữa bốn địa điểm



Trong các câu sau đây, những câu nào đúng khi nói về duyệt đồ thị?

- Duyệt đồ thị theo chiều sâu giúp ta xác định các đỉnh có thể tới được từ một đỉnh bất kì.
- Duyệt đồ thị theo chiều rộng không thể giúp ta xác định các đỉnh có thể tới được từ một đỉnh bất kì.
- Thứ tự thăm các đỉnh khi thực hiện cách duyệt đồ thị theo chiều rộng và theo chiều sâu sẽ giống hệt nhau.
- Để duyệt đồ thị theo chiều rộng chúng ta sử dụng hàng đợi, thăm các đỉnh theo nguyên tắc vào trước ra trước.
- Để duyệt đồ thị theo chiều sâu chúng ta sử dụng ngăn xếp, thăm các đỉnh theo nguyên tắc vào sau ra trước.

Tóm tắt bài học

- ✓ Có thể duyệt đồ thị theo hai cách: duyệt đồ thị theo chiều rộng và duyệt đồ thị theo chiều sâu.
- ✓ Để duyệt đồ thị theo chiều rộng chúng ta sử dụng hàng đợi, thăm các đỉnh theo nguyên tắc vào trước ra trước (FIFO).
- ✓ Để duyệt đồ thị theo chiều sâu chúng ta sử dụng ngăn xếp, thăm các đỉnh theo nguyên tắc vào sau ra trước (LIFO) hoặc theo kĩ thuật đệ quy.

Bài 5

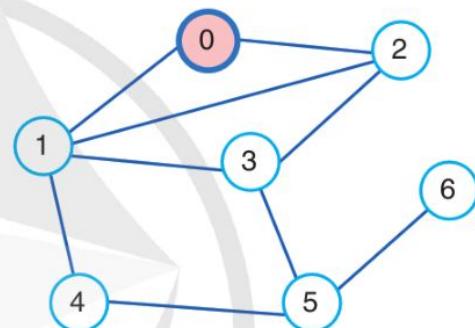
THỰC HÀNH DUYỆT ĐỒ THỊ

Học xong bài này, em sẽ:

- Tìm hiểu được một số ứng dụng thực tế của duyệt đồ thị.
- So sánh được hai cách duyệt đồ thị theo chiều rộng và chiều sâu.

Bài toán Phân nhóm người nghi nhiễm COVID-19

Một nhóm gồm 7 người được đánh số từ 0 đến 6, có một số cặp người thường xuyên tiếp xúc trao đổi với nhau được mô tả như trong *Hình 1*. Một ngày, người 0 xét nghiệm và được xác định là bị nhiễm COVID-19, người ta cần phân 7 người thành các nhóm, người 0 sẽ thuộc nhóm F0, những người tiếp xúc với người ở nhóm F0 sẽ được xếp vào nhóm F1, những người tiếp xúc với người ở nhóm F1 sẽ được xếp vào nhóm F2,...



Hình 1. Nhóm gồm 7 người và mối quan hệ thường xuyên tiếp xúc

Nhiệm vụ: Dùng thuật toán duyệt đồ thị theo chiều rộng bắt đầu từ người 0 để phân nhóm người nghi nhiễm COVID-19.

Gợi ý: Dùng mảng hàng đợi Q để thăm các đỉnh theo chiều rộng, mảng *level* để ghi nhận nhóm nghi nhiễm, *Hình 2, 3, 4* mô tả ba bước đầu tiên trong quá trình thực hiện.

Q	0					
level	0					

Hình 2. Trạng thái ở bước 1

Q	0	1	2			
level	0	1	1			

Hình 3. Trạng thái ở bước 2

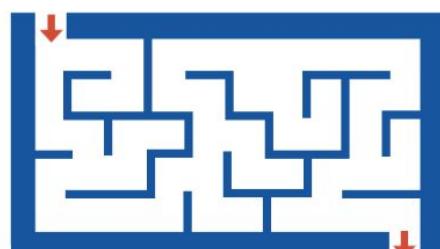
Q	0	1	2	3	4	
level	0	1	1	2	2	

Hình 4. Trạng thái ở bước 3



Bài toán Tìm đường đi trong mê cung

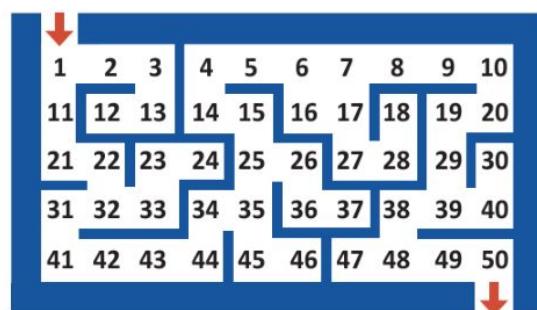
Nam đang chơi trò chơi tìm đường đi trong mê cung như trong *Hình 5*. Dùng thuật toán duyệt đồ thị theo chiều sâu để kiểm tra: Nam có thể di vào mê cung từ góc trái trên và ra khỏi mê cung ở góc phải dưới hay không?



Hình 5. Ví dụ một mê cung trong trò chơi

Gợi ý: Xây dựng đồ thị, đánh số các ô trong mê cung (*Hình 6*), mỗi ô tương ứng một đỉnh của đồ thị, hai ô kề cạnh có cạnh nối. Theo như *Hình 6*, đồ thị có 50 đỉnh.

Duyệt đồ thị theo chiều sâu bắt đầu từ ô số 1 theo thứ tự ưu tiên đi theo ô xuống dưới, sang phải, lên trên, sang trái.



Hình 6. Mê cung có các ô được đánh số

BÀI TÌM HIỂU THÊM

TÌM HIỂU CHƯƠNG TRÌNH GIẢI BÀI TOÁN PHÂN NHÓM NGƯỜI NGHI NHIỄM COVID-19

Nhập dữ liệu từ bàn phím các thông tin sau:

Hai số nguyên n, m lần lượt là số người và số cặp người thường xuyên tiếp xúc, đánh số mỗi người từ 0 đến $n - 1$.

Tiếp theo là m cặp số cho biết m cặp người thường xuyên tiếp xúc.

Kết quả ra màn hình gồm n số, số thứ i ($0 \leq i \leq n - 1$) cho biết người i thuộc nhóm nghi nhiễm nào.

DỮ LIỆU VÀO	KẾT QUẢ RA	GIẢI THÍCH
4 4 0 1 0 3 1 2 2 3	0 1 2 1	Người 0 thuộc F0; người 1 và người 3 thuộc F1; người 2 thuộc F2. <pre>graph TD; 0((0)) --- 1((1)); 1 --- 2((2)); 2 --- 0; 0 --- 3((3)); 3 --- 1;</pre>
6 7 0 1 0 3 1 2 2 3 0 2 3 4 4 5	0 1 1 1 2 3	Người 0 thuộc F0; người 1, người 2 và người 3 thuộc F1; người 4 thuộc F2; người 5 thuộc F3. <pre>graph TD; 0((0)) --- 1((1)); 1 --- 2((2)); 2 --- 0((0)); 0 --- 3((3)); 3 --- 1; 3 --- 4((4)); 4 --- 5((5));</pre>

Các bước xây dựng chương trình:

① Đọc dữ liệu vào và biểu diễn đồ thị trên máy tính dùng danh sách kề.

Đoạn chương trình đọc và lưu trữ đồ thị bằng mảng g (từ dòng lệnh 3 đến dòng lệnh 8 ở *Hình 7*).

② Duyệt đồ thị theo chiều rộng để phân nhóm người nghi nhiễm COVID-19.

Chương trình sử dụng hàng đợi trong thư viện *Queue* với hai thao tác *put* (đẩy vào hàng đợi) và *get* (lấy phần tử ở đầu hàng đợi), thực hiện thuật toán duyệt đồ thị theo chiều rộng.

Ngoài ra, chương trình sử dụng mảng *level* để lưu mức (nhóm nghi nhiễm) của từng đỉnh. Đỉnh v chưa được gán mức mà tiếp xúc với đỉnh u thì $level[v] = level[u] + 1$.

```
File Edit Format Run Options Window Help
1 from queue import Queue
2 n, m = map(int, input().split())
3 g = [[] for i in range(n)]
4 for k in range(m):
5     i, j = map(int, input().split())
6     g[i].append(j)
7     g[j].append(i)
8 level = [-1 for i in range(n)]
9 q = Queue(maxsize = n)
10 q.put(0)
11 level[0] = 0
12 while not q.empty():
13     u = q.get()
14     for v in g[u]:
15         if level[v] == -1:
16             level[v] = level[u] + 1
17             q.put(v)
18 for i in range(n):
19     print(level[i], end = " ")
```

Hình 7. Chương trình giải bài toán phân nhóm người nghi nhiễm COVID-19

Bài 6

DỰ ÁN HỌC TẬP: TÌM HIỂU CÁC VẤN ĐỀ ÚNG DỤNG ĐỒ THỊ

1 Mục đích của dự án

Dự án nhằm mục đích giúp học sinh khám phá, tìm hiểu các vấn đề liên quan đến đồ thị. Sau khi hoàn thành xong dự án, học sinh có khả năng:

- Tìm kiếm, khai thác, học hỏi mở rộng kiến thức từ các nguồn tài nguyên về lập trình trên mạng Internet.
- Vận dụng được kiến thức đã học để giải quyết các vấn đề cụ thể, thông thường.
- Tăng cường được khả năng làm việc nhóm.

2 Yêu cầu chung

- Lớp chia thành 4 nhóm độc lập. Mỗi nhóm lập trưởng nhóm và thảo luận tự chọn một vấn đề liên quan đến đồ thị.
- Thời gian thực hiện dự án là 2 tuần. Các nhóm sẽ thực hiện tạo sản phẩm ngoài giờ lên lớp và trong 2 tiết các nhóm báo cáo kết quả tại lớp, mỗi nhóm 15 phút.

3 Hướng dẫn thực hiện dự án

a) Lập kế hoạch

Bước 1. Xác định mục tiêu dự án.

- Nhóm thảo luận lựa chọn, thống nhất bài toán.
- Các nhóm có thể lựa chọn một trong các bài toán ở mục 5.

Bước 2. Lập danh sách các công việc chính, thời gian thực hiện, hoàn thành.

Bước 3. Dự kiến sản phẩm.

b) Thực hiện dự án

- Xác định các vấn đề liên quan đến dự án.
- Xây dựng một số ví dụ minh họa và mã giả giải quyết vấn đề.
- Sử dụng các lệnh trong Python để viết chương trình, thử nghiệm, kiểm thử để sửa lỗi, hiệu chỉnh chương trình (nếu có).

c) Báo cáo kết quả

Tiến trình thực hiện như đề xuất tham khảo trong bảng sau đây:

STT	NỘI DUNG, NHIỆM VỤ	NGƯỜI THỰC HIỆN	THỜI GIAN VÀ ĐỊA ĐIỂM	CẦN CHUẨN BỊ
1	Các nhóm báo cáo kết quả và tự đánh giá	Tất cả các nhóm	Phòng học bộ môn	Máy tính và máy chiếu
2	Tranh biện giữa các nhóm, đánh giá chéo	Tất cả các nhóm	Phòng học bộ môn	Máy tính và máy chiếu
3	Giáo viên tổng kết, đánh giá kết luận	Giáo viên và học sinh	Phòng học bộ môn	Máy tính và máy chiếu

4) Tiêu chí đánh giá sản phẩm

Việc đánh giá nội dung dựa trên các tiêu chí sau: Vấn đề có hấp dẫn, thú vị không? Mức độ khó của vấn đề? Ý nghĩa về khoa học? Ý nghĩa về thực tế?

Mỗi tiêu chí được đánh giá theo 4 mức A, B, C, D.

5) Gợi ý một số vấn đề

Vấn đề 1. Tìm đường đi ngắn nhất trên đơn đồ thị có hướng

Một dãy đỉnh $a = i_0, i_1, \dots, i_s = b$ ($a \neq b$) được gọi là đường đi từ đỉnh a tới đỉnh b nếu hai đỉnh liên tiếp trên đường đi có cạnh nối.

Trong cuộc sống chúng ta gặp rất nhiều bài toán liên quan đến tìm đường đi, dưới đây là một ví dụ cụ thể.

Có n sân bay được đánh số từ 0 đến $n - 1$, có m tuyến bay giữa các sân bay, tuyến thứ k sẽ bay từ sân bay i_k sang sân bay j_k . Em hãy tìm cách di chuyển từ sân bay a tới sân bay b với số tuyến bay là ít nhất.

Hướng dẫn: Bài toán này có thể giải quyết bằng cách duyệt đồ thị theo chiều rộng bắt đầu từ đỉnh A . Sử dụng mảng *len* và mảng *trace* trong quá trình duyệt đồ thị, trong đó *len[v]* là độ dài đường đi ngắn nhất từ đỉnh xuất phát đến đỉnh v và *trace[v] = u* cho biết để đến được đỉnh v là từ đỉnh u . Cụ thể thuật toán gồm hai bước:

Bước 1. BFS từ đỉnh xuất phát để tính được mảng *len* là mảng độ dài đường đi; mỗi khi tính *len[v] = len[u] + 1* thì cũng ghi nhận *trace[v] = u*.

Bước 2. Sử dụng mảng *trace* để tìm đường.

Vấn đề 2. Kiểm tra tính liên thông của đơn đồ thị vô hướng

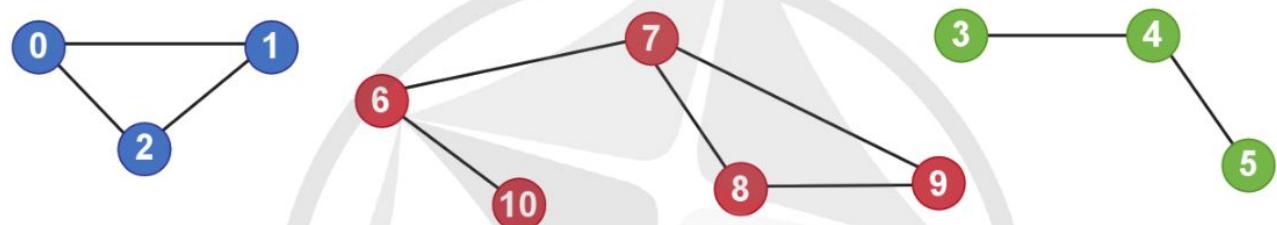
Một đơn đồ thị vô hướng được gọi là liên thông nếu hai đỉnh bất kì của đồ thị đều có đường đi tới nhau.

Bài toán kiểm tra tính liên thông của đơn đồ thị vô hướng xuất hiện nhiều trong nghiên cứu lí thuyết cũng như ứng dụng trong cuộc sống, sau đây là một ví dụ cụ thể:

Có n địa điểm được đánh số từ 0 đến $n - 1$, có m tuyến xe buýt hai chiều giữa các địa điểm, tuyến thứ k sẽ di chuyển giữa hai địa điểm i_k và j_k . Em hãy kiểm tra xem từ một địa điểm bất kỳ có thể tới được các địa điểm còn lại bằng cách chỉ sử dụng m tuyến xe buýt hay không.

Hướng dẫn: Bài toán này có thể giải quyết bằng cách duyệt đồ thị theo chiều rộng hoặc chiều sâu bắt đầu từ một đỉnh bất kỳ. Sau khi duyệt đồ thị, kiểm tra trong mảng đánh dấu các đỉnh đã thăm, nếu tồn tại một đỉnh chưa được thăm thì kết luận đồ thị không liên thông.

Ví dụ: Với hệ thống các tuyến xe buýt như *Hình 1*, từ địa điểm 8 chỉ có thể đến được các địa điểm 6, 7, 9, 10.



Hình 1. Hệ thống các tuyến xe buýt giữa các địa điểm

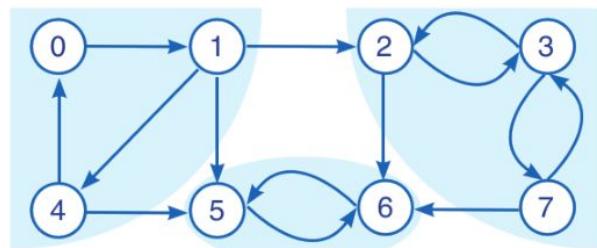
Vấn đề 3. Kiểm tra tính liên thông của đơn đồ thị có hướng

Trong đơn đồ thị có hướng, người ta thường quan tâm đến tính liên thông mạnh của đồ thị. Một đồ thị có hướng được gọi là liên thông mạnh nếu từ một đỉnh bất kỳ của đồ thị có thể đến được tất cả các đỉnh còn lại.

Các bài toán liên quan đến tính liên thông của đơn đồ thị có hướng cũng có nhiều ứng dụng như tính liên thông của đơn đồ thị vô hướng.

Hướng dẫn: Cách giải đơn giản cho bài toán này là duyệt đồ thị theo chiều rộng hoặc chiều sâu n lần (n là số đỉnh của đồ thị). Mỗi lần bắt đầu từ một đỉnh để kiểm tra từ đỉnh đó có tới được tất cả các đỉnh còn lại không. Cụ thể, sau khi duyệt đồ thị từ đỉnh u ($0 \leq u \leq n - 1$), kiểm tra trong mảng đánh dấu các đỉnh đã thăm, nếu tồn tại một đỉnh chưa được thăm thì đỉnh đó không thể đến được từ đỉnh u , khi đó có thể kết luận luôn đồ thị không liên thông mạnh.

Ví dụ: Với đơn đồ thị có hướng trong *Hình 2*, đồ thị này không có tính liên thông mạnh vì từ đỉnh 5 không có đường đi nào đến được đỉnh 4.



Hình 2. Ví dụ một đơn đồ thị có hướng không có tính liên thông mạnh

BẢNG GIẢI THÍCH THUẬT NGỮ

Thuật ngữ	Giải thích	Trang
BFS (Breadth-First Search)	duyệt đồ thị theo chiều rộng bắt đầu từ một đỉnh xuất phát, thực hiện thăm các đỉnh theo từng tầng bằng cách sử dụng hàng đợi.	62
DFS (Depth-First Search)	duyệt đồ thị theo chiều sâu bắt đầu từ một đỉnh xuất phát, thực hiện thăm các đỉnh theo từng nhánh bằng cách sử dụng ngăn xếp hoặc kĩ thuật đệ quy đi sâu vào nhánh con chưa được thăm để thăm tất cả các đỉnh trong nhánh đó, rồi quay lại và đi nhánh con tiếp theo.	65
duyệt cây nhị phân	quá trình đi đến tất cả nút của cây và mọi nút chỉ được duyệt đúng một lần. Tùy theo thứ tự duyệt nút, có ba thứ tự duyệt là: duyệt trước, duyệt giữa và duyệt cuối.	34
đơn đồ thị có hướng	đồ thị mà với hai đỉnh phân biệt u và v bất kì thì có tối đa một cạnh có hướng từ đỉnh u đến đỉnh v và ngược lại, không có cạnh nối cùng một đỉnh.	55
đơn đồ thị vô hướng	đồ thị mà với hai đỉnh phân biệt u và v bất kì thì có tối đa một cạnh không có hướng nối giữa đỉnh u và đỉnh v , không có cạnh nối cùng một đỉnh.	54
queue (hang đợi)	một cấu trúc dữ liệu tuyến tính dùng để lưu danh sách các phần tử. Hai thao tác cơ bản trên hàng đợi là: thao tác thêm vào (enqueue) ở cuối hàng đợi (back) và thao tác lấy ra (dequeue) ở đầu hàng đợi (front). Hai thao tác này thể hiện cơ chế hoạt động "Vào trước – Ra trước" (FIFO – First In, First Out).	6
stack (ngăn xếp)	một cấu trúc dữ liệu tuyến tính dùng để lưu danh sách các phần tử. Hai thao tác cơ bản trên ngăn xếp là: thao tác thêm vào (push) và thao tác lấy ra (pop) đều ở đỉnh ngăn xếp (top). Hai thao tác này thể hiện cơ chế hoạt động "Vào sau – Ra trước" (LIFO – Last In, First Out).	12
trường hợp tối nhất	trường hợp dữ liệu đầu vào của bài toán làm cho thời gian thực hiện của thuật toán là lớn/nhiều/chậm nhất hoặc vùng nhớ dùng thêm của thuật toán là lớn/nhiều nhất.	33

Chịu trách nhiệm tổ chức bản thảo và bản quyền nội dung:

CÔNG TY CỔ PHẦN ĐẦU TƯ XUẤT BẢN – THIẾT BỊ GIÁO DỤC VIỆT NAM

Chủ tịch Hội đồng Quản trị: NGUYỄN NGÔ TRẦN ÁI

Tổng Giám đốc: VŨ BÁ KHÁNH

Biên tập:

TRẦN THỊ HIÊN

Thiết kế và minh họa sách:

PHẠM VŨ TOẢN

Trình bày bìa:

TRẦN TIỀU LÂM – NGUYỄN MẠNH HÙNG

Sửa bản in:

TRẦN THỊ HIÊN

Trong sách có sử dụng một số hình ảnh trên Internet. Trân trọng cảm ơn các tác giả.

Chuyên đề học tập Tin học 12 – Khoa học máy tính

Mã số:

ISBN:

In , khổ 19 x 26,5cm, tại

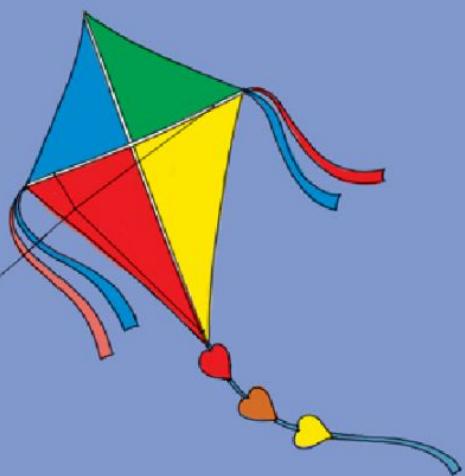
Địa chỉ:

Số xác nhận đăng ký xuất bản:

Quyết định xuất bản số:

..... in xong và nộp lưu chiểu tháng năm 20...

Mang cuộc sống vào bài học Đưa bài học vào cuộc sống



Quyển sách **Chuyên đề học tập Tin học 12 – Khoa học máy tính** sẽ hướng dẫn học sinh tìm hiểu và thực hành ứng dụng một số dạng cấu trúc dữ liệu thiết yếu trong tin học thông qua ba chuyên đề: *Tìm hiểu một vài kiểu dữ liệu tuyến tính; Tìm hiểu Cây tìm kiếm nhị phân trong sắp xếp và tìm kiếm; Tìm hiểu kỹ thuật duyệt Đồ thị và ứng dụng*. Các nội dung trong quyển sách này được thiết kế kĩ lưỡng, chú trọng vào việc thực hành và hoàn thiện sản phẩm. Điều đó giúp học sinh phát triển khả năng áp dụng phong phú các kiểu cấu trúc dữ liệu vào giải quyết vấn đề.

Sách do các nhà giáo giàu kinh nghiệm, nhiều tâm huyết trong lĩnh vực giáo dục tin học biên soạn.

- 1. Quét mã QR hoặc dùng trình duyệt web để truy cập website bộ sách Cánh Diều: www.hoc10.com
- 2. Vào mục Hướng dẫn (www.hoc10.com/huong-dan) để kiểm tra sách già và xem hướng dẫn kích hoạt sử dụng học liệu điện tử.

SỬ DỤNG
TEM CHỐNG GIẢ

SÁCH KHÔNG BÁN