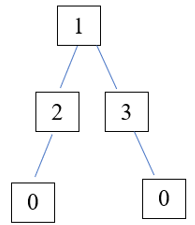
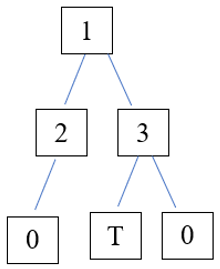
# Bài 10: Thực hành tổng hợp với cây tìm kiếm nhị phân

**Giải Chuyên đề Tin học 12 Bài 10: Thực hành tổng hợp với cây tìm kiếm nhị phân**  
**Khởi động trang 46 Chuyên đề Tin học 12**: Trong bài 9, chúng ta đã học thao tác duyệt cây. Với bài toán thực tế quản lí danh bạ điện thoại, làm thế nào để sử dụng các thao tác đó vào cây tìm kiếm nhị phân để thêm, tìm kiếm, hiển thị toàn bộ các liên hệ theo thứ tự sắp xếp của tên lên hệ trong danh bạ.  
**Lời giải:**  
Với bài toán thực tế quản lí danh bạn điện thoại, để sử dụng các thao tác đó vào cây tìm kiếm nhị phân để thêm, tìm kiếm, hiển thị toàn bộ các liên hệ theo thứ tự sắp xếp của tên lên hệ trong danh bạ ta phải viết ứng dụng quản lí danh bạ, sử dụng cấu trúc dữ liệu cây tìm kiếm nhị phân để viết ứng dụng này.  
**Luyện tập 1 trang 48 Chuyên đề Tin học 12**: Hãy vẽ cây tìm kiếm nhị phân ứng với  
a) Dữ liệu tệp contacts.inp ở trong phần thực hành.  
b) Từ cây nhận được ở ý a, thêm liên hệ “Anh, Nguyễn Văn Tùng, 0982 000 134”.  
**Lời giải:**  
a) Dữ liệu tệp contacts.inp ở trong phần thực hành.  
  
b) Từ cây nhận được ở ý a, thêm liên hệ “Anh, Nguyễn Văn Tùng, 0982 000 134”.  
  
Để vẽ cây tìm kiếm nhị phân ứng với dữ liệu từ tệp contacts.inp, chúng ta cần đọc dữ liệu từ tệp và thêm các liên hệ vào cây tương ứng. Sau đó, chúng ta có thể vẽ cây đó.  
Dưới đây là một phác thảo Python cho cách thực hiện điều này:  
class Contact:  
 def \_\_init\_\_(self, name, phone\_number):  
 self.name = name  
 self.phone\_number = phone\_number  
class TreeNode:  
 def \_\_init\_\_(self, contact):  
 self.contact = contact  
 self.left = None  
 self.right = None  
class PhoneBook:  
 def \_\_init\_\_(self):  
 self.root = None  
 def insert(self, contact):  
 self.root = self.\_insert\_recursive(self.root, contact)  
 def \_insert\_recursive(self, root, contact):  
 if root is None:  
 return TreeNode(contact)  
 if contact.name < root.contact.name:  
 root.left = self.\_insert\_recursive(root.left, contact)  
 elif contact.name > root.contact.name:  
 root.right = self.\_insert\_recursive(root.right, contact)  
 return root  
 def display\_contacts(self):  
 self.\_in\_order\_traversal(self.root)  
 def \_in\_order\_traversal(self, root):  
 if root:  
 self.\_in\_order\_traversal(root.left)  
 print("Name:", root.contact.name, "- Phone:", root.contact.phone\_number)  
 self.\_in\_order\_traversal(root.right)  
# Đọc dữ liệu từ tệp contacts.inp và thêm liên hệ vào danh bạ điện thoại  
phone\_book = PhoneBook()  
with open("contacts.inp", "r") as file:  
 for line in file:  
 parts = line.strip().split(", ")  
 name = parts[0]  
 phone\_number = parts[1]  
 phone\_book.insert(Contact(name, phone\_number))  
# Hiển thị toàn bộ danh sách liên hệ trước khi thêm liên hệ mới  
print("Contacts before adding new contact:")  
phone\_book.display\_contacts()  
# Thêm liên hệ mới  
new\_contact = Contact("Anh, Nguyễn Văn Tùng", "0982 000 134")  
phone\_book.insert(new\_contact)  
# Hiển thị toàn bộ danh sách liên hệ sau khi thêm liên hệ mới  
print("\nContacts after adding new contact:")  
phone\_book.display\_contacts()  
**\* Lưu ý thêm:**  
Sau khi chạy mã này, chúng ta sẽ có cây tìm kiếm nhị phân chứa tất cả các liên hệ từ tệp contacts.inp, và sau đó sẽ thêm một liên hệ mới vào cây. Tuy nhiên, để vẽ cây như bạn yêu cầu, chúng ta cần một số thư viện hỗ trợ vẽ đồ thị. Bạn có thể sử dụng thư viện như **matplotlib** hoặc **graphviz** để vẽ cây  
**Luyện tập 2 trang 48 Chuyên đề Tin học 12**: Tiếp tục với ứng dụng quản lí danh bạ, chức năng hiển thị danh sách liên hệ theo thứ tự từ điển. Do hạn chế của màn hình, mỗi trang chỉ hiển thị được 20 liên hệ. Hãy thêm tính năng in các liên hệ ở trang n bất kì do người dùng nhập vào, điều kiện n nguyên, lớn hơn 0 và nhỏ hơn hoặc bằng tổng số trang có thể hiển thị.  
**Lời giải:**  
Để thực hiện chức năng in các liên hệ ở trang n bất kỳ trong danh sách liên hệ theo thứ tự từ điển, chúng ta cần tính toán và hiển thị chỉ một phần của danh sách liên hệ tùy thuộc vào trang được yêu cầu. Dưới đây là một cách để thực hiện điều này:  
class PhoneBook:  
 def \_\_init\_\_(self):  
 self.contacts = []  
 self.page\_size = 20  
 def insert(self, contact):  
 self.contacts.append(contact)  
 self.contacts.sort(key=lambda x: x.name)  
 def display\_contacts(self, page\_number):  
 total\_pages = (len(self.contacts) + self.page\_size - 1) // self.page\_size  
 if page\_number < 1 or page\_number > total\_pages:  
 print("Invalid page number. Please enter a number between 1 and {}.".format(total\_pages))  
 return  
 start\_index = (page\_number - 1) \* self.page\_size  
 end\_index = min(start\_index + self.page\_size, len(self.contacts))   
 print("Contacts - Page", page\_number, "/", total\_pages)  
 for i in range(start\_index, end\_index):  
 print("Name:", self.contacts[i].name, "- Phone:", self.contacts[i].phone\_number)  
# Sử dụng  
phone\_book = PhoneBook()  
# Thêm các liên hệ  
phone\_book.insert(Contact("Anh An", "0901.000.159"))  
phone\_book.insert(Contact("Bố", "0983 000 131"))  
phone\_book.insert(Contact("Mẹ", "0962 000 481"))  
phone\_book.insert(Contact("ICTLab Station", "024 124 000 313"))  
phone\_book.insert(Contact("John Doe", "123456789"))  
phone\_book.insert(Contact("Alice Smith", "987654321"))  
phone\_book.insert(Contact("Bob Johnson", "456789123"))  
phone\_book.insert(Contact("Anh, Nguyễn Văn Tùng", "0982 000 134"))  
# Hiển thị danh sách liên hệ theo trang  
page\_number = int(input("Enter page number: "))  
phone\_book.display\_contacts(page\_number)  
**Vận dụng 1 trang 48 Chuyên đề Tin học 12**: Sử dụng cây tìm kiếm nhị phân để viết chương trình quản lí danh sách học sinh của một lớp. Thông tin mỗi học sinh gồm mã (duy nhất), tên đầy đủ, ngày sinh. Chương trình cho phép thêm mới thông tin các học sinh, in danh sách sắp xếp theo mã từ nhỏ đến lớn và từ lớn đến nhỏ, tìm kiếm học sinh theo mã.  
**Lời giải:**  
Bản phác thảo Python mẫu cho chương trình quản lí danh sách học sinh của một lớp sử dụng cây tìm kiếm nhị phân:  
class Student:  
 def \_\_init\_\_(self, student\_id, full\_name, date\_of\_birth):  
 self.student\_id = student\_id  
 self.full\_name = full\_name  
 self.date\_of\_birth = date\_of\_birth  
class TreeNode:  
 def \_\_init\_\_(self, student):  
 self.student = student  
 self.left = None  
 self.right = None  
class StudentDatabase:  
 def \_\_init\_\_(self):  
 self.root = None  
 def insert(self, student):  
 self.root = self.\_insert\_recursive(self.root, student)  
 def \_insert\_recursive(self, root, student):  
 if root is None:  
 return TreeNode(student)  
 if student.student\_id < root.student.student\_id:  
 root.left = self.\_insert\_recursive(root.left, student)  
 elif student.student\_id > root.student.student\_id:  
 root.right = self.\_insert\_recursive(root.right, student)  
 return root  
 def search(self, student\_id):  
 return self.\_search\_recursive(self.root, student\_id)  
 def \_search\_recursive(self, root, student\_id):  
 if root is None or root.student.student\_id == student\_id:  
 return root.student if root else None  
 if student\_id < root.student.student\_id:  
 return self.\_search\_recursive(root.left, student\_id)  
 else:  
 return self.\_search\_recursive(root.right, student\_id)  
 def display\_students\_in\_order(self, root):  
 if root:  
 self.display\_students\_in\_order(root.left)  
 print("ID:", root.student.student\_id, "- Name:", root.student.full\_name, "- Date of Birth:", root.student.date\_of\_birth)  
 self.display\_students\_in\_order(root.right)  
 def display\_students\_in\_reverse\_order(self, root):  
 if root:  
 self.display\_students\_in\_reverse\_order(root.right)  
 print("ID:", root.student.student\_id, "- Name:", root.student.full\_name, "- Date of Birth:", root.student.date\_of\_birth)  
 self.display\_students\_in\_reverse\_order(root.left)  
# Sử dụng  
student\_db = StudentDatabase()  
# Thêm học sinh mới  
student\_db.insert(Student(101, "John Doe", "2005-01-15"))  
student\_db.insert(Student(102, "Alice Smith", "2004-08-20"))  
student\_db.insert(Student(103, "Bob Johnson", "2005-03-10"))  
# In danh sách học sinh theo thứ tự mã từ nhỏ đến lớn  
print("Students sorted by ID (ascending):")  
student\_db.display\_students\_in\_order(student\_db.root)  
# In danh sách học sinh theo thứ tự mã từ lớn đến nhỏ  
print("\nStudents sorted by ID (descending):")  
student\_db.display\_students\_in\_reverse\_order(student\_db.root)  
# Tìm kiếm học sinh theo mã  
search\_id = 102  
found\_student = student\_db.search(search\_id)  
if found\_student:  
 print("\nStudent found - ID:", found\_student.student\_id, "- Name:", found\_student.full\_name, "- Date of Birth:", found\_student.date\_of\_birth)  
else:  
 print("\nStudent with ID", search\_id, "not found."  
**Vận dụng 2 trang 48 Chuyên đề Tin học 12**: Sử dụng cây tìm kiếm nhị phân để hiển thị các món trong tệp menu.inp ở Bài 8 theo thứ tự giá tiền tăng dần. Mỗi dòng in ra gồm tên món và giá tiền. Nếu có hai hoặc nhiều món cùng giá tiền thì các món đó được hiển thị theo thứ tự xuất hiện trong tệp menu.inp.u  
**Lời giải:**  
Để hiển thị các món trong tệp menu.inp theo thứ tự giá tiền tăng dần bằng cây tìm kiếm nhị phân, chúng ta cần đọc dữ liệu từ tệp, sau đó chèn mỗi món vào cây tìm kiếm nhị phân dựa trên giá tiền của món. Nếu có nhiều món có cùng giá tiền, chúng ta có thể sử dụng danh sách liên kết hoặc danh sách kết hợp để lưu trữ các món có cùng giá tiền. Dưới đây là một cách để thực hiện điều này:  
class MenuItem:  
 def \_\_init\_\_(self, name, price):  
 self.name = name  
 self.price = price  
class TreeNode:  
 def \_\_init\_\_(self, menu\_item):  
 self.menu\_item = menu\_item  
 self.left = None  
 self.right = None  
 self.same\_price = [] # Danh sách các món có cùng giá tiền  
class MenuDatabase:  
 def \_\_init\_\_(self):  
 self.root = None  
 def insert(self, menu\_item):  
 self.root = self.\_insert\_recursive(self.root, menu\_item)  
 def \_insert\_recursive(self, root, menu\_item):  
 if root is None:  
 return TreeNode(menu\_item)  
 if menu\_item.price < root.menu\_item.price:  
 root.left = self.\_insert\_recursive(root.left, menu\_item)  
 elif menu\_item.price > root.menu\_item.price:  
 root.right = self.\_insert\_recursive(root.right, menu\_item)  
 else:  
 root.same\_price.append(menu\_item)  
 return root  
 def display\_menu\_by\_price(self, root):  
 if root:  
 self.display\_menu\_by\_price(root.left)  
 print("Name:", root.menu\_item.name, "- Price:", root.menu\_item.price)  
 for item in root.same\_price:  
 print("Name:", item.name, "- Price:", item.price)  
 self.display\_menu\_by\_price(root.right)  
# Đọc dữ liệu từ tệp menu.inp và chèn mỗi món vào cây tìm kiếm nhị phân  
menu\_db = MenuDatabase()  
with open("menu.inp", "r") as file:  
 for line in file:  
 name, price = line.strip().split(", ")  
 menu\_item = MenuItem(name, float(price))  
 menu\_db.insert(menu\_item)  
# In danh sách món theo thứ tự giá tiền tăng dần  
print("Menu sorted by price (ascending):")  
menu\_db.display\_menu\_by\_price(menu\_db.root)