Equipment with new general education program, Volume 2, Issue 315 (June 2024)
ISSN 1859 - 0810

Ứng dụng kiểu Dữ liệu danh sách vào bài toán Phân tích thiết kế hệ thống

Vũ Thị Sơn

ThS. Trường Đại học Hải Phòng

Received: 9/5/2024; Accepted: 16/5/2024; Published:20/5/2024

Abstract: *Information technology is one of the scientific fields that is increasingly interested and widely used in all areas of life. With strong advantages that can be applied in many ways in life, information technology has made management work easier. To manage a specific job of a certain agency, the manager needs to perform many actions. Thanks to software, that job becomes much simpler. Therefore, management work using manual methods will not meet the system's tasks and goals and it will also greatly affect the productivity, quality and efficiency of the work. To overcome the above shortcomings, and at the same time thanks to the rapid development of Information Technology, building a "Computer Room" system management program to help with management is an urgent need. today's urgent need to automate work.*

Keywords: *Stack, queue, List, Data, Information sequential pattern, structure, technology.*

1. Đặt vấn đề

Danh sách (DS) là tập hợp các phần tử có cùng kiểu dữ liệu (type) và giữa các phần tử trong DS có một mối quan hệ nhất định nào đó. Số phần tử của DS được gọi là chiều dài của DS. DS có chiều dài (length) bằng 0 được gọi là DS rỗng.

Một DS mà quan hệ lân cận giữa các phần tử được hiển thị ra thì được gọi là DS tuyến tính (Linear list). Ví dụ, DS học sinh trong một lớp, DS các máy tính trong phòng máy là một hình ảnh của DS tuyến tính (số thứ tự của HS là thể hiện rất rõ mối quan hệ lân cận). Thao tác này bao gồm những công việc sau:

- Xin cấp phát ô nhớ cho một nút mới **p.**
- Đưa giá trị mới vào trường info của con trỏ **p.**
- Gán giá trị NULL cho trường link của **p.**
- Gắn nút **p** vào cuối danh sách.
- Cho trường rear của con trỏ **Q** trỏ vào **p.**
- Nếu **p** là nút duy nhất của danh sách thì cho con trỏ front trỏ vào nút này.

2.Nội dung nghiên cứu

2.1. Các phép toán trên danh sách

Tuỳ thuộc vào đặc điểm của từng loại DS sẽ xác định được các phép toán thường xuyên thực hiện trên DS đó là gì. Tuy nhiên, có một số phép phổ biến đối với DS, cụ thể là:

(1). Phép toán khởi tạo (create) một danh sách

Là thao tác khởi tạo nội dung của các phần tử trong DS, bởi vậy cần xác định rõ chiều dài của DS trước khi khởi tạo. Trong một số trường hợp chúng ta chỉ khởi tạo giá trị và trạng thái ban đầu của DS.

(2). Thêm (insert) một phần tử vào trong danh sách

Là thao tác bổ sung thêm một phần tử vào trong DS. Nếu bổ sung thành công thì chiều dài của DS sẽ tăng lên 1 đơn vị. Vị trí chèn thêm một phần tử vào trong DS sẽ tuỳ thuộc vào từng bài toán cụ thể.

(3). Tìm kiếm (search) một phần tử trong danh sách

Là thao tác sử dụng các kỹ thuật tìm kiếm để tìm kiếm các phần tử trong DS thoả mãn một (hoặc một số) ràng buộc nào đó. Kết quả của bài toán tìm kiếm có thể là thành công hoặc thất bại.

(4). Loại bỏ (delete) một phần tử ra khỏi danh sách

Là thao tác loại bỏ (xoá) một phần tử ra khỏi DS, nếu loại bỏ thành công thi chiều dài của DS sẽ giảm đi 1 đơn vị. Khi thực hiện thao tác loại bỏ, trước tiên cần phải kiểm tra xem phần tử cần loại bỏ có tồn tại trong DS hay không (bài toán tìm kiếm).

(5). Cập nhật (update) giá trị của một phần tử trong danh sách

Là thao tác sửa đổi nội dung của một phần tử trong DS. Tương tự như thao tác loại bỏ, trước khi tiến hành sửa đổi cần phải kiểm tra xem phần tử cần sửa đổi có tồn tại trong DS hay không.

(6). Sắp xếp (sort) các phần tử trong danh sách

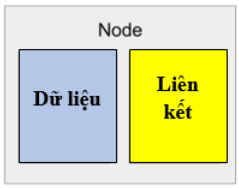
Là thao tác sử dụng các thuật toán sắp xếp để sắp xếp các phần tử trong DS theo một trật tự xác định (tiêu chí sắp xếp) nào đó.

2.2. Danh sách nối đơn

DS liên kết là một cấu trúc dữ liệu có kiểu truy cập tuần tự. Mỗi phần tử trong DS liên kết có chứa thông tin về phần tử tiếp theo, qua đó ta có thể truy

cập tới phần tử này. R. Sedgewick (Alogrithms in Java - 2002) định nghĩa DS liên kết như sau:

DS liên kết là một cấu trúc dữ liệu bao gồm một tập các phần tử, trong đó mỗi phần tử (Node) bao gồm hai thành phần: (i) dữ liệu của nút và (ii) thông tin về liên kết tới nút tiếp theo trong DS.



Có thể nói DS liên kết là một cấu trúc dữ liệu được định nghĩa kiểu đệ quy, vì trong định nghĩa một nút của DS có tham chiếu tới khái niệm nút. Thông thường, một nút thường có liên kết trỏ tới một nút khác, tuy nhiên nó cũng có thể trỏ tới chính nó.

DS liên kết có thể được xem như là một sự bố trí tuần tự các phần tử trong một tập. Bắt đầu từ một nút, ta coi đó là phần tử đầu tiên trong DS. Từ nút này, theo liên kết mà nó trỏ tới, ta có nút thứ 2, được coi là phần tử thứ 2 trong DS, v.v. cứ tiếp tục như vậy cho đến hết DS.

Nút cuối cùng có thể có liên kết là một liên kết null, tức là không trỏ tới nút nào, hoặc nó có thể trỏ về nút đầu tiên để tạo thành một vòng.

Đặc điểm của DS liên kết là sử dụng mối nối và con trỏ để tổ chức DS. Do đó ngoài các thao tác cơ bản của DS nói chung, ta cần thêm thao tác tạo và cấp phát bộ nhớ cho một nút mới.

Khởi tạo một danh sách rỗng

Là thao tác đầu tiên và rất quan trọng đối với DS, nếu thiếu thao tác này chương trình sẽ gây lỗi khi thực hiện.

Khởi tạo một DS rỗng tức là gán giá trị NULL cho con trỏ chứa địa chỉ nút đầu tiên của DS.

```
void initializeList (listnode   *P)
{
                *P=NULL;
}
```

Tạo và cấp phát bộ nhớ cho một nút

Đặc điểm của DS liên kết là sử dụng biến con trỏ và cấp phát động, mỗi khi cần lưu thông tin vào một nút mới ta phải xin máy cấp phát số ô nhớ đủ cho một nút

Input: x //giá trị trường info của nút mới

Process:

Bước 1: Tạo nút mới.

- Xin cấp phát bộ nhớ cho nút mới (con trỏ **q**);
- Gán giá trị **x** cho trường **info** của con trỏ **q**;
- Gán giá trị **NULL** cho trường **link** của con trỏ **q.**

Bước 2: Trả kết quả (return (**q**)).

Output: Nút mới được tạo.

Chèn một nút mới vào danh sách

Mỗi DS liên kết luôn phải có một con trỏ lưu trữ địa chỉ của nút đầu tiên, ta không thể truy cập vào DS nếu không biết địa chỉ của nút đầu tiên này. Để chèn nút mới vào trước nút đầu tiên ta chỉ cần cho trường link của nút mới *q* trỏ vào *P* (con trỏ chứa địa chỉ nút đầu tiên), sau đó cho con trỏ *P* trỏ vào nút mới *q*.

Input:

P //là con trỏ trỏ vào nút đầu tiên của DS

x // giá trị trường info của nút mới

Process:

Bước 1: Tạo và cấp phát bộ nhớ cho nút mới **q.**

Bước 2: Xét các trường hợp:

- Nếu danh sách rỗng (**P**==NULL):

- q là nút đầu tiên và duy nhất của DS (Cho **P** trỏ vào **q).**

- Nếu danh sách không rỗng (**P**!= NULL): Ghép nút q

- vào đầu danh sách bằng cách:

- Cho trường link của con trỏ **q** trỏ vào **P.**

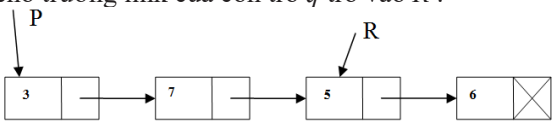
- Cho con trỏ **P** trỏ vào **q.**

Output: Nút mới **q** là nút đầu tiên của DS.

Muốn chèn một nút mới vào danh sách ta phải tìm tới địa chỉ của nút cuối cùng, thao tác này cần một con trỏ phụ *m* bắt đầu từ nút đầu tiên và dịch chuyển dần qua từng nút cho tới nút cuối cùng (nút có trường link bằng NULL). Để gắn nút mới vào DS bằng cách cho trường link của con trỏ *m* trỏ vào nút mới *q*.

Chèn một nút mới vào trước nút R trong danh sách

Thao tác này cần một con trỏ phụ *m,* bắt đầu từ nút đầu tiên và dịch chuyển dần qua các nút cho tới nút ngay trước *R* và gắn nút mới vào DS bằng cách cho trường link của con trỏ *m* trỏ vào nút mới *q,* rồi cho trường link của con trỏ *q* trỏ vào *R* .



Input:

P *//là con trỏ trỏ vào nút đầu tiên của DS*

R *// là con trỏ trỏ vào nút bất kỳ trong DS*

x *// giá trị trường info của nút mới*

Process:

Bước 1: Xét các trường hợp

- Nếu DS rỗng (P==NULL)

  • Thông báo danh sách rỗng và kết thúc

- Nếu DS không rỗng (P! = NULL) chuyển sang bước 2.

Bước 2:

- Nếu **R** trỏ vào nút đầu tiên (**R==P**):

Chèn nút **q** vào đầu DS: Gọi hàm Insertfirst(p,x)

- Nếu **R** không trỏ vào nút đầu tiên (**R!=P**)

• Tạo và cấp phát bộ nhớ cho một nút mới **q**

• Tìm tới nút ngay trước nút **R** trong DS:

Input:

P *//là con trỏ trỏ vào nút đầu tiên của DS*

x /**là khóa tìm kiếm và có kiểu dữ liệu phù hợp với trường khóa của các nút trong DS*/*

Process:

Bước 1: Xét các trường hợp

- Nếu danh sách rỗng (**P**==NULL):

Thông báo danh sách rỗng và kết thúc

- Nếu danh sách không rỗng (**P!**=NULL): Chuyển sang bước 2

Bước 2: Tìm nút có giá trị trường khóa của **info** bằng **x**

- Lặp lại công việc sau cho tới khi tìm thấy hoặc hết DS (từ nút đầu tiên):

• Nếu giá trị trường khóa của info trùng với x thì return (địa chỉ của nút này);

• Nếu khác thì dịch chuyển sang nút đứng sau.

- Không tìm thấy: return (NULL)

Output: Địa chỉ của nút được tìm thấy hoặc giá trị NULL nếu không tìm thấy.

Giả thiết, hàm tìm kiếm xem trong danh sách nối đơn lưu trữ những số nguyên, nếu có một nút mà giá trị trường info bằng x (là một số cần tìm) thì hàm trả ra địa chỉ ô nhớ của nút đó, ngược lại hàm trả ra giá trị NULL.

2.3. Danh sách nối kép

Phần tử của DS nối kép được lưu trữ trong một phần tử nhớ mà ta gọi là nút (Node). Mỗi nút bao gồm một số từ máy kế tiếp. Các nút này có thể nằm bất kỳ ở chỗ nào trong bộ nhớ. Trong mỗi nút, ngoài phần thông tin ứng với mỗi phần tử, còn có chứa địa chỉ của phần tử đứng ngay trước và sau nó trong DS. Qui cách của mỗi nút có thể hình dung như sau:

PLeft	INFO	PRight

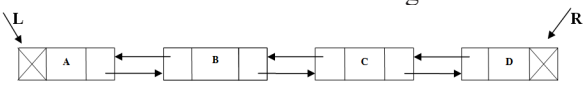
Trong đó:

Trường INFO chứa thông tin của mỗi nút trong DS.

Trường PLeft chứa địa chỉ của nút đứng ngay trước, riêng nút đầu tiên không có nút đứng trước nên PLeft có giá trị NULL.

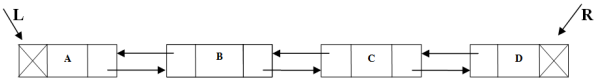
Trường Pright chứa địa chỉ của nút đứng ngay sau, riêng nút cuối cùng không có nút đứng sau nên PRight có giá trị NULL.

Để có thể truy nhập vào mọi nút trong DS ta cần phải có hai con trỏ L và R. Con trỏ L trỏ tới nút đầu tiên và con trỏ R trỏ tới nút cuối cùng.



Output: Nút mới được tạo.

Nếu dùng mũi tên để chỉ mối nối, ta sẽ có hình ảnh một DS nối đơn như sau:



Khi danh sách rỗng thì L=R=NULL

Để lưu trữ DS liên kết kép trong ngôn ngữ C, có thể dùng cấu trúc tự trỏ như sau:

```
struct node
{ElementType        info;
      struct node*  PLft;
struct node*        PRight;
};
typedef struct node* DoubleListnode;
      else
                  {(*R)->PRight=q;
                        q->PLeft=*R;
*R=q;             }}
```

3. Kết luận

Do bản chất động của DS liên kết, kích thước của DS liên kết có thể linh hoạt hơn nhiều so với mảng. Kích thước của DS không cần phải khai báo trước, bất kỳ lúc nào có thể tạo mới một phần tử và thêm vào vị trí bất kỳ trong DS. Nói cách khác, mảng là một tập có số lượng cố định các phần tử, còn DS liên kết là một tập có số lượng phần tử không cố định. Tác giả chọn hai kiểu dữ liệu để so sánh cho thấy độ chính xác và thể hiện các thao tác tìm kiếm, bổ sung, thêm sửa, xóa hợp lý và có tính logic, hiệu quả cao.

Tài liệu tham khảo

1. M. D. Buhmann (2013), *Radial Basis Functions: Theory and Implementations*, *Cambridge University Press, Cambridge*

2. Boztosun et al (2001)., *Thin-Plate Spline Radial Basis Function Scheme for Advection-Diffusion Problems, Electronic Journal of Boundary Elements, Vol. BETEQ 2001, No. 2, pp. 267-282 Morton KW and Mayers DF, Numerical Solution of Partial Differential Equations, Cambridge University Press*

3. Schaback R (2007), *Lecture notes on Reconstruction of Multivariate Functions from Scattered Data*