Các cấu trúc dữ liệu cơ bản

Nội dung

- Danh sách liên kết
- Hàng đợi
- Ngăn xếp

Các kiểu dữ liệu tuyến tính

- Kiểu mảng
  - Thuận tiện với dữ liệu có số phần tử cố định
  - Được lưu trữ liên tục trong bộ nhớ, truy cập ngẫu nhiên
- Cần kiểu dữ liệu có số phần tử động
  - Dễ dàng thêm/bớt phần tử

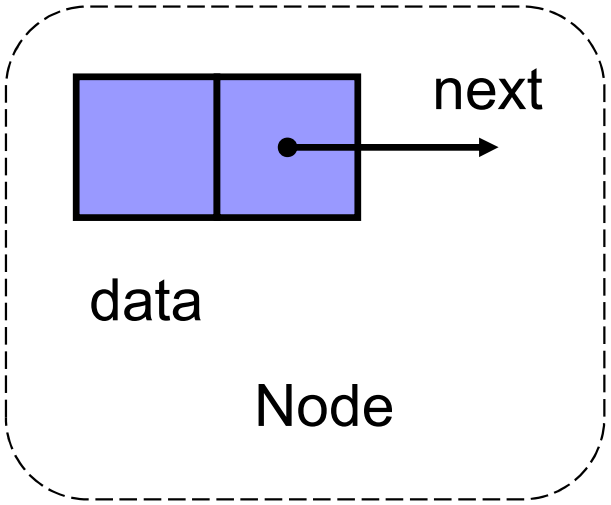
Danh sách liên kết - Linked list

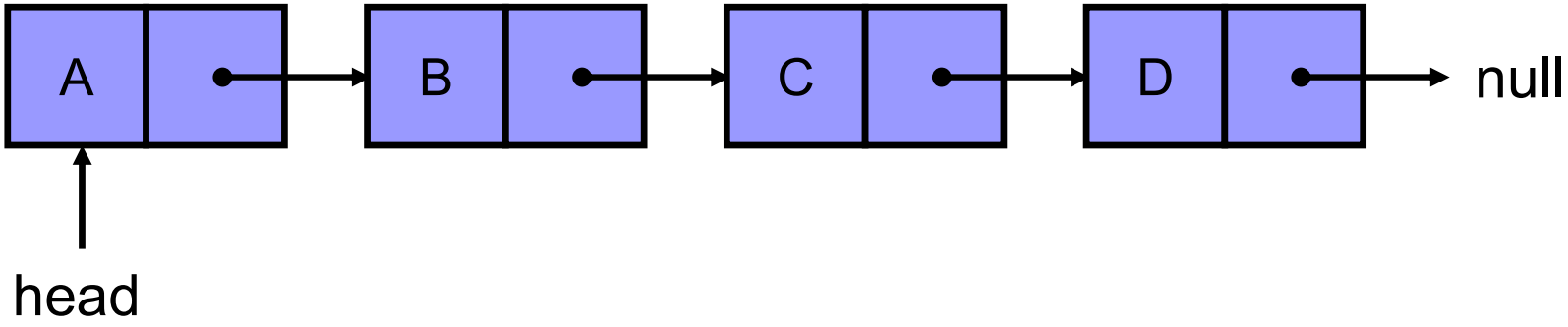
- Các phần tử không được lưu trữ liên tục trong bộ nhớ mà được kết nối thông qua con trỏ
- Dễ dàng thêm/bớt phần tử
  - Cấp phát bộ nhớ động, hiệu quả đối với dữ liệu không biết trước số lượng
  - Thuận tiện đối với kiểu dữ liệu có kích thước lớn

Danh sách liên kết đơn

Singly linked list

- Dãy các node được kết nối
- Mỗi node lưu trữ
  - Dữ liệu
  - Con trỏ tới node tiếp theo





Các thao tác đối với danh sách

- Khởi tạo danh sách liên kết
  - head = null;
  - head = new Node();
- Duyệt danh sách
- Thêm phần tử
- Xóa phần tử

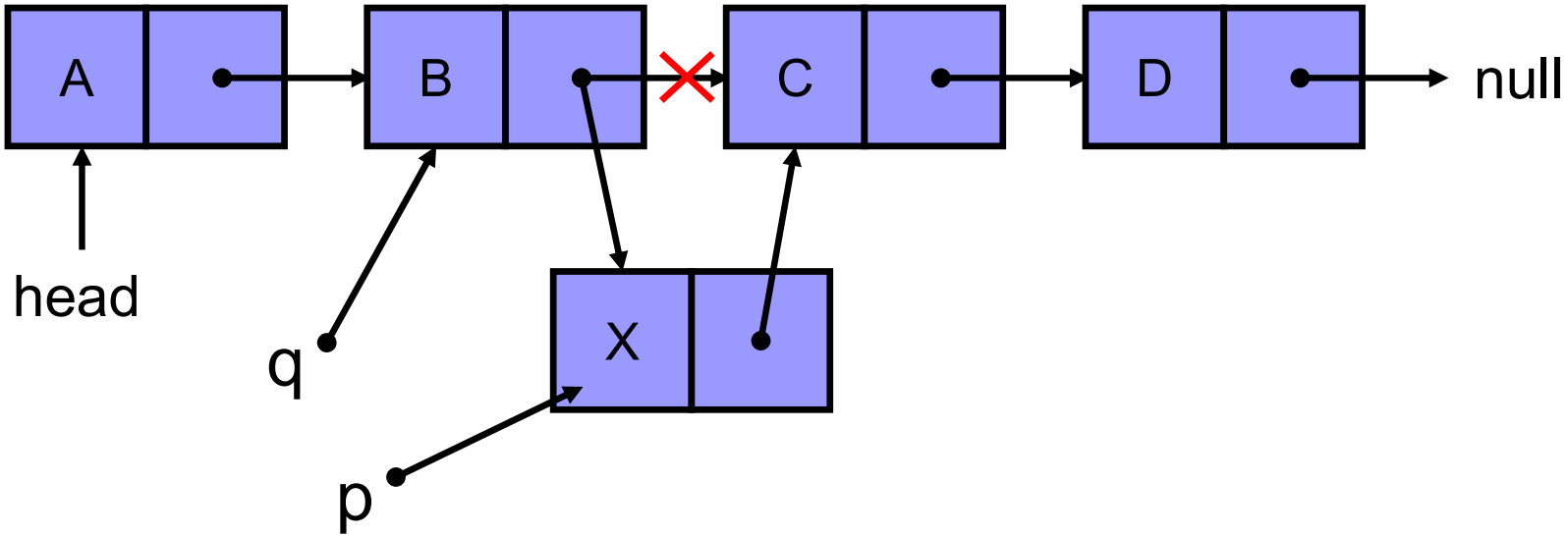
Duyệt tìm phần tử

- Duyệt từ đầu danh sách cho đến khi gặp phần tử hoặc hết danh sách (next = null)

```
Node* p = head;
while (p != null) {
    if (p->data == key) break;
    p = p->next;
}
return p;
```

Thêm phần tử

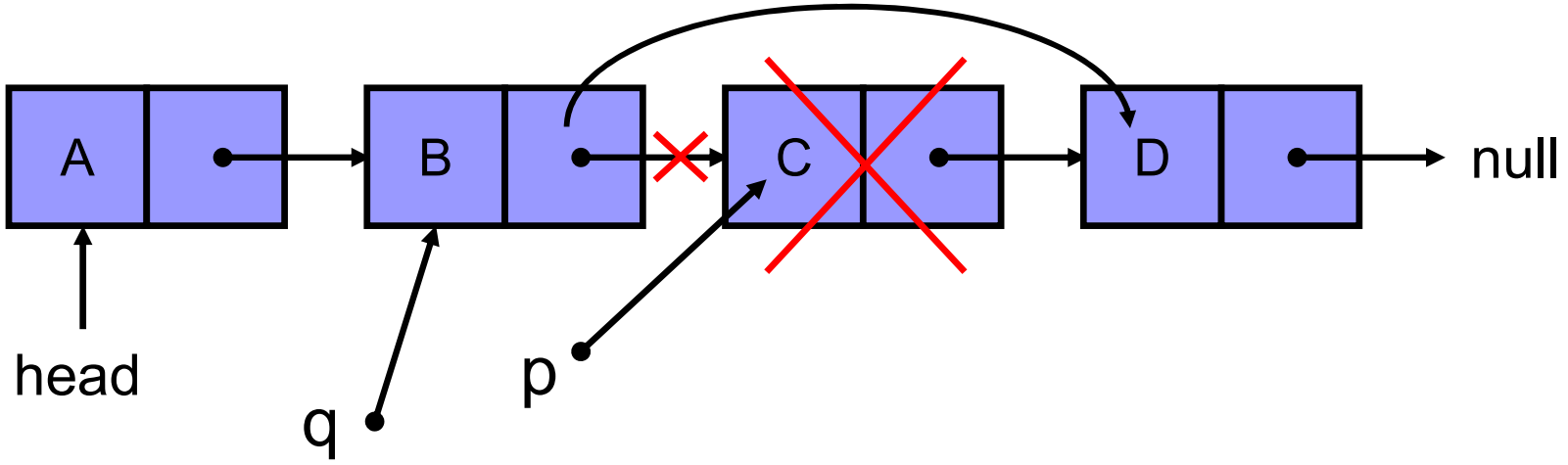
Chèn phần tử vào sau node có giá trị B




```
1. Duyệt tìm node B (q)
2. Chèn p vào sau node B
     p->next = q->next;
     q->next = p;
```

Xóa phần tử

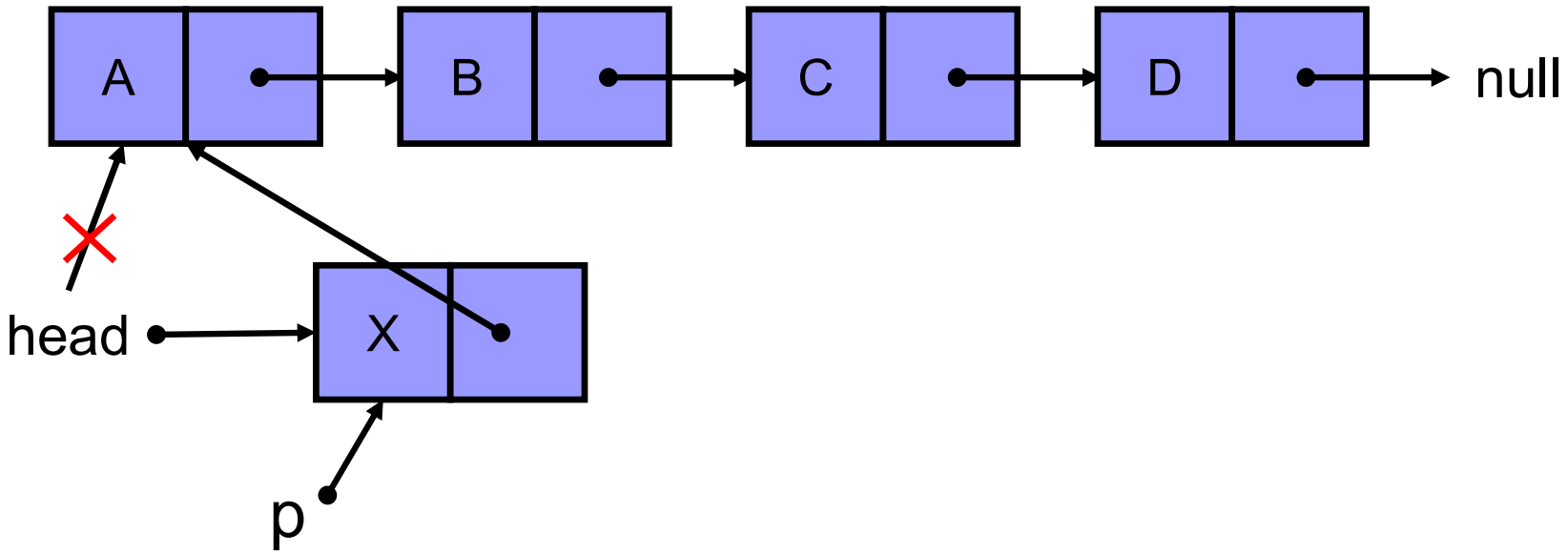
Xóa phần tử được trỏ đến bởi p



1. Duyệt tìm phần tử đứng trước (q)
2. Xóa

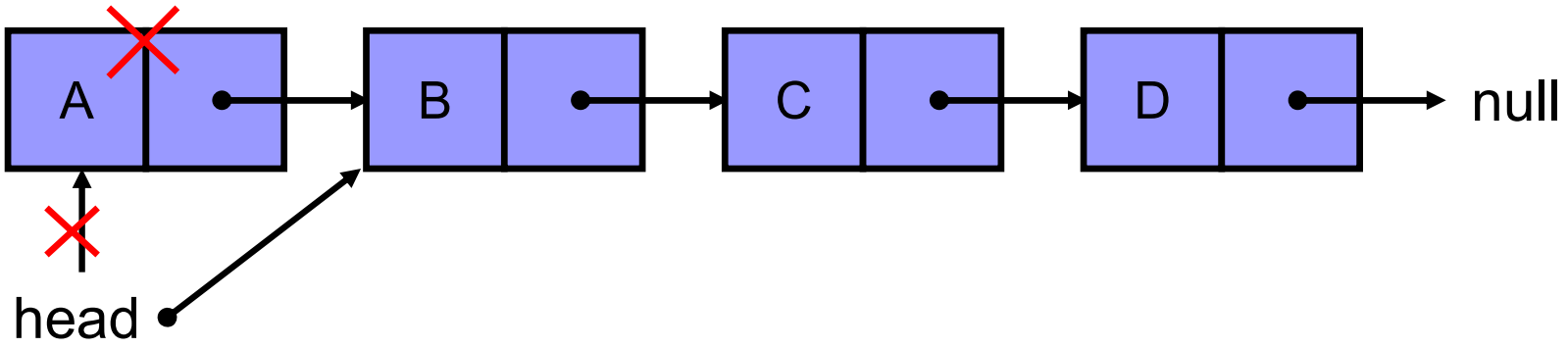
```
q->next = p->next;
delete p;
```

Thêm đầu danh sách



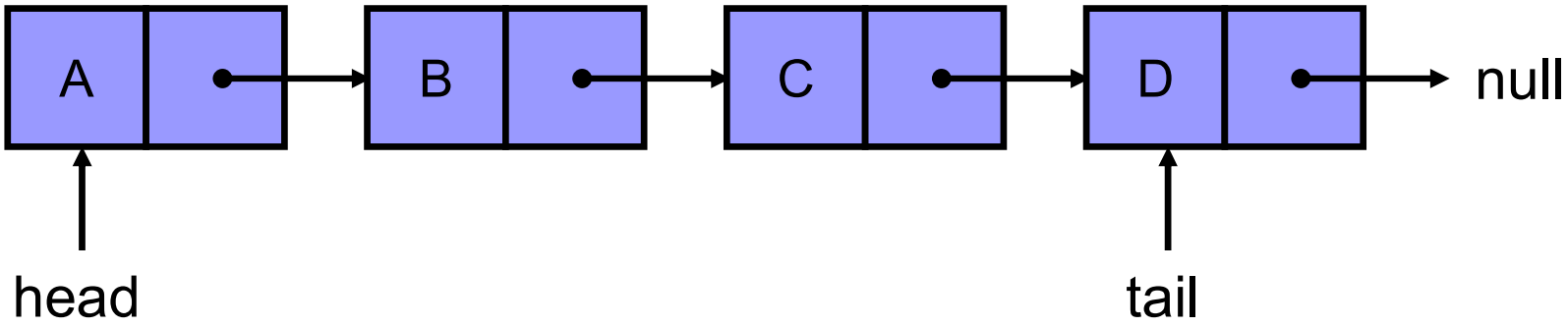

```
p->next = head;
head = p;
```

Xóa nút đầu danh sách




```
if (head) {
    p = head;
    head = p->next;
    delete p;
}
```

Danh sách liên kết với con trỏ đuôi



- Dễ dàng thêm phần tử vào cuối danh sách
- Khi thêm hay xóa nút cuối danh sách cần cập nhật con trỏ tail

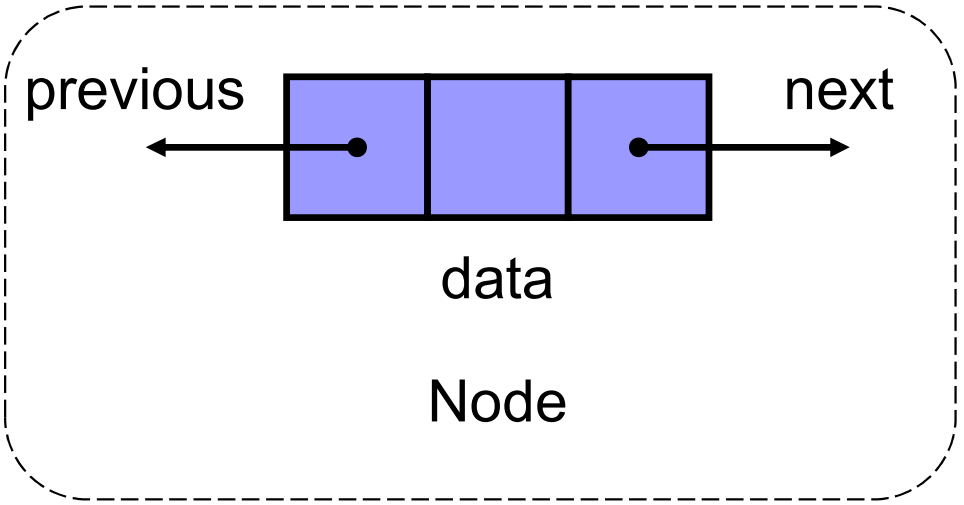
Xóa nút cuối danh sách

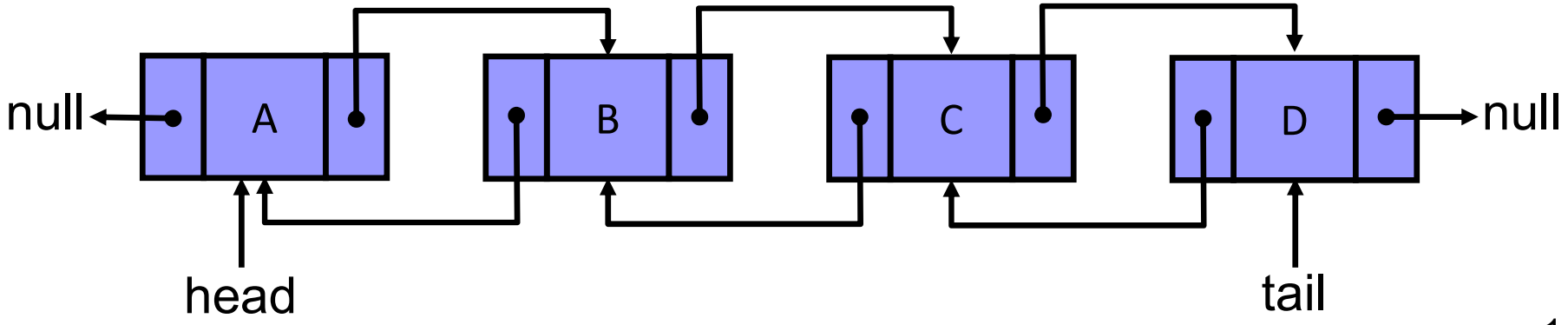
- Với danh sách liên kết đơn, việc xóa phần tử cuối cùng tốn nhiều thời gian
  - Phải duyệt tuần tự từ đầu danh sách để biết phần tử kề trước đó
- Thuận tiện hơn nếu có thể duyệt danh sách theo hai chiều
  - Bổ sung thêm con trỏ trỏ tới nút trước nó

Danh sách liên kết đôi

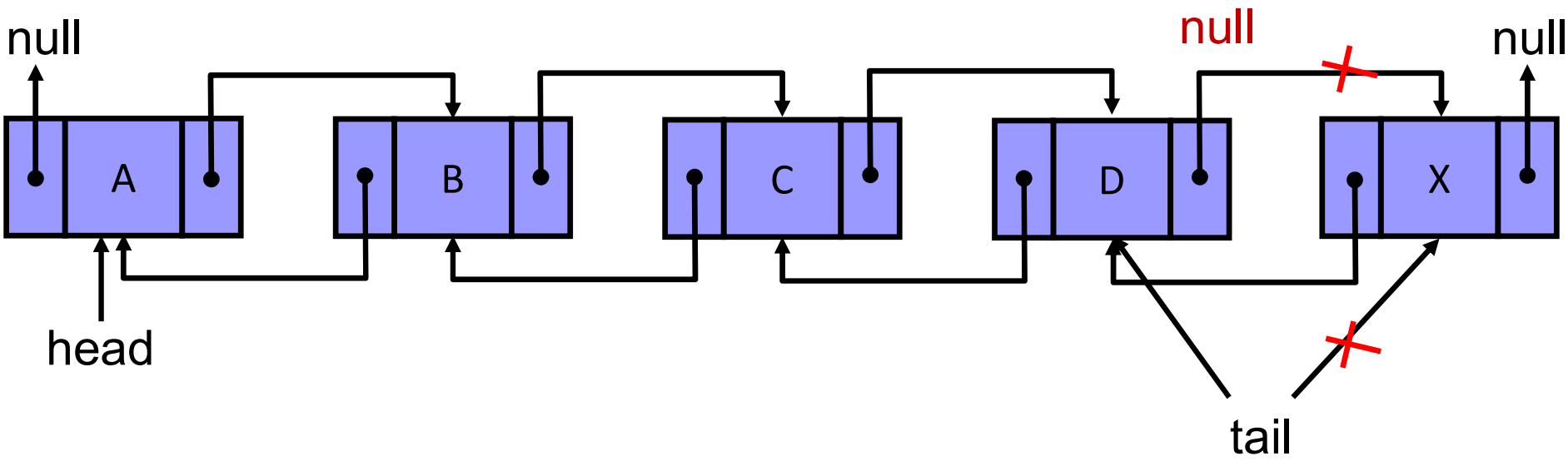
Doubly linked list

- Mỗi node lưu trữ
  - Dữ liệu
  - Con trỏ tới node đứng sau
  - Con trỏ tới node đứng trước





Xóa phần tử cuối danh sách

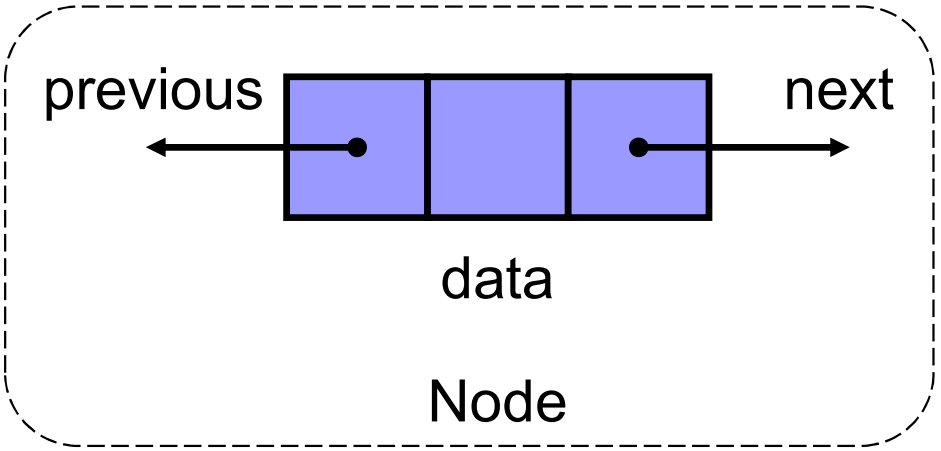


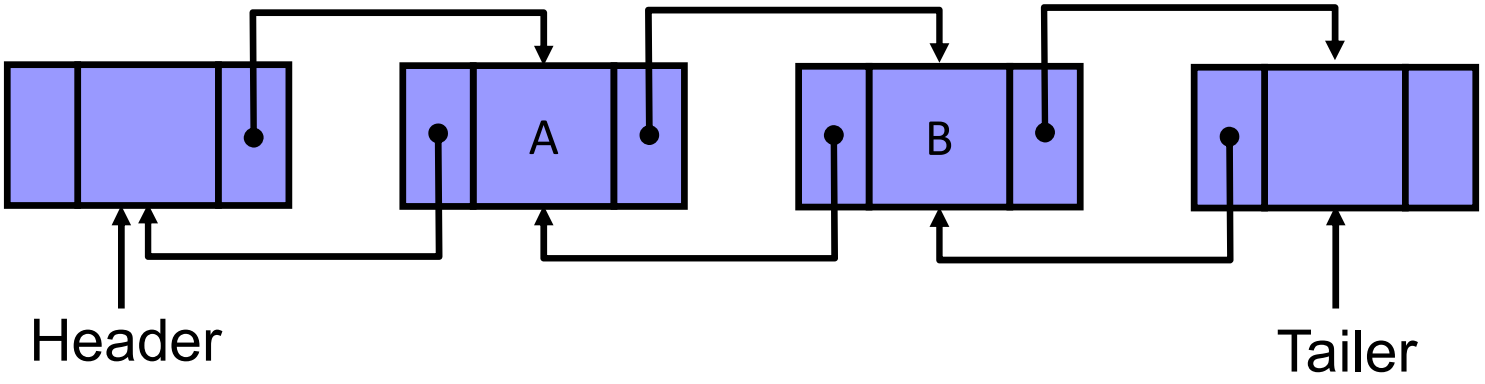

```
if (tail) {
    p = tail;
    tail = tail->prev;
    tail->next = null;
    delete p;
}
```

Nếu xóa danh sách chỉ có 1 phần tử duy nhất thì cần điều chỉnh cả head

Cải tiến danh sách liên kết đôi

- Bổ sung node chặn đầu và chặn cuối danh sách
  - Là các node đặc biệt không lưu dữ liệu
  - Dễ dàng quản lý danh sách



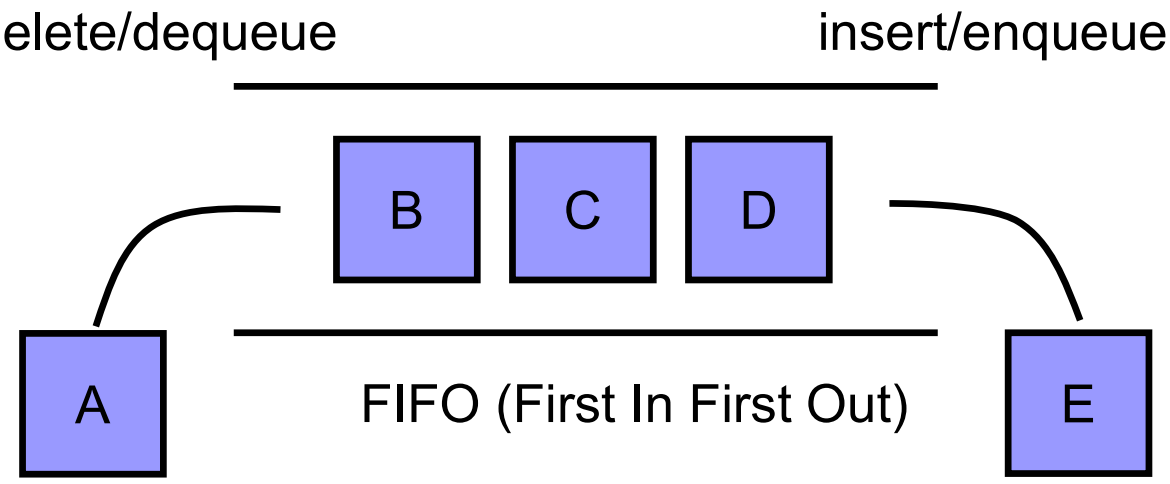


Danh sách liên kết

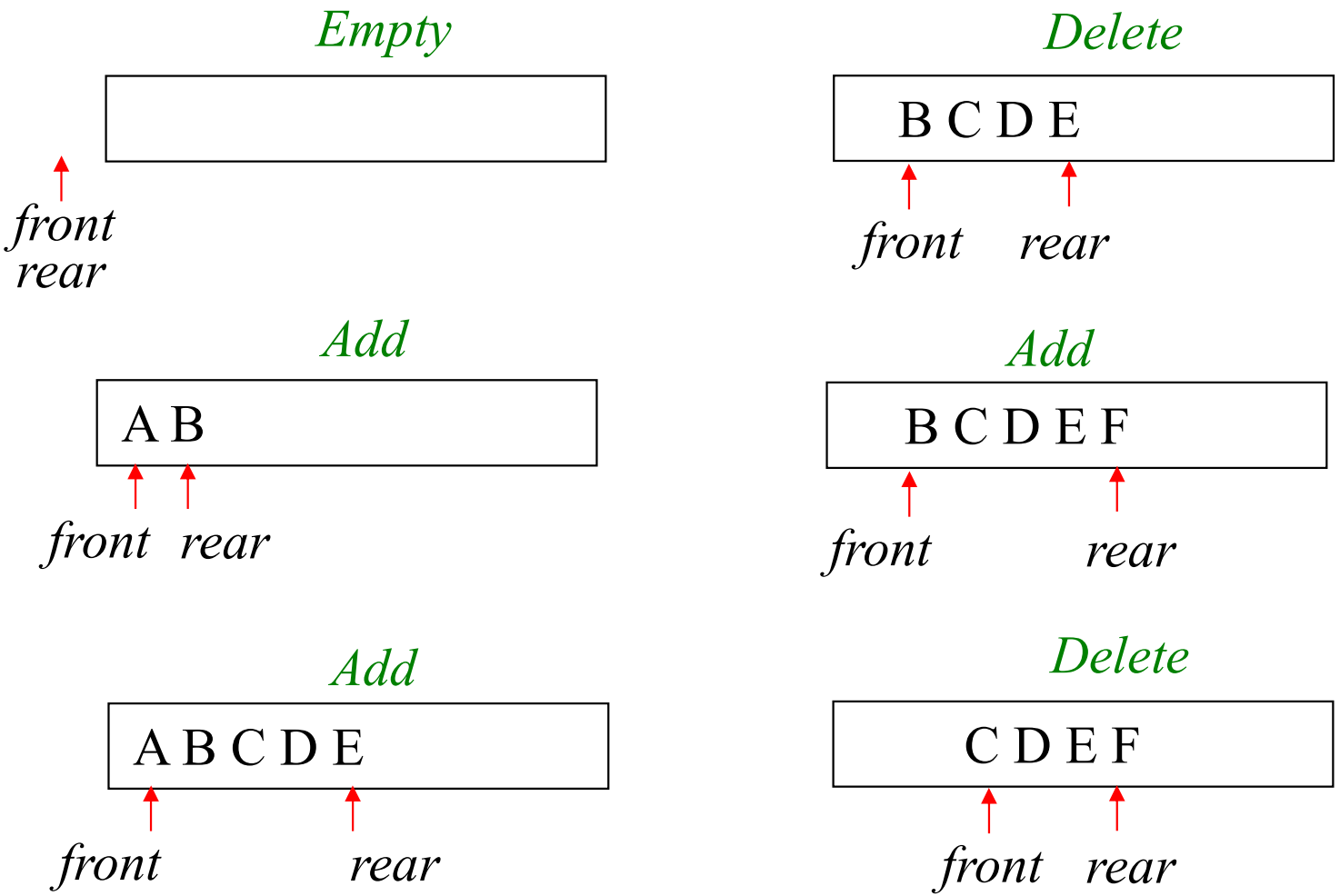
- Dễ dàng thêm, bớt phần tử
- Là cấu trúc cơ bản để cài đặt các kiểu cấu trúc khác
  - Ngăn xếp, hàng đợi, cây (tree)
- Danh sách liên kết vs. mảng
  - Không cho phép truy cập ngẫu nhiên
  - Cần bộ nhớ để lưu giá trị con trỏ
  - Độ phức tạp?
    - Truy vấn
    - Thêm / bớt

Hàng đợi - queue

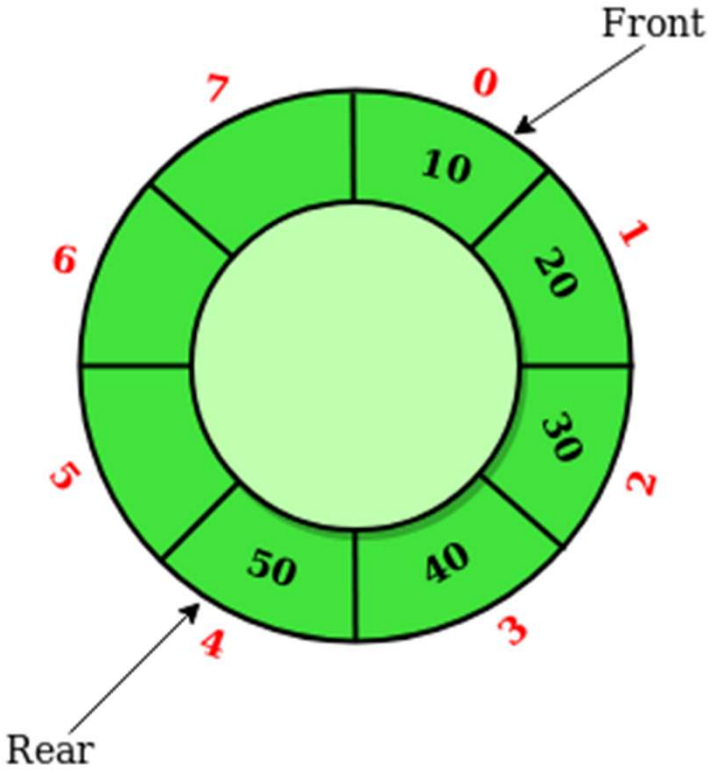
- Phần tử đưa vào trước thì được lấy ra trước (First In First Out – FIFO)

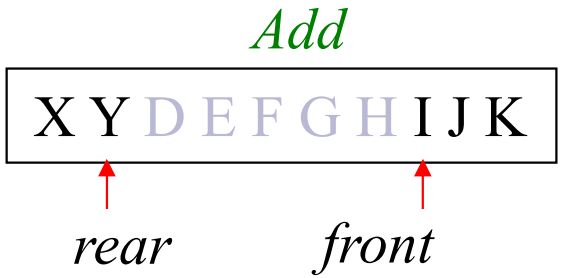


Hàng đợi: thêm, bớt phần tử



Cài đặt hàng đợi dùng mảng

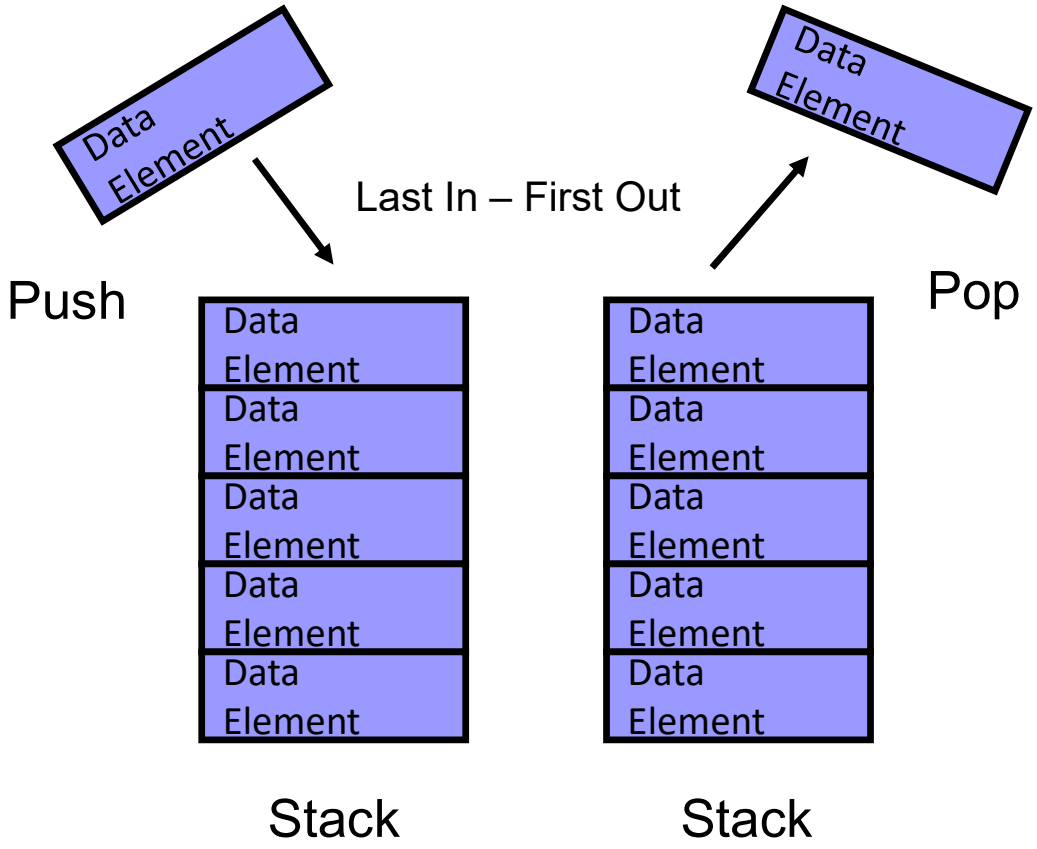




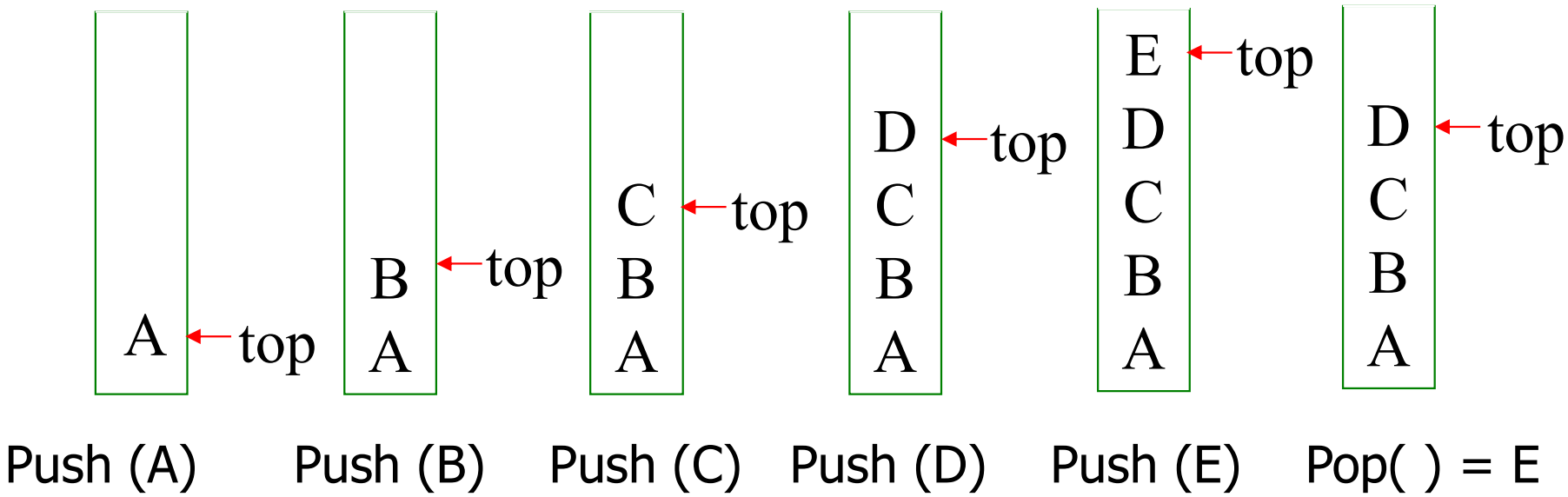
- Xoay vòng hàng đợi để đảm bảo hoạt động liên tục
- Các trường hợp đặc biệt
  - Hàng đợi rỗng, không có phần tử để lấy ra
  - Hàng đợi đầy, không thể thêm phần tử

Ngăn xếp (stack)

- Tập hợp các phần tử với 2 thao tác cơ bản
  - Thêm phần tử - Push
  - Lấy phần tử - Pop
- Phần tử đưa vào sau thì được lấy ra trước
  - Last In, First Out - LIFO

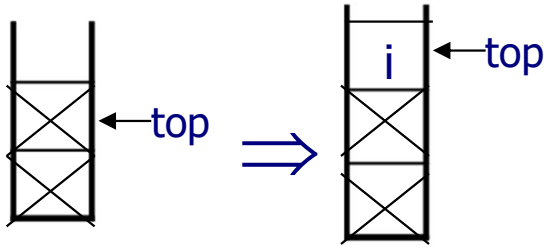


Ví dụ


A
top
B
A
top
C
B
A
top
D
C
B
A
top
E
D
C
B
A
top
D
C
B
A
top
Push (A)
Push (B)
Push (C)
Push (D)
Push (E)
Pop( ) = E

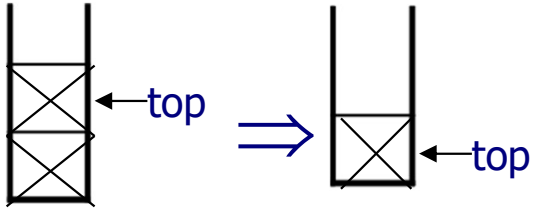
Thao tác trên stack

1. Tạo stack: tạo một stack rỗng S
2. Thêm phần tử (Push)



Push(i,S)

3. Lấy ra (Pop):

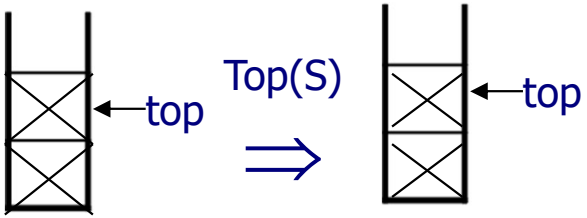


var←Pop(S)

4. IsEmpty(S):

{ true if S = empty → top = -1
  false

5. Xem giá trị đỉnh (Top)

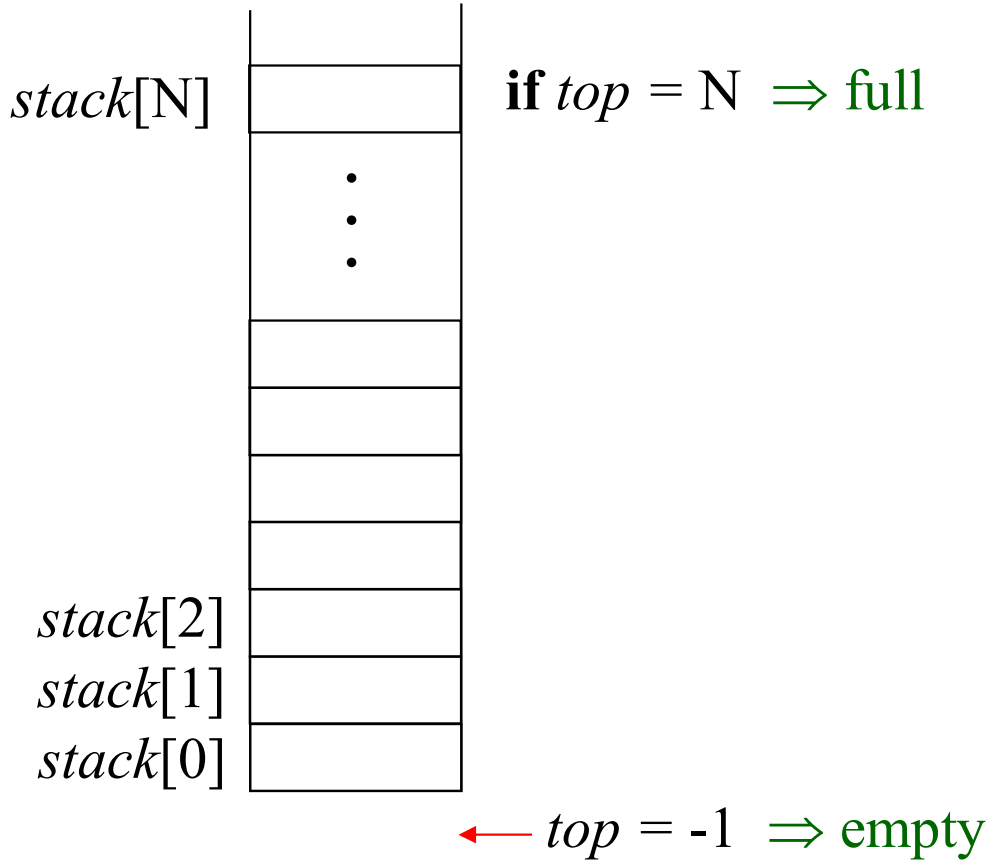


var = Top(S)

Cài đặt stack dùng mảng

```
bool Push(type item) {
    if (top >= N) {
        stack_full_msg();
        return false;
    }
    stack[++top] = item;
    return true;
}
```

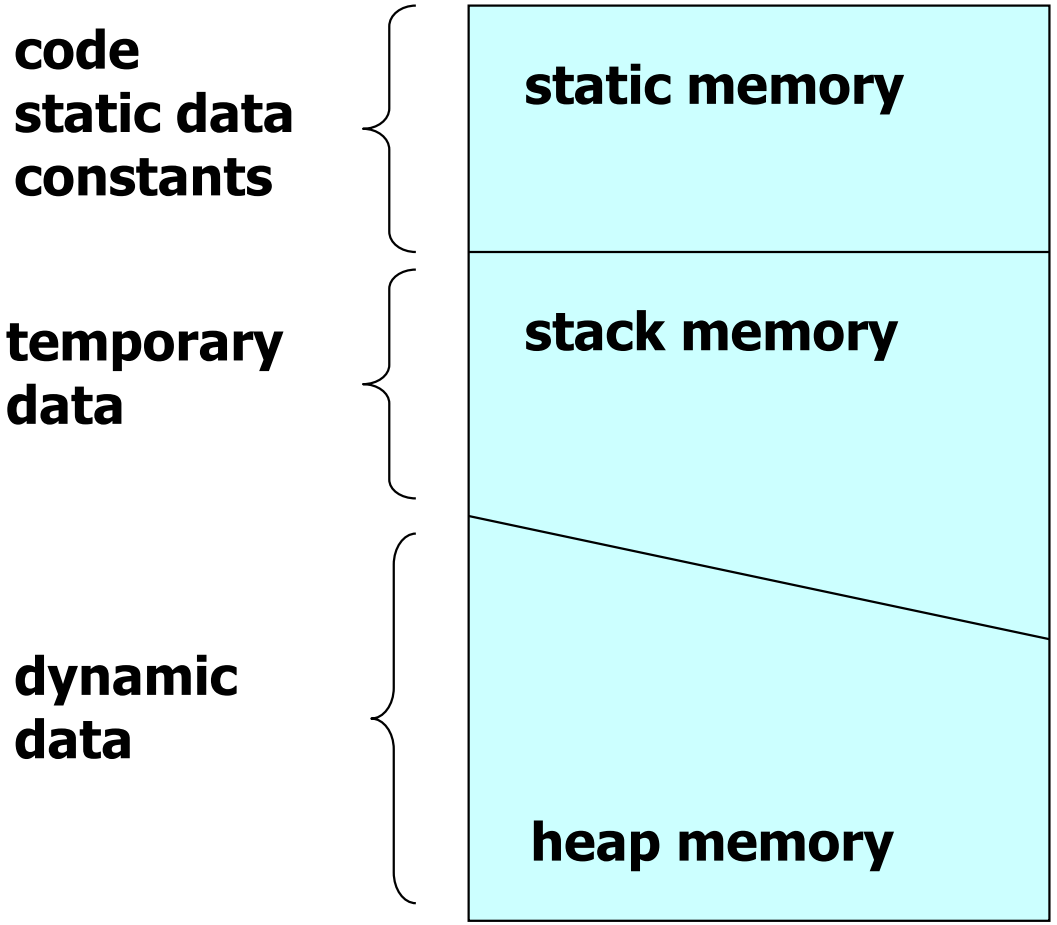
constant N = 1000
type stack[N];



Stack và gọi hàm (function call)

- Các chương trình con có thể gọi lồng nhau nhiều lần, hoặc gọi đệ qui. Cần có cơ chế lưu trữ bộ nhớ sử dụng, giá trị trả về và địa chỉ quay lại
- Stack được ứng dụng để quản lý việc gọi hàm
  - Là cơ chế bắt buộc của kiến trúc máy tính hiện đại
  - Gồm vùng bộ nhớ call stack và thanh ghi địa chỉ stack

3 vùng bộ nhớ cho chương trình

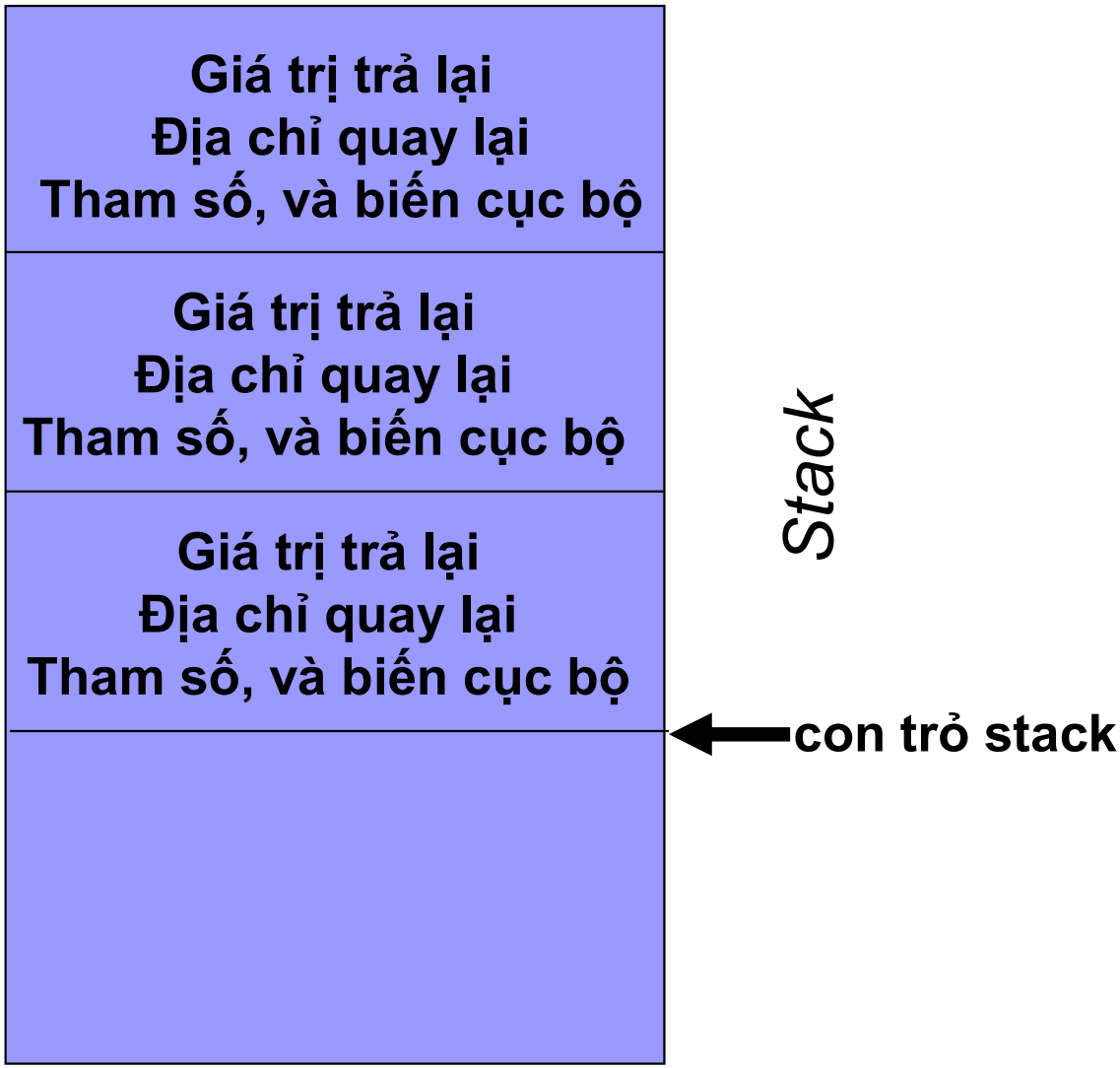


Gọi hàm và bộ nhớ ngăn xếp

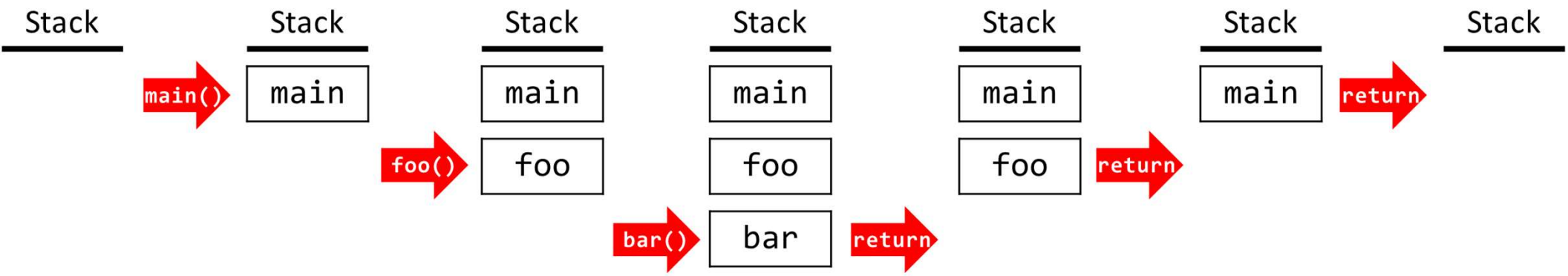
Vùng nhớ cho lời gọi hàm thứ nhất

Vùng nhớ cho lời gọi hàm thứ hai

Vùng nhớ cho lời gọi hàm thứ ba



```
void bar() { }
void foo() {
    bar();
}
int main() {
    foo();
}
```



Ví dụ: Tính giá trị biểu thức

- Biểu thức sử dụng các bộ dấu ngoặc {}, [], (), kiểm tra xem các dấu ngoặc có được đặt hợp lệ không

{ (( ) [ ]) } → true

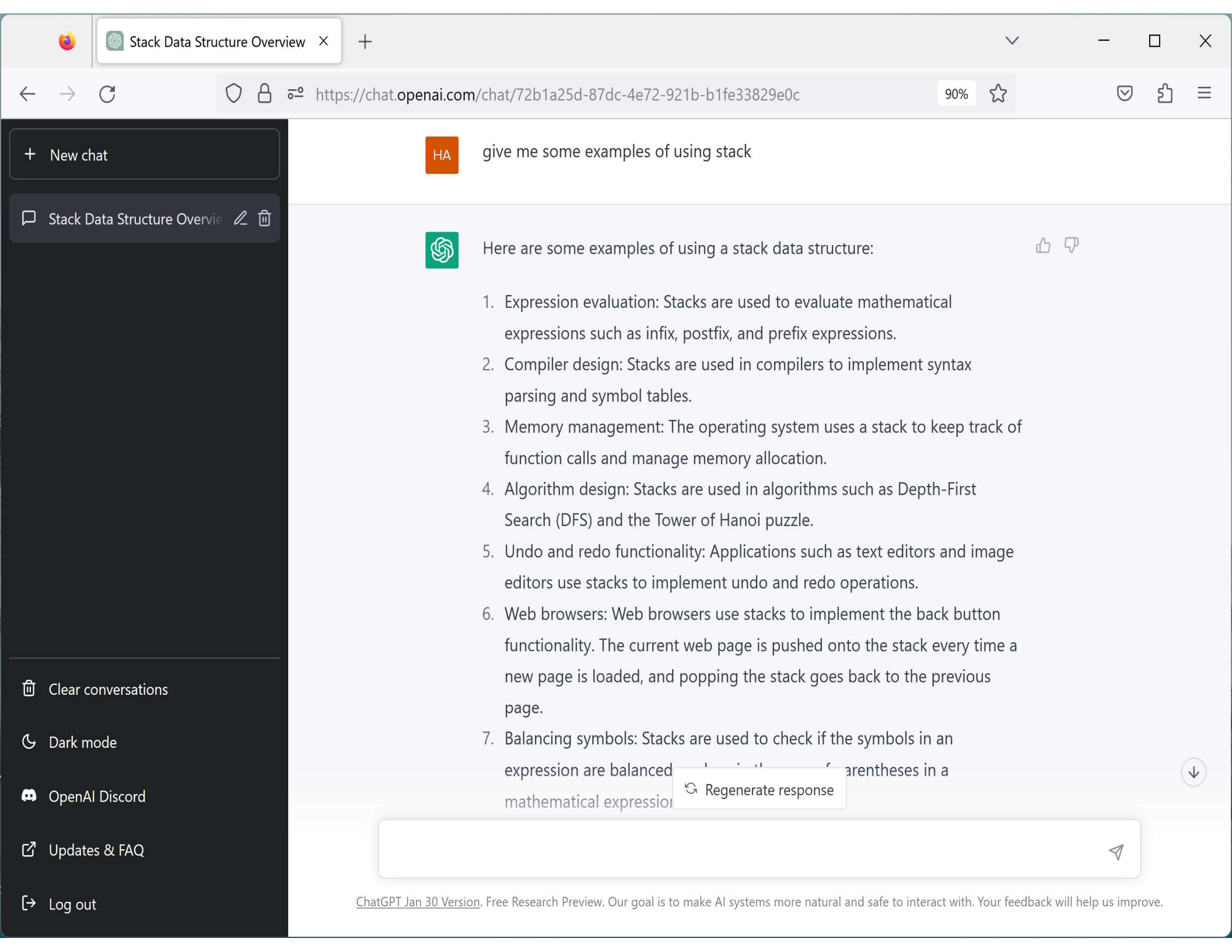
(()]) → false

- Tính giá trị biểu thức

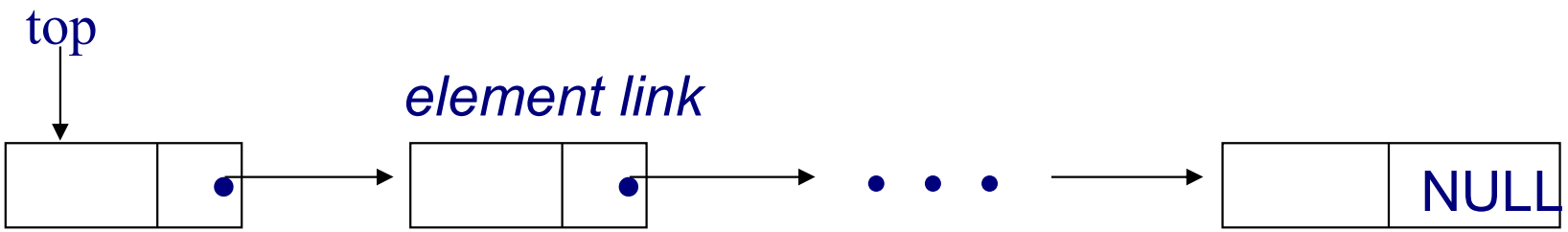
(1 + ((2 + 3) * (4 + 5)))

Dijkstra's two-stack algorithm

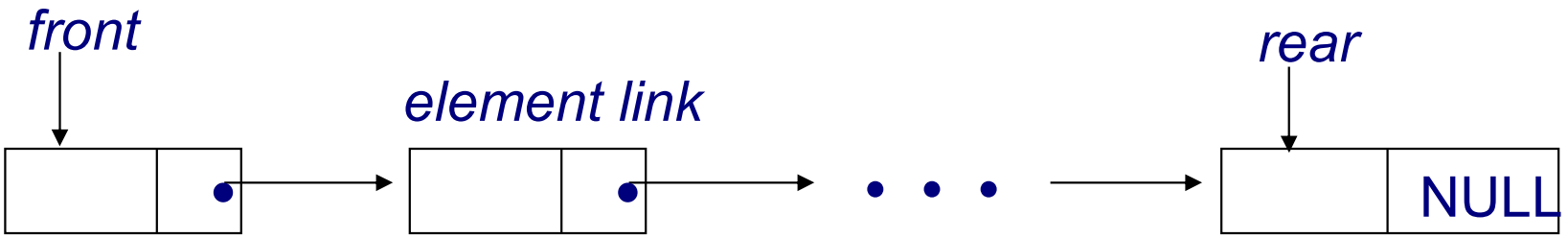
(1 + ((2 + 3) * (4 + 5)))


Stack Data Structure Overview
https://chat.openai.com/chat/72b1a25d-87dc-4e72-921b-b1fe33829e0c
90%
New chat
Stack Data Structure Overvie
Clear conversations
Dark mode
OpenAI Discord
Updates & FAQ
Log out
HA
give me some examples of using stack
Here are some examples of using a stack data structure:
1. Expression evaluation: Stacks are used to evaluate mathematical expressions such as infix, postfix, and prefix expressions.
2. Compiler design: Stacks are used in compilers to implement syntax parsing and symbol tables.
3. Memory management: The operating system uses a stack to keep track of function calls and manage memory allocation.
4. Algorithm design: Stacks are used in algorithms such as Depth-First Search (DFS) and the Tower of Hanoi puzzle.
5. Undo and redo functionality: Applications such as text editors and image editors use stacks to implement undo and redo operations.
6. Web browsers: Web browsers use stacks to implement the back button functionality. The current web page is pushed onto the stack every time a new page is loaded, and popping the stack goes back to the previous page.
7. Balancing symbols: Stacks are used to check if the symbols in an expression are balanced
arentheses in a mathematical expressio
Regenerate response
ChatGPT Jan 30 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

Linked Stacks and Queues

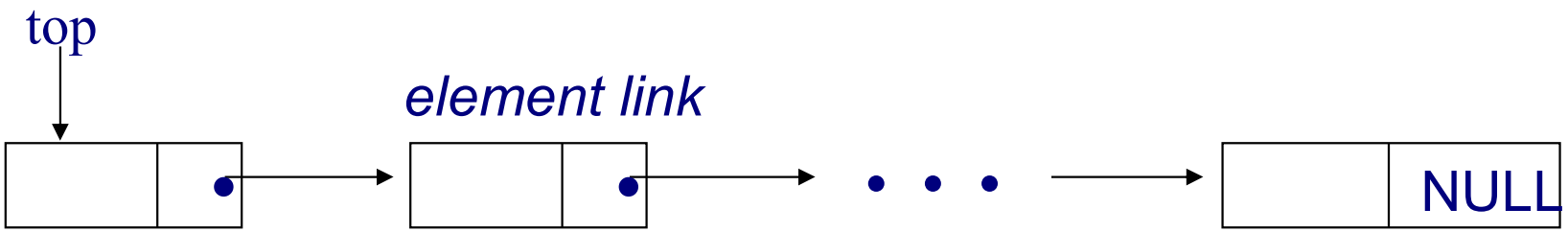


(a) Linked Stack

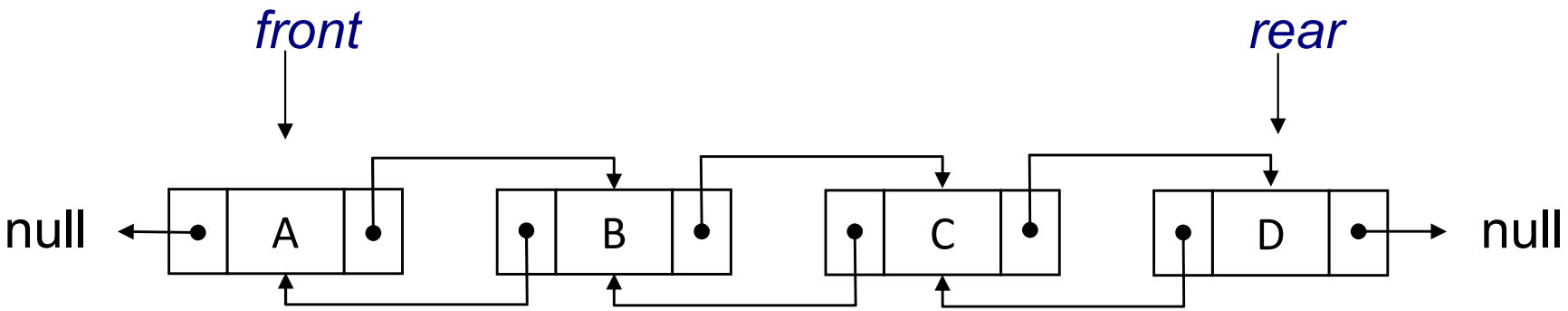


(b) Linked Queue

Linked Stacks and Queues



(a) Linked Stack



(b) Linked Queue

Bài tập/thực hành

- Cài đặt danh sách liên kết đơn, danh sách liên kết đôi
- Cài đặt ngăn xếp, hàng đợi dùng mảng và dùng danh sách liên kết
- Cài đặt ngăn xếp, hàng đợi cho kiểu dữ liệu bất kỳ (dùng template)
- Tìm hiểu các ứng dụng của stack, queue, linked list

Tự học

- Tìm hiểu về ký pháp tiền tố, hậu tố và giải thuật chuyển đổi biểu thức trung tố sang biểu thức tiền tố, hậu tố

Infix	Prefix	Postfix
x+y	+xy	xy+
x+y-z	-+xyz	xy+z –
x+y*z	+x*yz	xyz*+
x+(y-z)	+x-yz	xyz-+

Chuẩn bị, bài tập

- Hàng đợi ưu tiên
- Cây tìm kiếm nhị phân
- Áp dụng các thuật toán sắp xếp, tìm kiếm cho danh sách liên kết
- Tìm giải pháp truy vấn nhanh đối với danh sách liên kết