TRƯỜNG ĐẠI HỌC PHENIKAA
KHOA KHOA HỌC CƠ BẢN

□□□□□

BÀI TẬP LỚN KẾT THÚC HỌC PHẦN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT.1

Sinh viên : NGUYỄN VIẾT TUẤN

Lớp : K14-CNTT4

Mã SV : 20010932

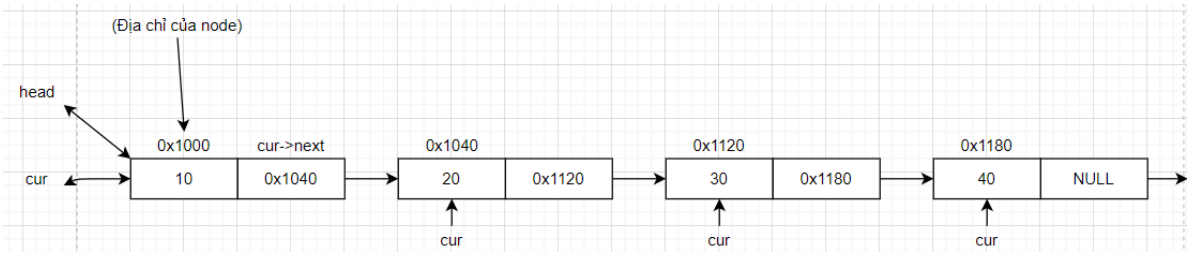
1.

1.1. Danh sách liên kết(LinkedList).

- Các thao tác cơ bản trong danh sách liên kết:
  - PrintList: Liệt kê toàn bộ phần tử trong danh sách
  - InsertNode: Chèn nút mới
  - RemoveNode: Xóa nút
  - FindNode: Tìm kiếm nút trong danh sách

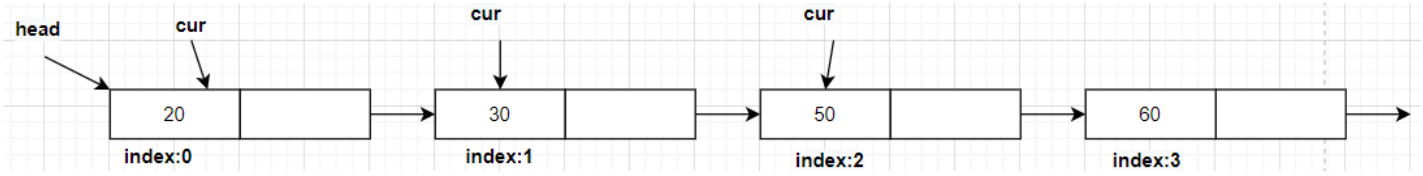
- Hàm PrintList:
  - Liệt kê tất cả các phần tử trong danh sách
  - liên kết bắt đầu từ phần tử đầu tiên và duyệt danh sách tới phần tử cuối cùng
  - Tại mỗi nút, sử dụng con trỏ next để di chuyển tới phần tử tiếp theo

Ví dụ: Liệt kê tất cả các phần tử trong danh sách liên kết



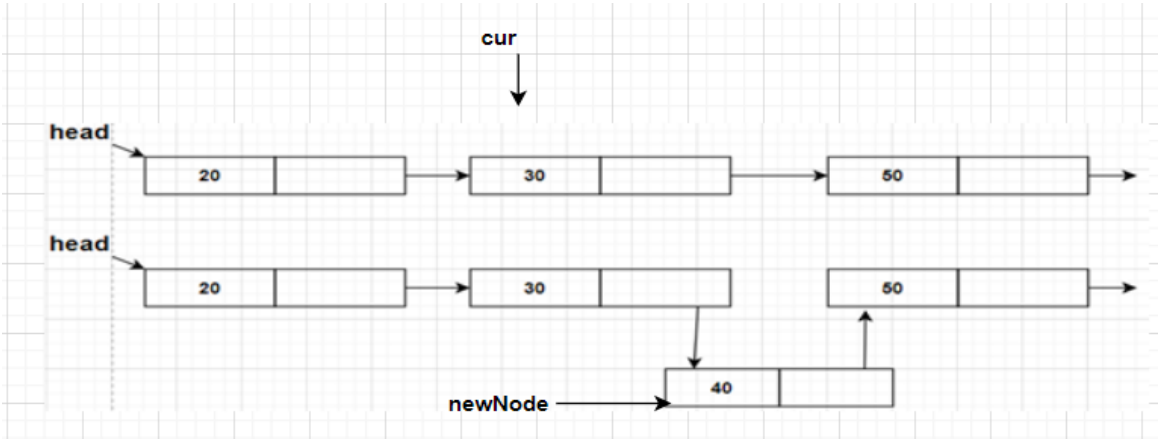
- Hàm findNode:

  VD: Tìm con trỏ chỉ tới nút có chỉ số i



- Hàm insertNode:
  - Chèn vào nút thứ I trong danh sách liên kết
  - Thêm nút này vào vị trí có chỉ số 2, ngay sau nút có chỉ số 1

VD: Thêm nút (40) vào giữa của danh sách liên kết không rỗng

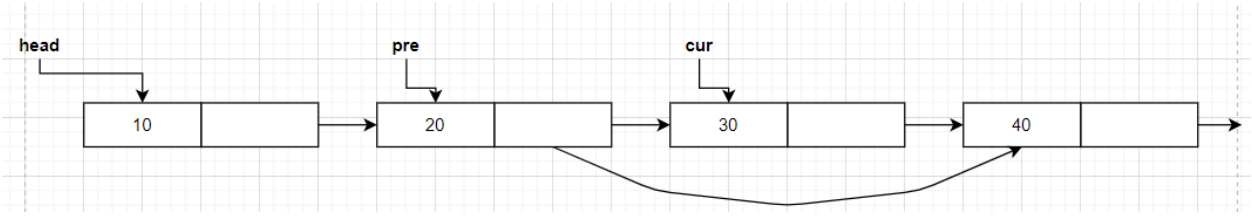


- Hàm RemoveNode:

  - Xóa nút thứ i trong danh sách liên kết

  - Tìm nút, loại bỏ

  VD:Xóa nút (30) ở giữa ra khỏi danh sách liên kết.



1.2. Hàng đợi (Queues)

• Các thao tác cơ bản trên hàng đợi:

  - Enqueue: Thêm vào một phần tử ở cuối hàng đợi

  - Dequeue: Lấy ra một phần tử ở đầu hàng đợi

  - Peek: Xem một phần tử ở đầu hàng đợi nhưng không lấy ra khỏi hàng đợi

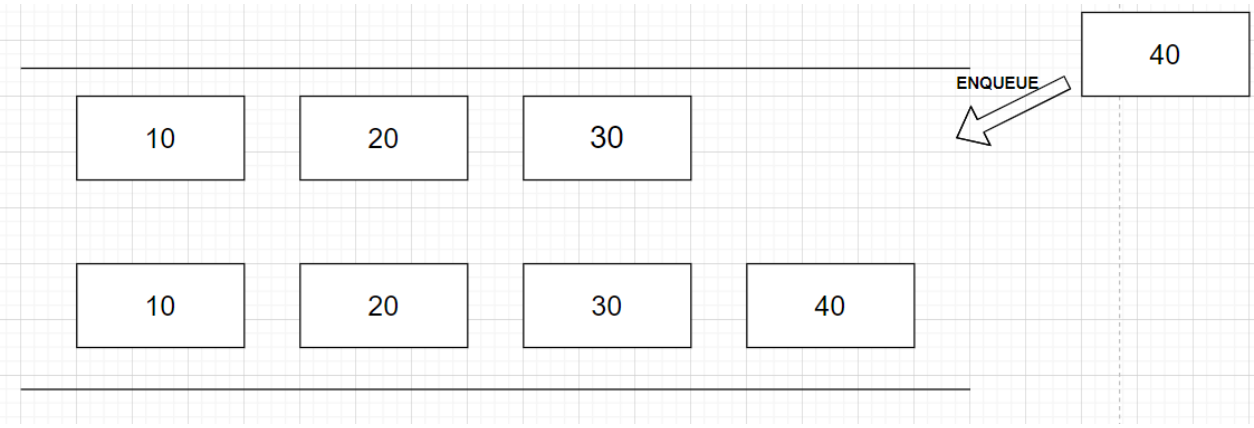
  - IsEmptyQueue: Kiểm tra xem hàng đợi có phần tử nào không

• Hàm Enqueue:

  - enqueue() là cách duy nhất để thêm một phần tử vào cấu trúc dữ liệu hàng đợi.

- enqueue() chỉ cho phép thêm một phần tử vào cuối hàng đợi
- Nút đầu tiên của danh sách liên kết là đầu của hàng đợi (hoặc cuối của hàng đợi thì thay đổi code).

VD: Thêm dữ liệu (40) vào cuối hàng đợi

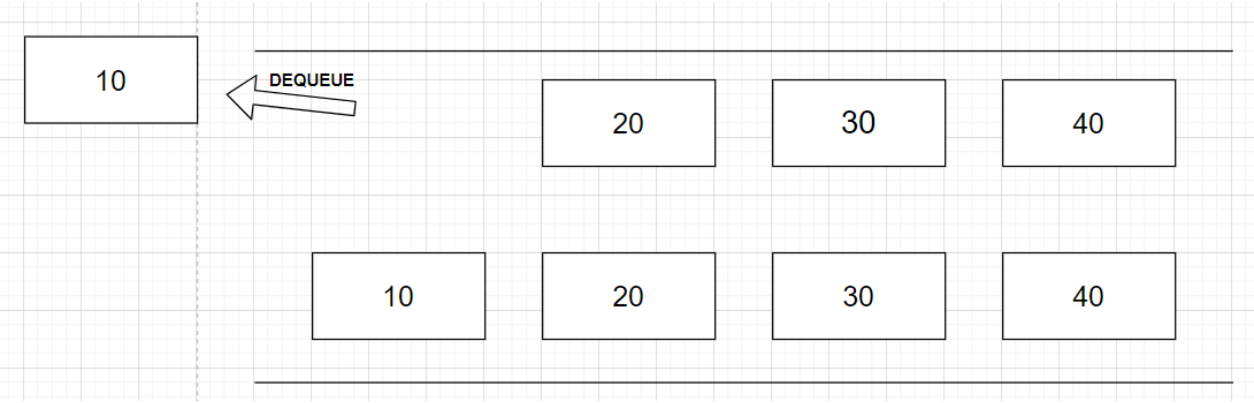


• Hàm Dequeue:

Thao tác lấy một phần tử ra khỏi hàng đợi gồm 2 bước

- Lấy giá trị của nút ở đầu hàng đợi
- Xóa nút đó trong danh sách liên kết

VD: Lấy dữ liệu đầu có giá trị (10) ra khỏi hàng đợi.



• Hàm Peek:

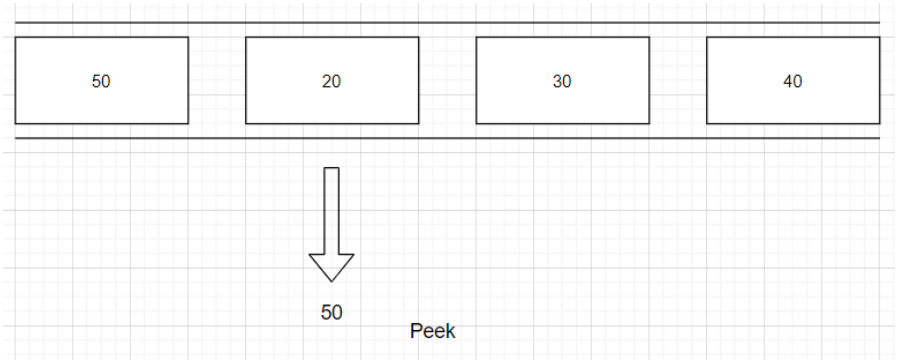
- Không tạo sự thay đổi gì trong hàng đợi

- Lấy giá trị của nút ở đầu hàng đợi:

+ Lấy giá trị của nút đầu tiên của danh sách liên kết

+ Không xóa nút này.

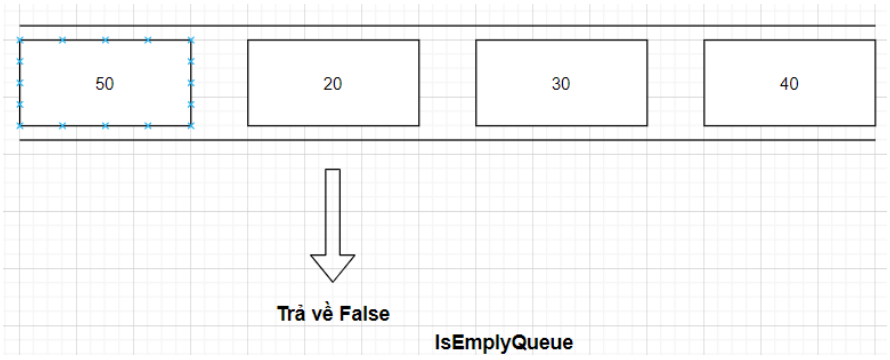
VD: Sử dụng Hàm Peek để lấy giá trị nút đầu tiên ra khỏi danh sách liên kết.



• - IsEmptyQueue:

- Kiểm tra xem số phần tử trong hàng đợi có phải bằng 0 không(rỗng)

Ví dụ: Trả về True nếu giá trị phần tử bên trong bằng 0 và False nếu #0



1.3. Ngăn Xếp (Stack)

• Các thao tác cơ bản trên ngăn xếp:

- Push: thêm phần tử vào trên cùng ngăn xếp.

- Pop: lấy phần tử nằm trên cùng ra khỏi ngăn xếp và trả về giá trị của phần tử đó (nếu ngăn xếp không rỗng).

- Peek (Lấy giá trị của phần tử trên cùng): Lấy ra giá trị của phần tử trên cùng mà không thực hiện xóa nó.

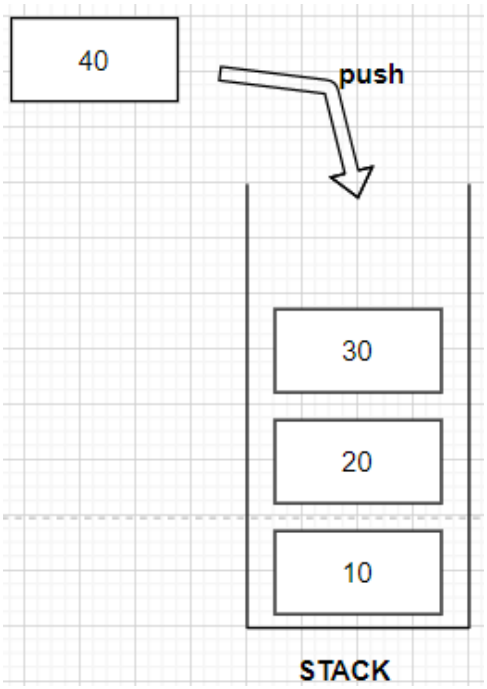
- IsEmptyStack (Kiểm tra rỗng hay không): Kiểm tra xem Stack có trống rỗng hay không.

• Hàm push:

- push là cách duy nhất để thêm một phần tử vào cấu trúc dữ liệu ngăn xếp

- push chỉ thao tác trên đỉnh của ngăn xếp.

VD:Đưa giá trị (40) vào đầu ngăn xếp



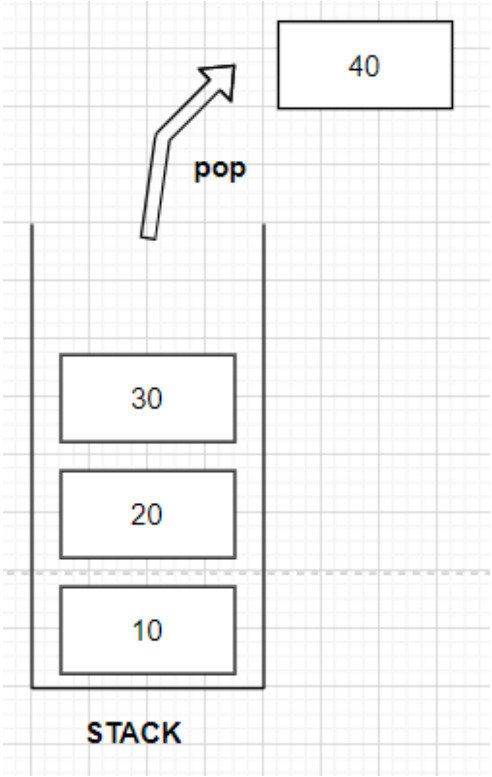
• Hàm pop:

Thao tác lấy một giá trị khỏi ngăn xếp gồm 2 bước:

- Lấy giá trị của nút ở đỉnh của danh sách liên kết

- Xóa nút này khỏi danh sách liên kết

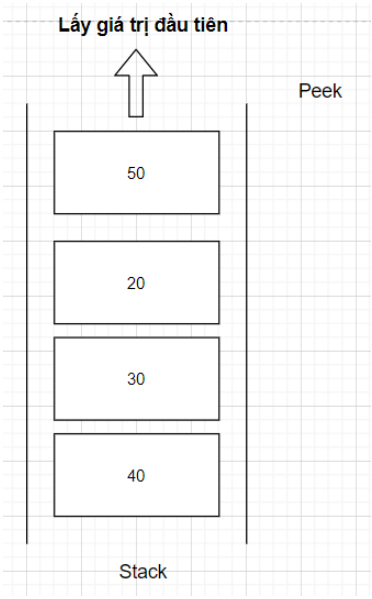
VD: Lấy dữ liệu có giá trị là (40) ra khỏi ngăn xếp



• **Hàm peek:**

- Lấy giá trị của nút ở đỉnh của danh sách liên kết
- Không xóa nút này khỏi danh sách liên kết.

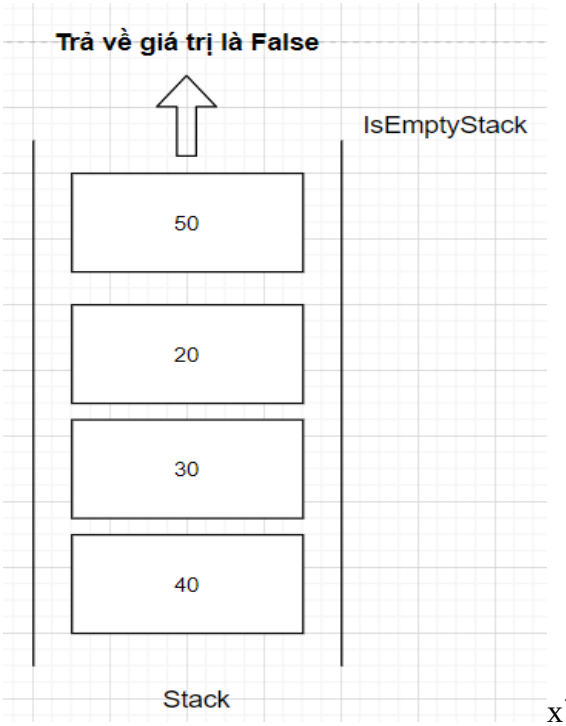
Ví dụ: Trả về giá trị của dữ liệu đầu ngăn xếp.



• **Hàm IsEmptyStack:**

- Kiểm tra xem số phần tử trong ngăn xếp có phải bằng 0 không.

VD: Trả về giá trị là True nếu số phần tử trong ngăn xếp có giá trị là 0 và False nếu số phần tử trong ngăn xếp #0

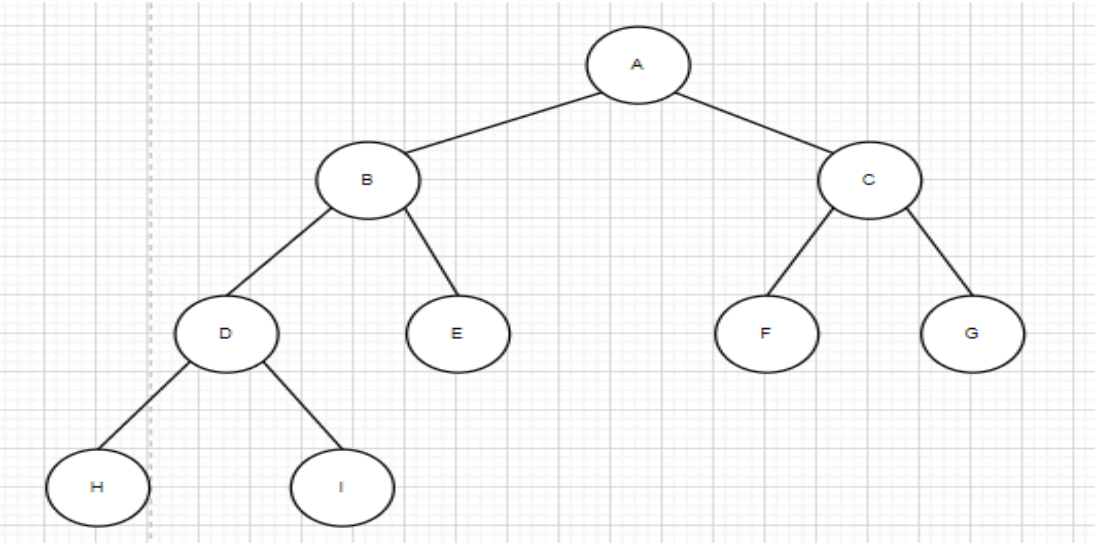


x`

1.4. Duyệt Cây Nhị Phân:

- Duyệt tiền tự: Duyệt cây sẽ thăm lần lượt các con trái rồi đến con phải sao cho chỉ đi qua nó đúng 1 lần.

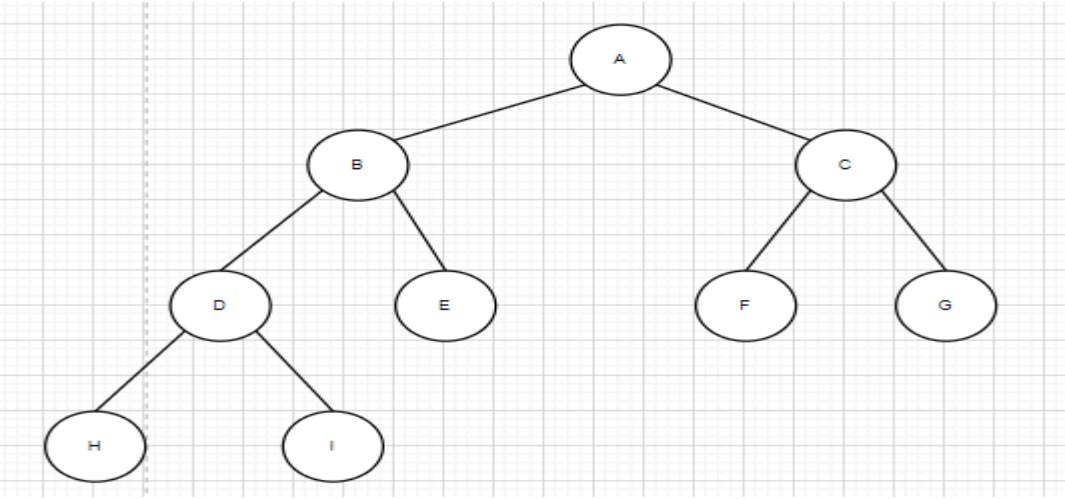
VD: Duyệt tiền tự:



Duyệt cây nhị phân trên ta có : A B D H I E C F G

- Duyệt trung tự: Duyệt cây sẽ thăm lần lượt các con trái rồi đến nút cha rồi đến con phải sao cho chỉ đi qua nó đúng 1 lần.

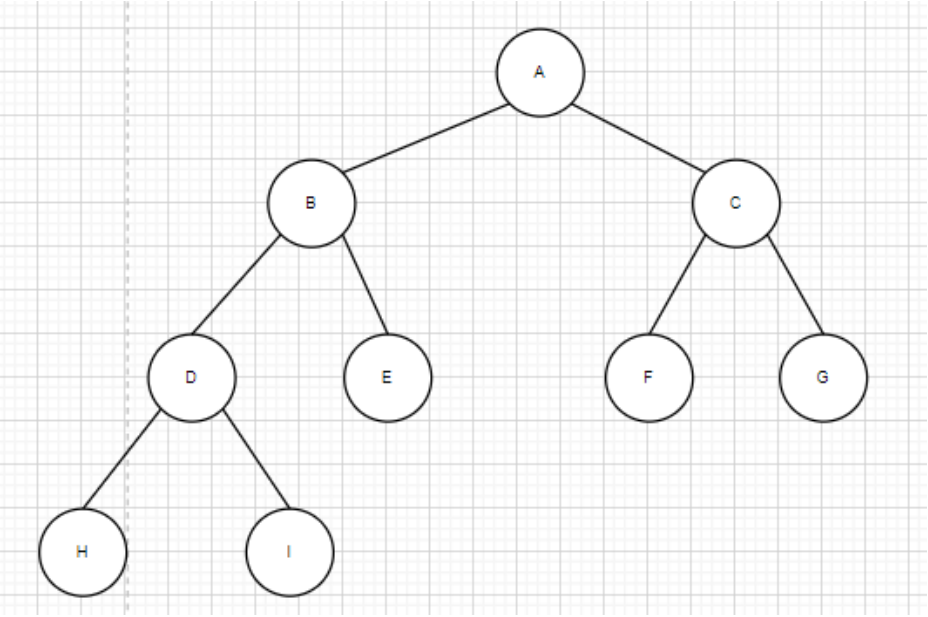
VD: Duyệt trung tự



Duyệt cây nhị phân trên ta có : H D I B E A F C G

- Duyệt Hậu Tự: Duyệt cây sẽ thăm lần lượt các con trái rồi đến con phải rồi xét nút cha sao cho chỉ đi qua nó đúng 1 lần.

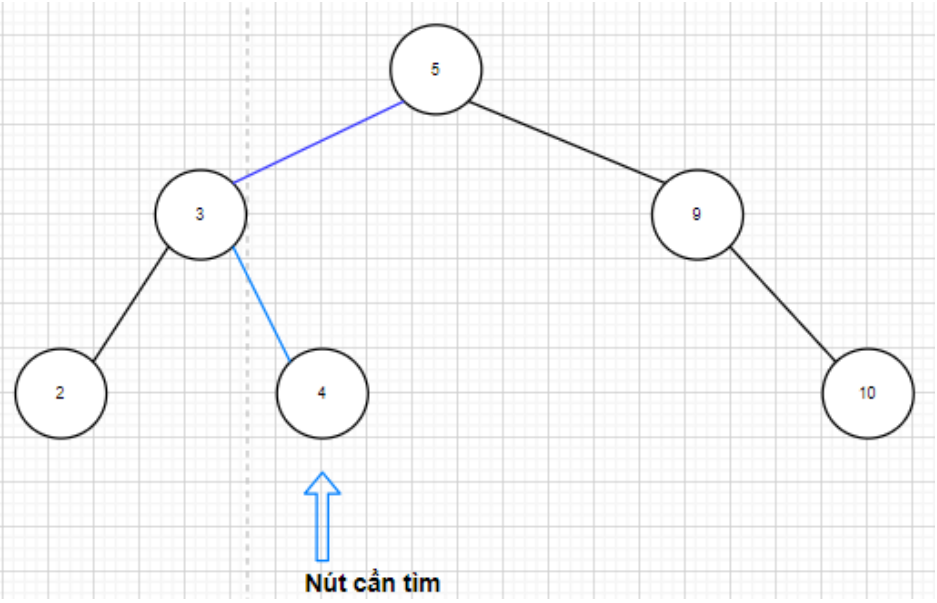
VD: Duyệt hậu tự



Duyệt cây nhị phân trên ta có : H I D E B F G C A

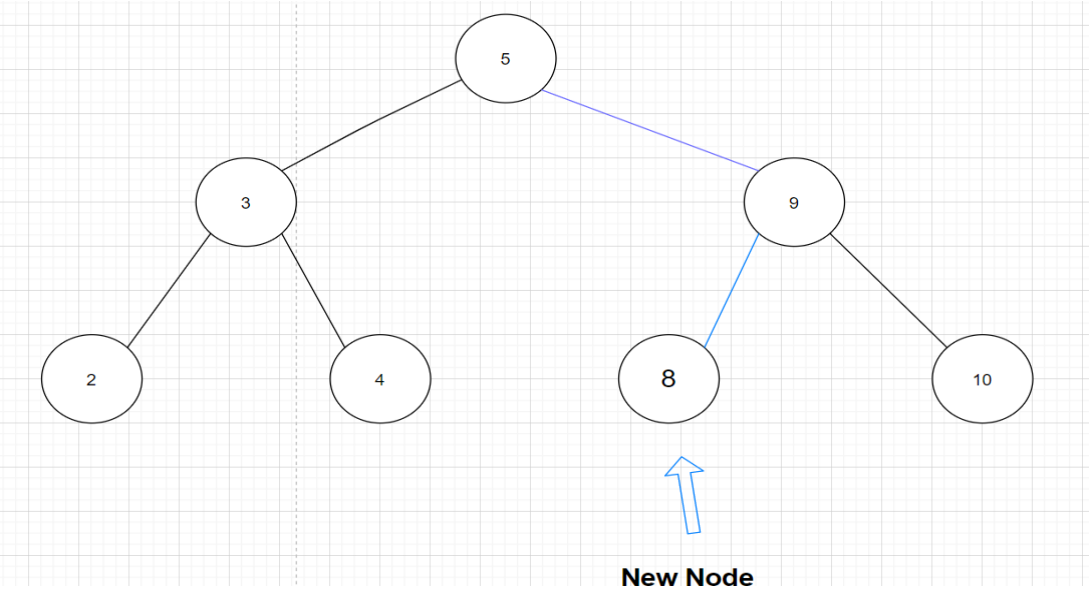
1.5 Tìm kiếm và chèn trên cây tìm kiếm nhị phân

VD: Tìm kiếm trên cây tìm kiếm nhị phân



Để tìm kiếm 1 phần tử trên cây nhị phân ta sử dụng cây nhị phân tìm kiếm như hình trên. Ví dụ ta muốn tìm nút có giá trị là 4 thì ta sẽ duyệt. So sánh ta thấy 4 < 5 (nút gốc) nên ta sẽ duyệt cây bên trái tiếp tục so sánh ta thấy 4 > 3 (nút vừa duyệt) nên ta duyệt sang bên phải tìm thấy nút số 3.

VD: Chèn trên cây tìm kiếm nhị phân



Để chèn 1 nút vào cây tìm kiếm nhị phân . Ví dụ chèn nút 8 vào cây nhị phân này .Đầu tiên ta so sánh nút 8 với nút gốc ta thấy 8 > 5 (nút gốc) suy ra ta sẽ duyệt bên trái . Tiếp tục ta thấy nút 8 < 9 (nút vừa duyệt) nên ta chèn nút 8 vào bên trái ta có nút cha là 9.

1.6. Giải thuật sắp xếp cơ bản

- Thuật toán Selection-sort: sắp xếp một mảng bằng cách liên tục tìm phần tử nhỏ (xét theo thứ tự tăng dần) từ phần không được sắp xếp và đặt nó ở đầu.

i	K(i)	Lượt 1	Lượt 2	Lượt 3	Lượt 4	Lượt 5	Kết quả
1	37	9	9	9	9	9	9
2	11	11	11	11	11	11	11
3	9	37	24	24	24	24	24
4	65	65	65	37	37	37	37
5	24	24	37	65	57	57	57
6	57	57	57	57	65	65	65

- Thuật toán Insert-sort: thể so sánh phần tử mới lần lượt với phần tử thứ (i-1), thứ (i-2)... để tìm ra "chỗ" thích hợp và "chèn" nó vào chỗ đó

Lượt	1	2	3	4	5	6	Kết quả
K(i)	37	11	9	65	24	57	
1	37	11	9	9	9	9	9
2		37	11	11	11	11	11
3			37	37	24	24	24
4				65	37	37	37
5					65	57	57
6						65	65

-Thuật toán bubble-sort: • Bảng các khoá sẽ được duyệt từ đáy lên đỉnh. Dọc đường, nếu gặp hai khoá kế cận ngược thứ tự thì đổi chỗ chúng cho nhau.

• Như vậy trong lượt đầu khoá có giá trị nhỏ nhất sẽ chuyển dần lên đỉnh. Đến lượt thứ hai khoá có giá trị nhỏ thứ hai sẽ được chuyển lên vị trí thứ hai...

i	K(i)	Lượt 1	Lượt 2	Lượt 3	Kết quả
1	37	9	9	9	9
2	11	37	11	11	11
3	9	11	37	24	24
4	65	24	24	37	37
5	24	65	57	57	57
6	57	57	65	65	65

1.7. Giải thuật sắp xếp Quicksort

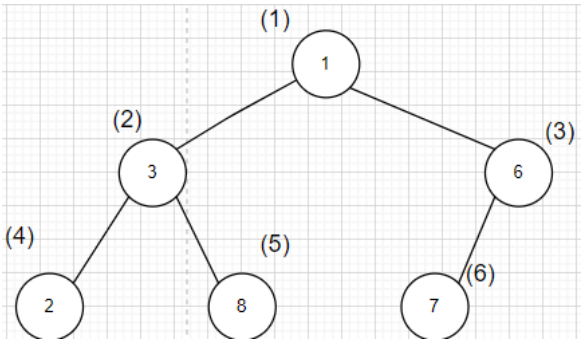
Lượt	K(i)	37	11 i->	9	65 i	24 j	57 <-j
1		37	11	9	24 <-j j	65 i-> i	57

2		(24 i->	11	9) <-j j	37 i	(65	57) i-> <-j j i
3		(9 j	11) i-> <-j i	24	37	(57)	65
4		9	(11)	24	37	57	65
Kết quả		9	11	24	37	57	65

1.8 .Thuật toán sắp xếp heapsort

* Bước tạo đống K(i)

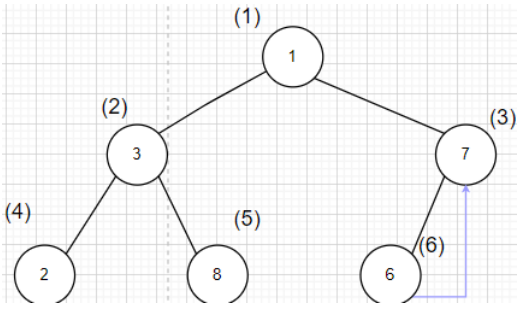
1	3	6	2	8	7



ADJUST(3,6)

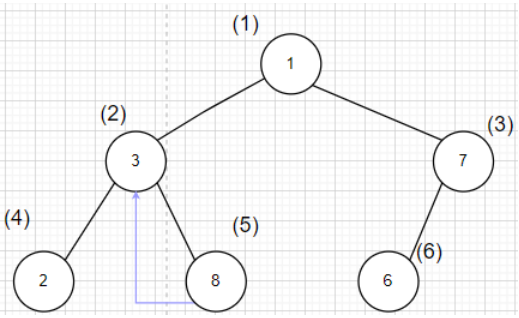
1	3	7	2	8	6

K(i)



ADJUST(2,6) K(i)

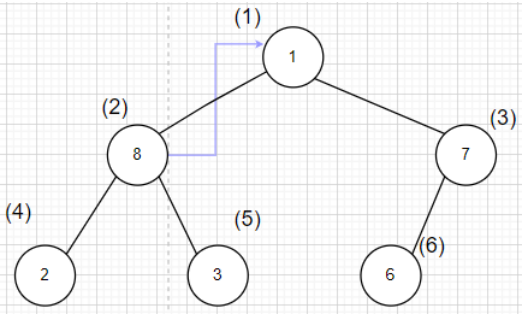
1	3	7	2	8	6

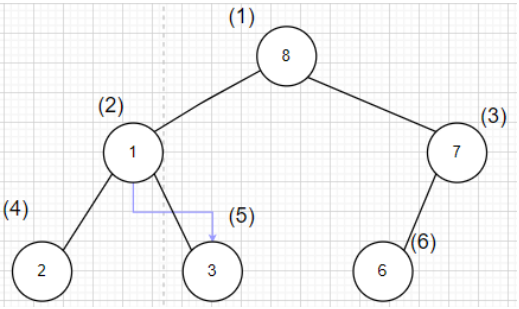


ADJUST(1,6)

1	8	7	2	3	6

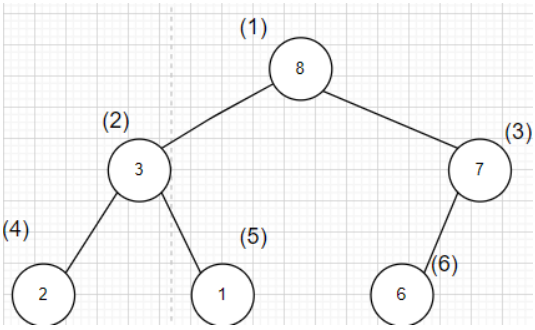
K(i)





K(i)

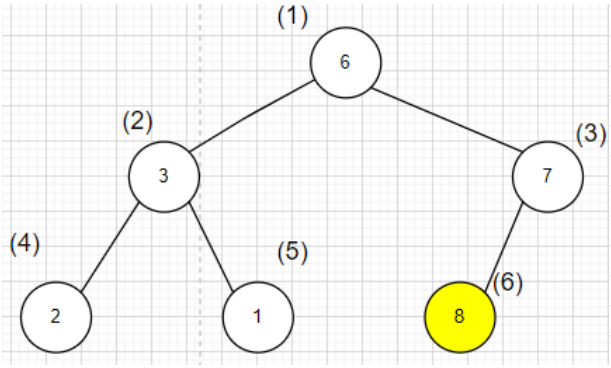
8	1	7	2	3	6

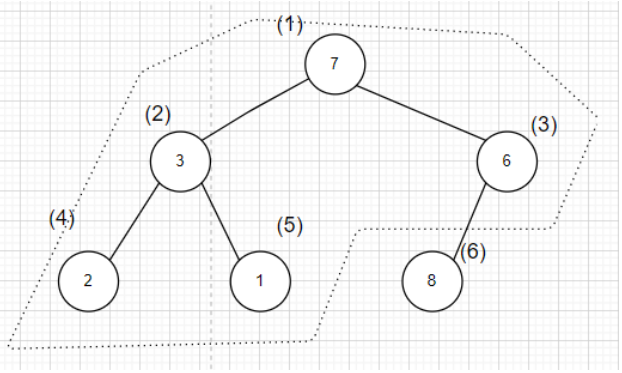


Đống đã được tạo

* Bước vun đống:

Sau khi đổi chỗ và thực hiện ADJUST (1,5)

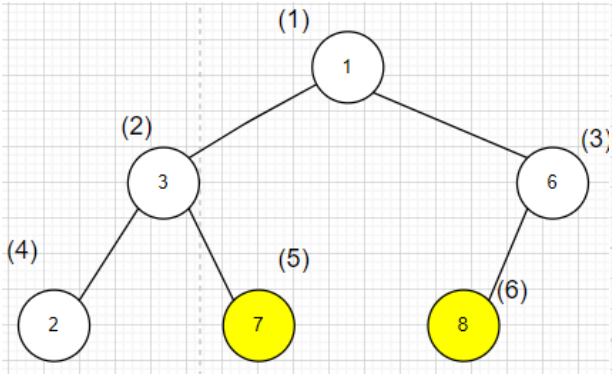


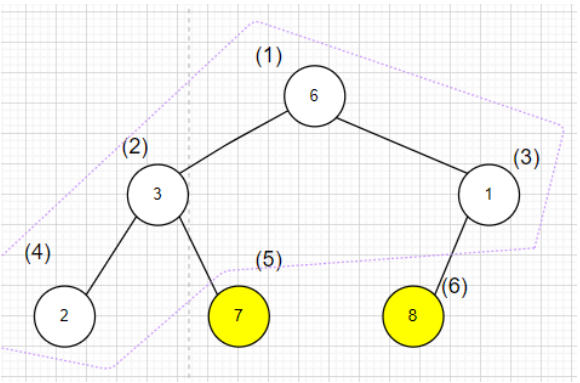


7	3	6	2	1	8

K(i)

Sau khi đổi chỗ và thực hiện ADJUST(1,4)

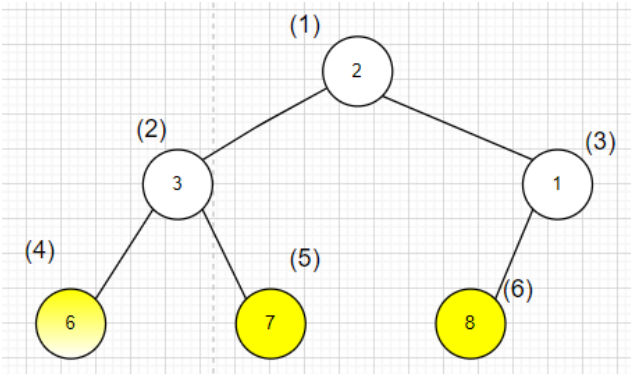


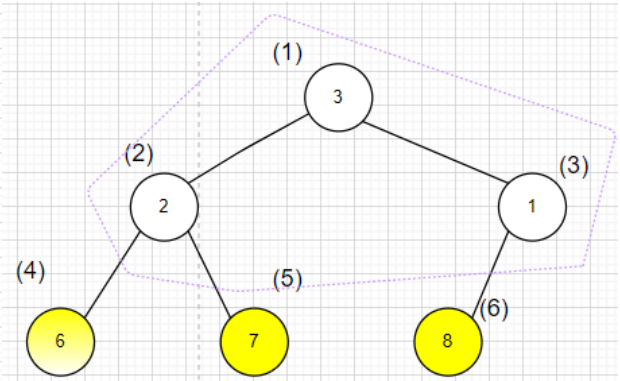


K(i)

6	3	1	2	7	8

Sau khi đổi chỗ và thực hiện ADJUST(1,3)

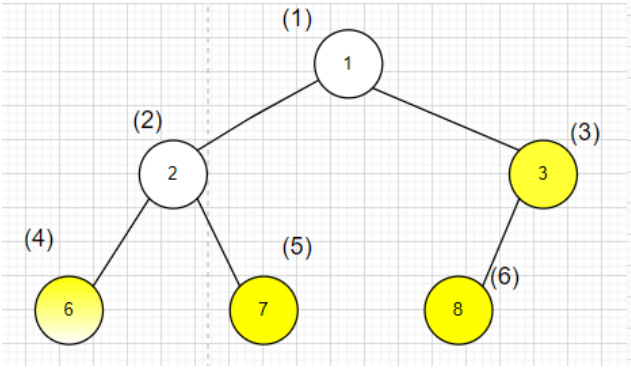


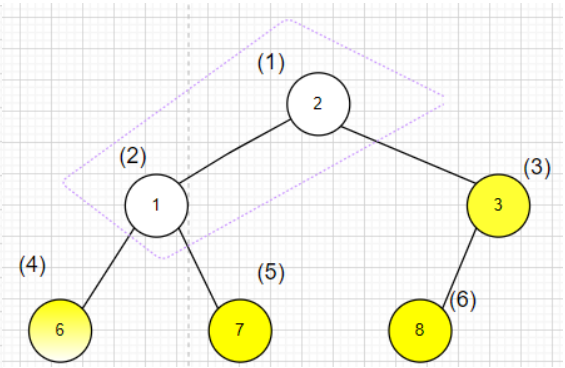


K(i)

3	2	1	6	7	8

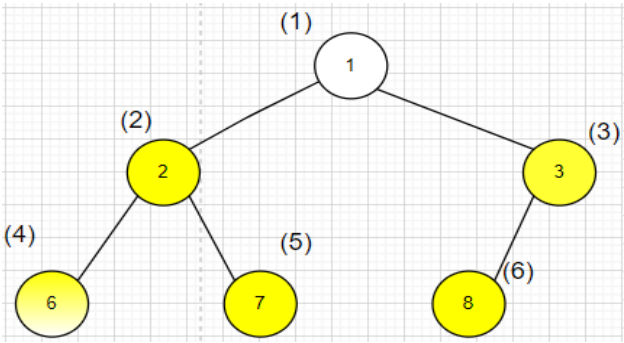
Sau khi đổi chỗ và thực hiện ADJUSC(1,2)

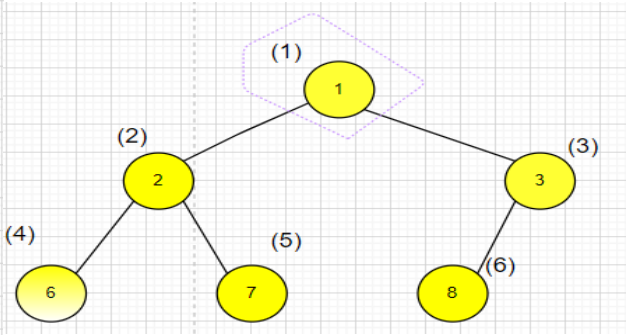




K(i)	2	1	3	6	7	8

Sau khi đổi chỗ và thực hiện ADJUST(1, 1)

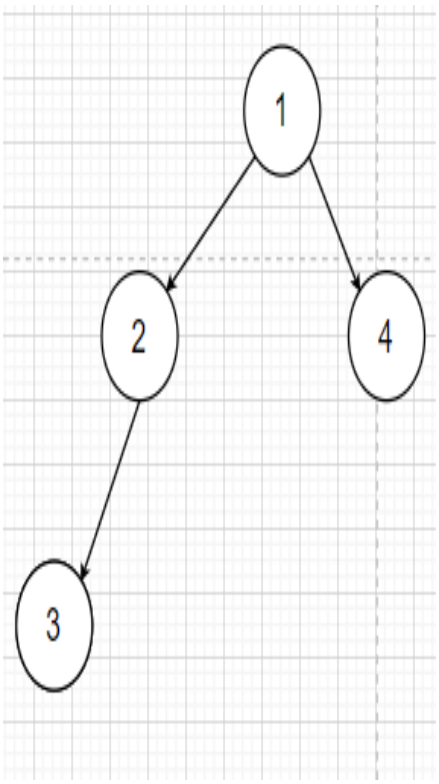




K(i)	1	2	3	6	7	8

1.9. Duyệt đồ thị theo chiều sâu và rộng

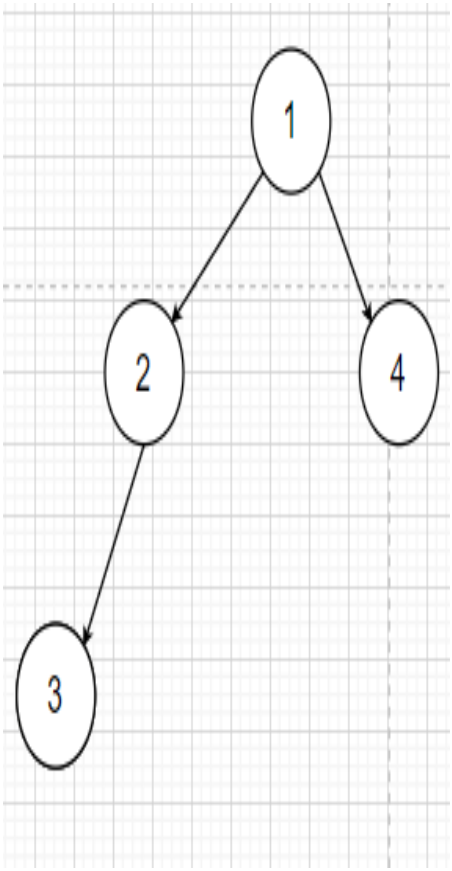
+Duyệt theo chiều rộng:



Ta duyệt như sau:

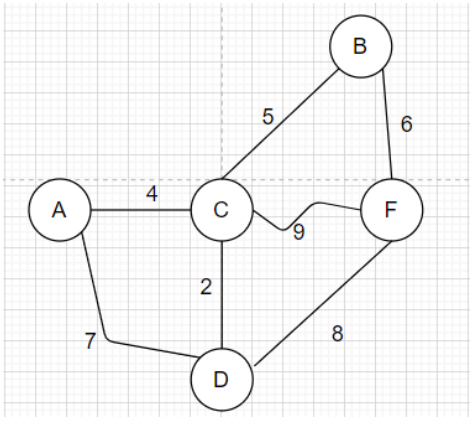
Lần 1: - Hàng đợi 1
Cur: trống
Kết quả duyệt: trống
Lần 2: - Hàng đợi 1
Cur: 1
Kết quả duyệt: 1
Lần 3: - Hàng đợi 2
Cur:1
Kết quả duyệt: 1
Lần 4: - Hàng đợi : 2 – 4
Cur: 1
Kết quả duyệt: 1
Lần 5: - Hàng đợi: 4
Cur: 2
Kết quả duyệt: 1 – 2
Lần 6: - Hàng đợi: 4 – 3
Cur :2
Kết quả duyệt 1 – 2
Lần 7: Hàng đợi: 3
Cur : 4
Kết quả duyệt: 1 -2- 4
Lần 8: Hàng đợi : Trống
Cur: 3
Kết quả duyệt: 1 – 2 – 4 – 3

+Duyệt theo chiều sâu:



Ta duyệt như sau:
Lần 1: - Hàng đợi 1
Cur: trống
Kết quả duyệt: trống
Lần 2: - Hàng đợi 1
Cur: 1
Kết quả duyệt: 1
Lần 3: - Hàng đợi 2
Cur:1
Kết quả duyệt: 1
Lần 4: - Hàng đợi : 2 – 3
Cur: 1
Kết quả duyệt: 1
Lần 5: - Hàng đợi: 3
Cur: 2
Kết quả duyệt: 1 – 2
Lần 6: - Hàng đợi: 3 – 4
Cur :2
Kết quả duyệt 1 – 2
Lần 7: Hàng đợi: 4
Cur : 3
Kết quả duyệt: 1 - 2 - 3
Lần 8: Hàng đợi : Trống
Cur: 4
Kết quả duyệt: 1 – 2 – 3 – 4

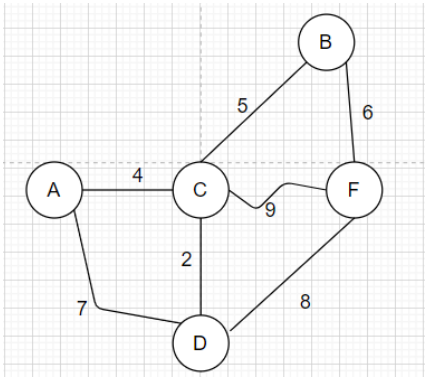
1.10. Tìm đường đi ngắn nhất bằng thuật toán Dijsktra



Bước	N'	D(D) p(D)	D(C) p(C)	D(B) P(B)	D(F) p(F)
0	A	7,A	4,A	Vô cùng	Vô cùng
1	AC	6,C		9,C	13,C
2	ACD			9,C	13,C
3	ACDB				13,C
4	ACDBF				

Vậy đường đi ngắn nhất từ : A -> C là 4
-> D là 6 theo đường A->C->D
-> B là 9 theo đường A->C->B
-> F là 13 theo đường A->C->F

1.11. Tìm đường đi ngắn nhất bằng thuật toán Bellman-Ford



	A	B	C	D	F
1	0	Vô cùng	4	7	Vô cùng
2	0	9	4	6	13
3	0	9	4	6	13

Vậy theo thuật toán Bellman-Ford ta thấy:

Đường đi ngắn nhất từ : A->B là 9 Theo đường A->C->B

A->C là 4 Theo đường A->C

A->D là 6 Theo đường A->C->D

A->F là 13 theo đường A->C->F