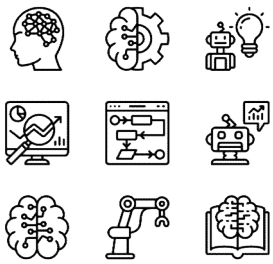
Computer Science for Practicing Engineers

Stack và Queue

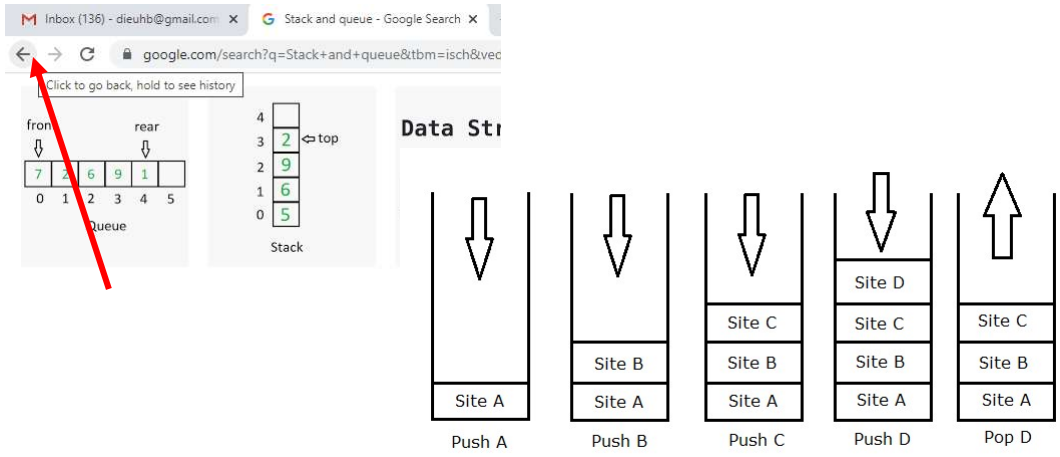


TS. Huỳnh Bá Diệu
Email: dieuhb@gmail.com
Phone: 0914146868

Nội dung

1. Ngăn xếp
2. Hàng đợi và hàng đợi ưu tiên
3. Các ứng dụng của ngăn xếp và hàng đợi

Ngăn xếp



Ngăn xếp (Stack) là cấu trúc lưu trữ, trong đó:

- Các phần tử cùng kiểu
- Số phần tử không cố định
- Các phần tử được đưa vào và lấy ra ở đỉnh ngăn xếp
- Ngăn xếp hoạt động theo cơ chế LIFO (Last In First Out)

Ví dụ:

- Chồng đĩa khi rửa + Chồng đĩa khi đưa ra phục vụ

Ngăn xếp (Stack)

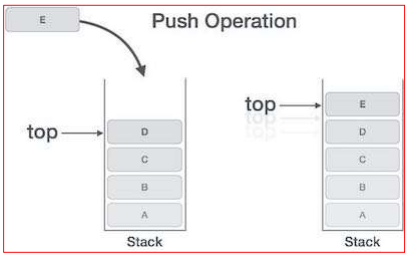
Có 4 thao tác cơ bản trên Stack

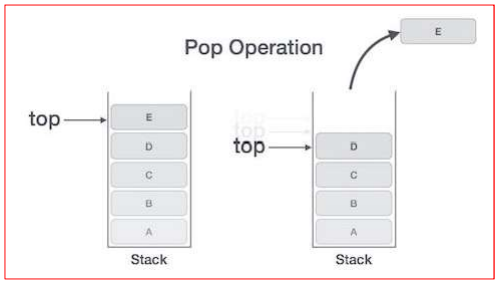
Tạo mới ngăn xếp;

Kiểm tra ngăn xếp có rỗng không;

Thêm 1 phần tử vào đỉnh ngăn xếp;

Lấy 1 phần tử ở đỉnh ngăn xếp ra.





Ngăn xếp (Stack)

Có thể tự định nghĩa kiểu Stack hoặc dùng kiểu có sẵn trong ngôn ngữ lập trình.

import java.util.Stack; Stack <String> S; **S= new Stack<>()**;

SN	Methods with Description
1	**boolean empty()**
2	***Object peek( )***
3	**Object pop( )**
4	**Object push(Object element)**
5	***int search(Object element)***

Ngăn xếp (Stack)

```
import java.util.Stack;
public class CSPE_MyP12_Using_StackType {
   Stack <String> S;
   void them()    {  // hãy viết code của bạn ở đây???                }

     public static void main(String[] args) {
     CSPE_MyP12_Using_StackType m= new CSPE_MyP12_Using_StackType();
      m.them();
   }
}
```

Ngăn xếp (Stack)

```
void them()
{
        S= new Stack <>();
        for(int i =1; i<=10; i++)     S.push(" Phan tu thu " + i);
        System.out.println("\n Noi dung Stack: \n");
        while (!S.empty()) { String x= S.pop();   System.out.println(" --" + x); }
        System.out.println("\n Noi dung Stack: "+ S);
    }
```

Khai báo Ngăn xếp (Stack)

```
class Stack{
        struct Node { int data; Node *next;};
        Node *top;
        public:
                CreateS() {  top=NULL;}
                bool EmptyS(){ return top==NULL;}
                int PopS() {}
                void PushS(int x) {}
};
```

Ngăn xếp (Stack)

```
int PopS()   {
   int x=0;
   if(top!=NULL){ Node *t=top; x=top->data; top=top->next; delete t; }
   else cout<<"\n Ngan xep rong!!!";
   return x;
}
void PushS(int x) {
  Node *t= new Node; t->data=x; t->next=top; top= t;
}
```

Ngăn xếp

```
3   class Stack
4   {
5       struct Node { int data; Node *next;};
6       Node *top;
7       public:
8           CreateS() { top=NULL;}
9           bool EmptyS(){ if(top==NULL) return true; else return false;}
10          int PopS()
11          {
12              int x=0;
13              if(top!=NULL) { Node *t=top; x=top->data; top=top->next; delete t;  }
14              else cout<<"\n Ngan xep rong!!!";
15              return x;
16          }
17          void PushS(int x) {
18              Node *t= new Node; t->data=x; t->next=top; top= t;
19          }
20  };
21  int main()
22  {
23      Stack S; S.CreateS();
24      S.PushS(5);S.PushS(6);  S.PushS(7); S.PushS(8); S.PushS(9);
25      while(!S.EmptyS()) cout<<S.PopS()<<"  ";
26  }
```

Ngăn xếp

```
18              Node *t= new Node; t->data=x; t->next=top; top= t;
19          }
20          void nhap()
21          {
22              CreateS();
23              int x;
24              while(1)
25              {
26                  cout<<" nhap so (<>0)de them vao ngan xep:"; cin>>x;
27                  if(x==0) break;
28                  PushS(x);
29              }
30
31          }
32  };
33  int main()
34  {
35      Stack S; S.CreateS();
36      //S.PushS(5);S.PushS(6);     S.PushS(7); S.PushS(8); S.PushS(9);
37      S.nhap();
38      while(!S.EmptyS()) cout<<S.PopS()<<"  ";
39  }
40
```

Ứng dụng của ngăn xếp

```
31          }
32          void in()
33          {
34              Stack t; t.CreateS();
35              cout<<"\n Noi dung ngan xep: ";
36              while(!EmptyS()) { int x= PopS(); cout<<x<<"  "; t.PushS(x);}
37              while(!t.EmptyS()) PushS(t.PopS());
38          }
39  };
40  int main()
41  {
42      Stack S; S.CreateS();
43      //S.PushS(5);S.PushS(6);     S.PushS(7); S.PushS(8); S.PushS(9);
44      S.nhap(); |  S.in();
45  }
46
```

Ứng dụng của ngăn xếp

Dùng kiểu dữ liệu Stack, viết chương trình kiểm tra chuỗi T chứa các ký tự "(", ")" có tạo thành chuỗi hợp lệ không.

()

()()

(()())

()(()())

)(

((((

())(

Giải pháp của bạn???

Ứng dụng của ngăn xếp

Dùng kiểu dữ liệu Stack, viết chương trình kiểm tra chuỗi T chứa các ký tự "(", ")" có tạo thành chuỗi hợp lệ không.

Thuật toán:

1. Tạo ngăn xếp S rỗng
2. Duyệt từng phần tử x của chuỗi T
   - Nếu x là dấu (thì cho vào ngăn xếp S
   - Ngược lại Nếu x là dấu) thì kiểm tra ngăn xếp có rỗng không. Nếu rỗng thì trả về false ngược lại lấy ra 1 phần tử từ S
3. Nếu ngăn xếp rỗng thì trả về true ngược lại trả về false

Ứng dụng của ngăn xếp

```
#include <stack>
bool kiemtra(string t) {
   stack <char> S;
   for(int i=0; i<t.length(); i++)
       if(t[i]=='(') S.push(t[i]);
        else if(t[i]==')')
                  if(!S.empty()) S.pop();  else return false;
              else { cout<<"\n chuoi chua ky tu khong hop le!!!";return false; }
    if(S.empty()) return true; else return false;
}
```

Ứng dụng của ngăn xếp

```
int main()
{
        string t;
        cout<<"nhap chuoi can kiem tra:"; cin>>t;
        if(kiemtra(t)) cout<<"\n chuoi ngoac nhap la chuoi hop le";
        else cout<<"\n chuoi ngoac nhap la chuoi KHONG hop le";

}
```

Find-duplicate-parenthesis-expression

Cho chuỗi S biểu diễn một biểu thức. Kiểm tra xem chuỗi S có chứa cặp ngoặc thừa hay không?

Ví dụ:

(a+b)*(c-a): không có

(a+b) + z: không có

((a+b))*(c-a): có

Giải pháp của bạn???

Find-duplicate-parenthesis-expression

Chúng ta dùng stack để kiểm tra. Ý tưởng là xét từng ký tự của biểu thức:

- If the current character in the expression is a not a closing parenthesis `')'`, then we push the character into the stack.

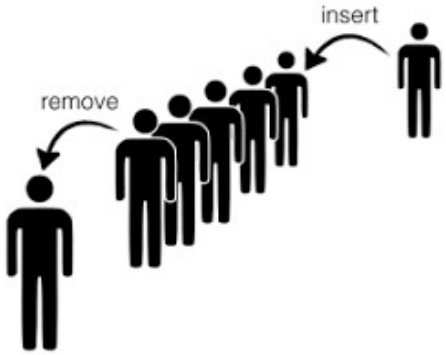
- If the current character in the expression is a closing parenthesis `')'`, we check if the topmost element in the stack is an opening parenthesis or not. If it is opening parenthesis, then the sub-expression ending at current character is of the form `((exp))` else we continue popping characters from the stack till matching `'('` is found for current `')'`.

Ứng dụng ngăn xếp

Khử đệ qui bài toán tháp Hà Nội

- **Cách lưu trữ 1 phần tử trong ngăn xếp???**
- **Giải thuật để khử đệ qui như thế nào???**

Hàng đợi (Queue)



Hàng đợi (Queue)

- Các phần tử cùng kiểu
- Số phần tử không cố định
- Các phần tử được đưa vào ở đuôi hàng đợi và lấy ra ở đầu hàng đợi
- Hàng đợi hoạt động theo cơ chế FIFO (First In First Out)

Ví dụ:

- Xếp hàng vô thang máy
- Máy in trong mạng

Các thao tác Cơ bản trên Hàng đợi (Queue)

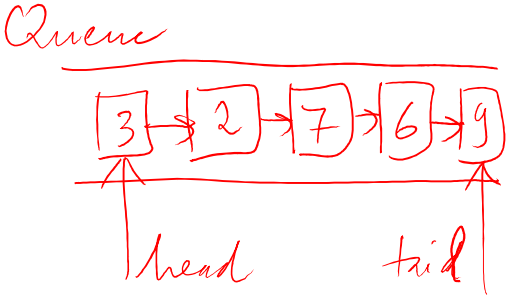
Có bốn thao tác chính:

- Tạo hàng đợi rỗng
- Kiểm tra hàng đợi có rỗng hay không
- Thêm 1 phần tử có giá trị x vào đuôi hàng đợi
- Lấy 1 phần tử ở đầu hàng đợi ra

[kiểm tra hàng đợi đã đầy chưa]

Có thể dùng mảng hoặc danh sách liên kết để cài đặt hàng đợi.

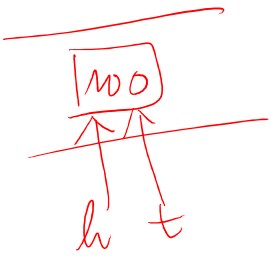
Các thao tác Cơ bản trên Hàng đợi (Queue)

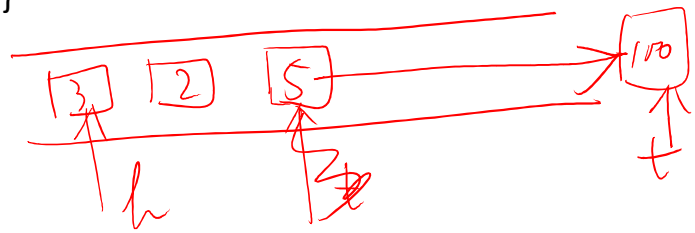



```
class Queue{
    struct Node { int data; Node *next;};
    Node *head, *tail;
   public:
        void CreateQ() {head=tail=NULL;}
        bool EmptyQ(){ if(head==NULL) return true; else return false;}
        void AddQ(int x) { }
        int RemoveQ() { }
}
```

Các thao tác Cơ bản trên Hàng đợi (Queue)

```
void AddQ(int x)
{
    Node *t= new Node; t->data=x; t->next=NULL;
    if(head==NULL) head= tail= t;
    else {tail->next=t; tail=t;}
}
```

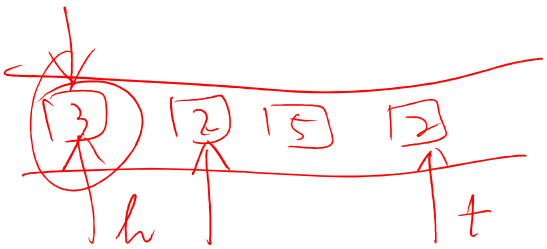




Các thao tác Cơ bản trên Hàng đợi (Queue)

```
int RemoveQ() {
        int x=0;
        if(head==NULL) cout<<"\n Hang doi rong!!!";
        else{
                x=head->data;  Node *t=head;
                if(head->next==NULL) head= tail=NULL; else head=head->next;
                delete t;
        }
        return x;
}
```

x=3



Các thao tác Cơ bản trên Hàng đợi (Queue)

```
int main()
{
        Queue Q; Q.CreateQ();
        Q.nhap();   Q.in();
}
```

Các thao tác Cơ bản trên Hàng đợi (Queue)

```
void nhap() {
        CreateQ();
        int x;
        while(1){
           cout<<" nhap so (<>0)de them vao hang doi:"; cin>>x;
           if(x==0) break;
           AddQ(x);
        }
}
```

Các thao tác Cơ bản trên Hàng đợi (Queue)

```
void in(){
        Queue t; t.CreateQ();
        cout<<"\n Noi dung hang doi: ";
        while(!EmptyQ()) { int x= RemoveQ(); cout<<x<<"  "; t.AddQ(x);}
        while(!t.EmptyQ()) AddQ(t.RemoveQ());
}
```

Cài đặt Hàng đợi (Queue)

```
public class MyQueue
{
        Node head, tail;
        MyQueue() { head= tail= null;}
        boolean EmptyQ()
                {return head==null;}
        void AddQ(int x)
        {
                Node t= new Node (x);
                if(head== null) head= tail=t;
                else { tail.next = t; tail = t;}
        }
```

```
        int RemoveQ()
        {
                int x= 0;
                if(head==null)
                System.out.println("Hang doi rong");
                else
                {
                   x= head.data;
                     if(head==tail) head=tail=null;
                    else   head= head.next;
                  return x;
                }
        }
```

Cài đặt Hàng đợi (Queue)

```
public static void main( String argvs)
{
        MyQueue Q= new MyQueue();
        for(int i=1; i<=10; i++) Q.AddQ(i);
        System.out.println("\n cac so trong hang doi:");
        while (!Q.EmptyQ()) System.out.println(Q.RemoveQ() + " ");
}
```

Lưu tạm Hàng đợi

```
int getk(int k)   {
       int x=0;  MyQueue t= new MyQueue();
       int vt=1;
       // lay k-1 phan tu o Q bo sang t
       while(!EmptyQ() && vt<k) { t.AddQ(RemoveQ()); vt++; }
        if(vt==k) x= RemoveQ(); // lay phan tu thu k neu nhu hang doi con
        else System.out.println("\n Hang doi khong du " + k+ " phan tu!");
       while(!EmptyQ() ) t.AddQ(RemoveQ()); // lay cac phan tu con lai trong Q bo vao t
       while(!t.EmptyQ() ) AddQ(t.RemoveQ());   // Lay cac phan tu tu t bo lai hang doi Q
      return x;
    }
```

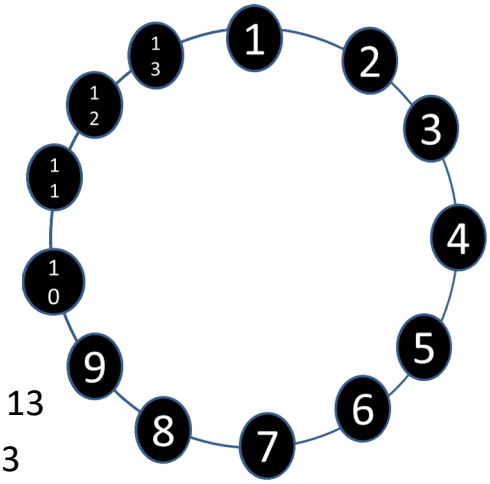
Sử dụng Hàng đợi của Java

Type of Operation	Throws exception	Returns special value
Insert	`add(e)`	`offer(e)`
Remove	`remove()`	`poll()`
Examine	`element()`	`peek()`

Sử dụng Hàng đợi của Java

```
import java.util.*;
public class Dung_Queue {
public static void main(String[] args) throws InterruptedException {
    int time = 30;
    Queue<Integer> Q = new LinkedList<Integer>();
    for (int i = time; i >= 0; i--) queue.add(i);
    while (!Q.isEmpty())
      { System.out.println(Q.remove()); Thread.sleep(1000); }
}
}
```

Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]



N= 13

K= 3

Có n người xếp thành vòng tròn, đánh số từ 1 đến n. Cần chọn 1 người làm vua trong n người.

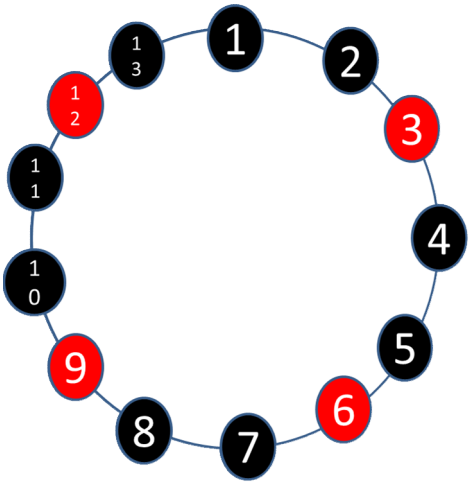
Với khoá chọn là k, ta sẽ loại người ở vị trí thứ k trên vòng tròn cho đến khi chỉ còn duy nhất một người.

Ví dụ với n =13 và k = 3, theo bạn ai là người được chọn???

Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

N= 13

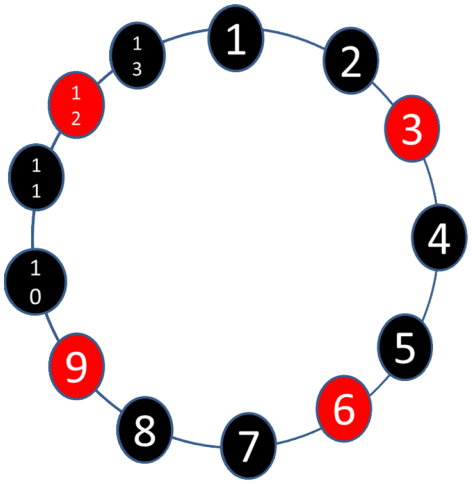
K= 3



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

N= 13

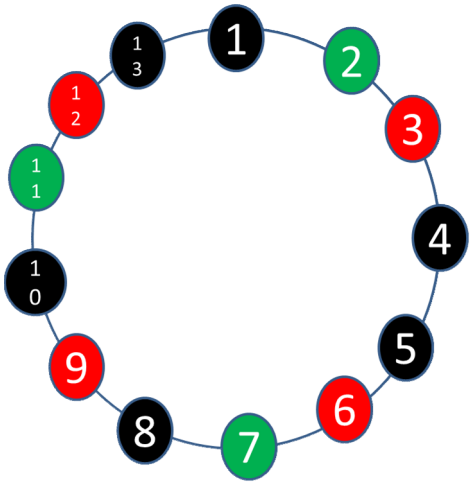
K= 3



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

N= 13

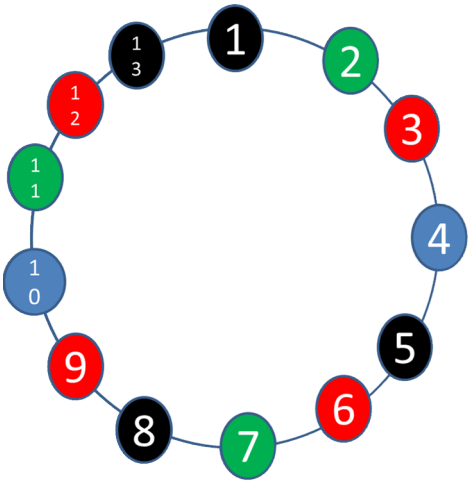
K= 3



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

N= 13

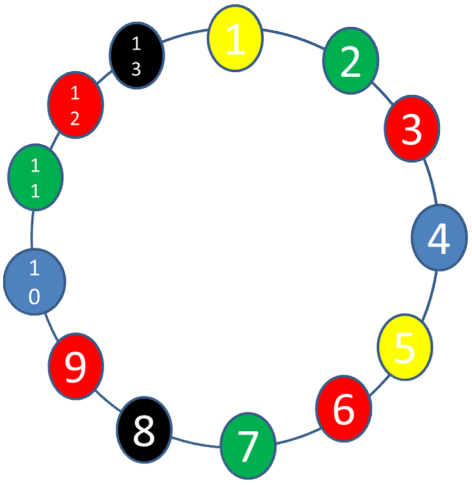
K= 3



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

N= 13

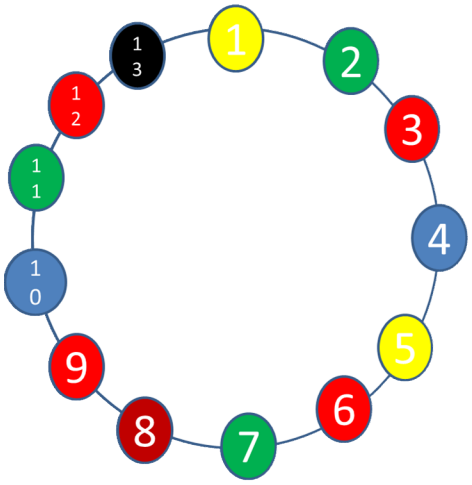
K= 3



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

N= 13

K= 3



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

Giải pháp 1: Josephus(int n, int k)

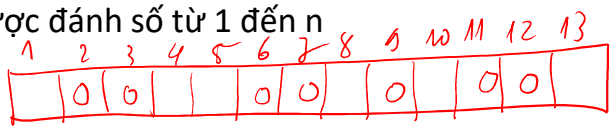
1. Khởi tạo mảng a gồm n phần tử được đánh số từ 1 đến n
2. Dem=n; i=1, vt=0;
3. Lặp khi dem>1

   nếu a[i]>0 thì vt=vt+1;

   nếu vt%k=0 và a[i]>0 thì { a[i]=0; vt=0; dem--;}

   i=i+1;

   nếu i>n thì i=1; // hết cuối thì quay lại đầu
4. Duyệt mảng tìm phần tử lớn hơn 0 và in ra kết quả



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

Viết chương trình:
void Josephus(int n, int k) { }

Nhập n, k và thử kết quả
N=13, k=3
N=1000000, k= 5
N=1000000, k= 500
Ghi lại thời gian thực thi

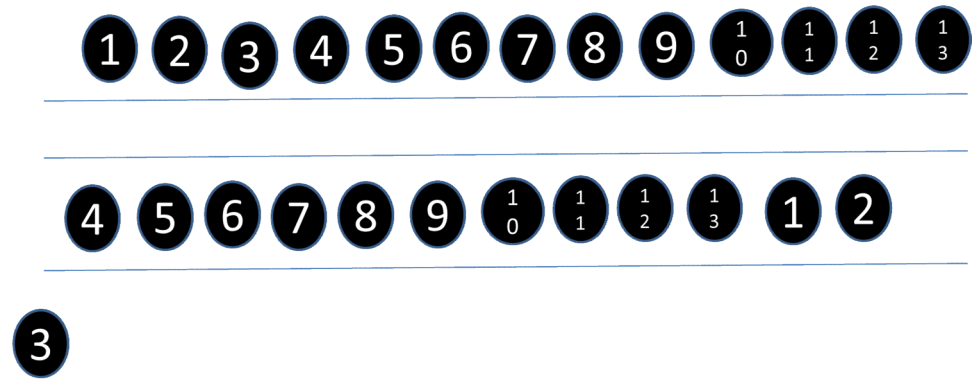
Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

```
void Josephus(int n, int k) {
        int *a= new int[n+1];  for(int i=0; i<=n; i++) a[i] =i;
        int dem=n, vt=0, i=1;
        while(dem>1) {
                if(a[i]>0) vt++;
                if(a[i]>0 && vt%k==0) { vt=0; dem--; a[i]=0;}
                i=i+1; if(i>n) i=1;
        }
     for(int i=1; i<=n; i++) if(a[i]>0) { cout<<“vua la: “<< i<< " “; break;}
}
```

Dùng Hàng đợi cho bài Josephus

N= 13

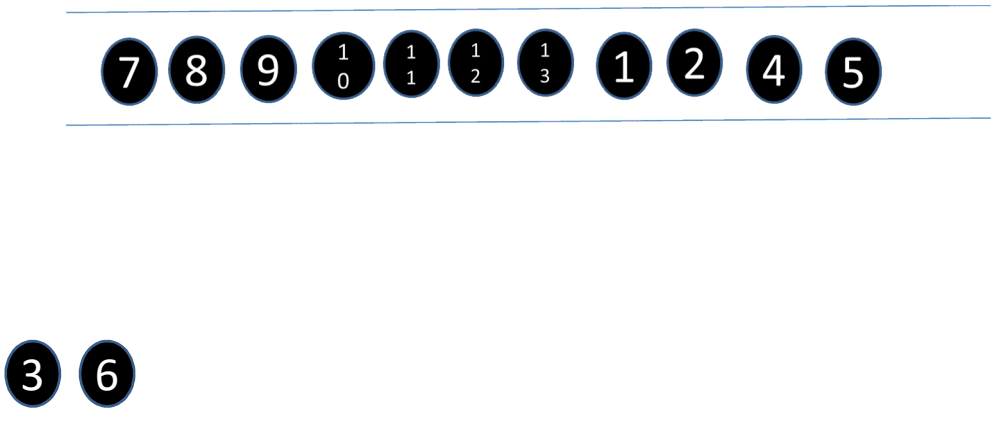
K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

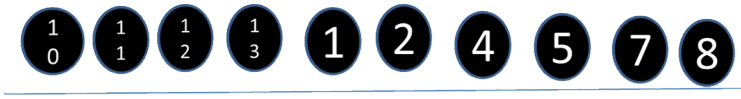
K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

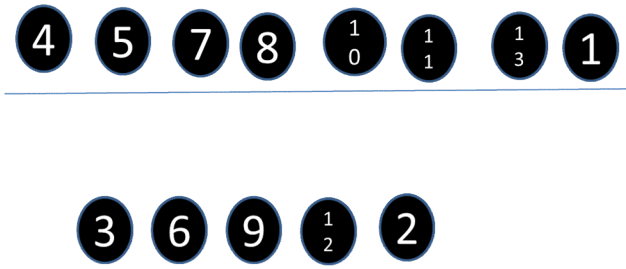
N= 13

K= 3

Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

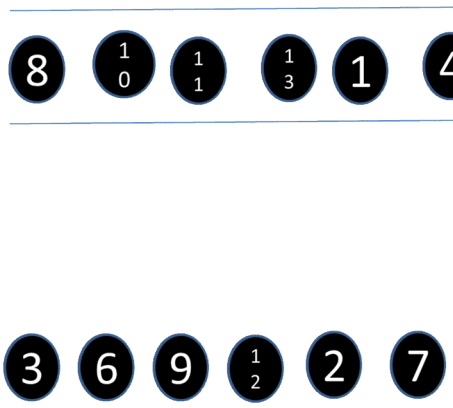
K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

K= 3

Ví dụ Ứng dụng Hàng đợi (Queue)

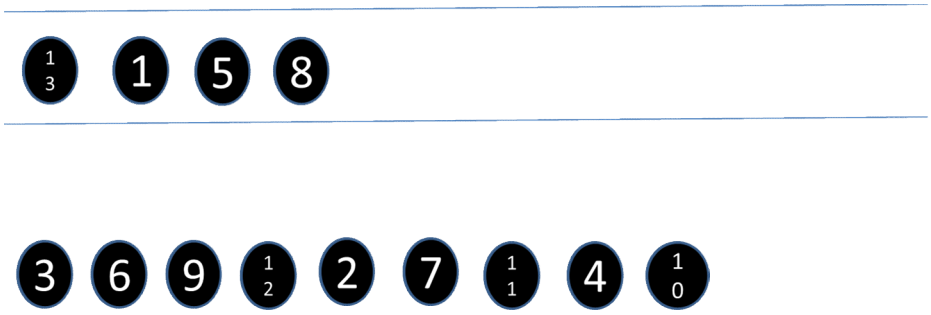
N= 13

K= 3

Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

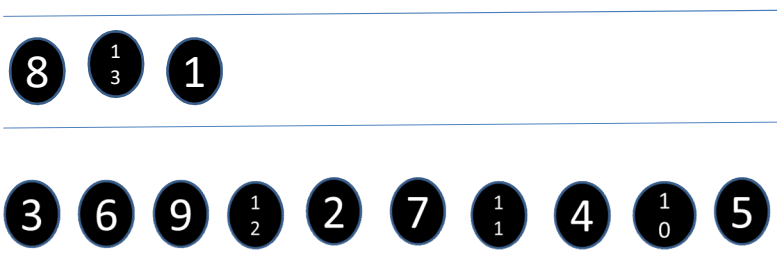
K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

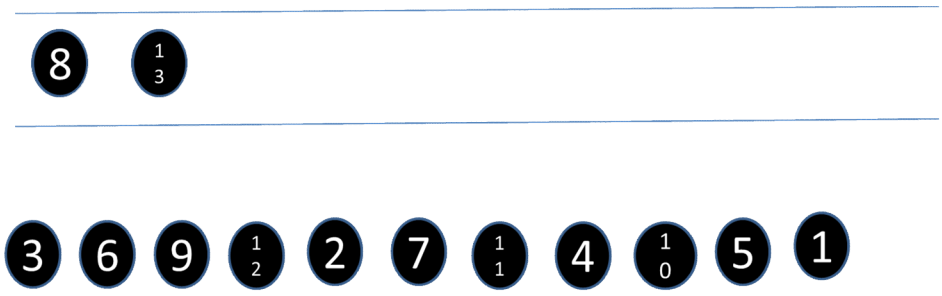
K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

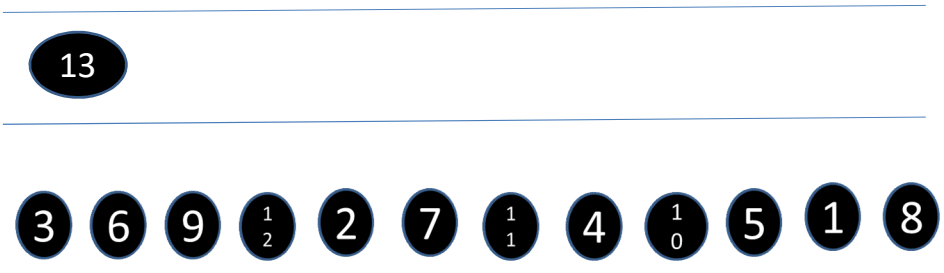
K= 3



Ví dụ Ứng dụng Hàng đợi (Queue)

N= 13

K= 3



Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

Giải pháp 2: Josephus2(int n, int k)

1. Khởi tạo hàng đợi Q chứa n phần tử được đánh số từ 1 đến n
2. Dem=n; vt=0;
3. Lặp khi dem>1
   Lấy phần tử x ở đầu hàng đợi Q;
   vt=vt+1;
   nếu vt%k=0 thì { vt=0; dem--;}
   ngược lại thêm x vào sau hàng đợi Q;
4. Lấy phần tử ra khỏi Q và in ra kết quả

Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

```
void josephus2(int n, int k){
        queue<int> q;
        for(int i=1; i<=n; i++) q.push(i);
        int d=n, v=0, x;
        while (d>1) {
                int x=q.front(); q.pop(); ++v;
                if(v==k) { d--; v=0;} else q.push(x);
        }
        cout<<"\n Phan tu con sot lai la: "<<q.front();
}
```

Ví dụ Ứng dụng Hàng đợi [JOSEPHUS]

Thảo luận theo nhóm!!!

Nhận định của nhóm về bài toán Josephus, nếu được cài đặt bằng mảng và cài đặt bằng hàng đợi

Đưa ra các giá trị của n và k để so sánh.

Ví dụ Ứng dụng Hàng đợi (Queue)

Cho file OPR.inp chứa n lệnh.

Lệnh $+v_i$ có nghĩa là thêm giá trị v_i vào hàng đợi

Lệnh – có nghĩa là lấy max trong hàng đợi.

Hãy thực hiện các lệnh trong file OPR.inp sau đó ghi các kết quả ra file OPR.out gồm các số còn lại trong hàng đợi theo thứ tự giảm dần

7
+4
+5
-
+9
-
+7
+1

7 4 1

Các thao tác Cơ bản trên Hàng đợi (Queue)

```
static <E> List<E> heapSort(Collection<E> c)
{
        Queue<E> queue = new PriorityQueue<E>(c);
        List<E> result = new ArrayList<E>();
        while (!queue.isEmpty()) result.add(queue.remove());
        return result;
}
```

Hàng đợi ưu tiên (Priority Queue)

```
public class MyPriorityQueue {
    Node head;
    MyPriorityQueue() { head= null;}
    boolean EmptyQ() {  return head==null;}
    int RemoveQ()     {
        int x=0;
        if(head==null) System.out.println(" \n Hang doi rong!");
        else  {       x= head.data; head= head.next;       }
        return x;
    }
```

```
class Node{
   int data;
   Node next;
   Node(int x) {data=x; next = null;}
   Node(int x, Node t) {data=x; next = t;}
}
```

Hàng đợi ưu tiên (Priority Queue)

```
public class MyPriorityQueue {
    void AddQ(int x) {
        Node t= new Node(x);
        if(head== null) head=t;
        else     if(x>=head.data) { t.next= head; head=t;}
                 else {
                          Node y= head,p= head;
                         while (p!=null && p.data>x) { y= p; p=p.next;   }
                             y.next=t; t.next=p;
                       }
     }
}
```

Hàng đợi ưu tiên (Priority Queue)

```
void xuly()    {
      try {
          FileReader fi= new FileReader("D:\\OPR.inp");    Scanner kb = new Scanner(fi);
          MyPriorityQueue Q= new MyPriorityQueue();

          Hãy bổ sung mã lệnh của bạn ở đây???

       } catch (FileNotFoundException e)
         {System.out.print("\n LOI DOC FILE: \n" + e.getMessage());}
    }
```

Hàng đợi ưu tiên (Priority Queue)

```
int n= kb.nextInt();
for(int i=1; i<=n; i++) {
    String t = kb.next();
   if(t.charAt(0)=='-')
     { int x= Q.RemoveQ();   System.out.print("\n Lay phan tu max: " +x); }
   else   {
    int p= Integer.parseInt(t);
    System.out.print("\n Them phan tu "+ p + " vao hang doi");  Q.AddQ(p);
   }
  }
 System.out.print("\n CAC SO CON LAI TRONG HANG DOI LA:\n");
while (!Q.EmptyQ()) System.out.print(" " + Q.RemoveQ());
```

Bài tập về nhà

1. Cài đặt thuật toán sắp xếp radix sort
2. Cài đặt thuật toán chuyển biểu thức trung tố sang dạng hậu tố

Tài liệu đọc thêm về ngăn xếp và hàng đợi

https://www.geeksforgeeks.org/queue-data-structure/

https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec06.pdf

Link YouTube

https://www.youtube.com/watch?v=FNZ5o9S9prU

https://www.youtube.com/watch?v=bxRVz8zklWM

https://www.youtube.com/watch?v=gnYM_G1ILm0