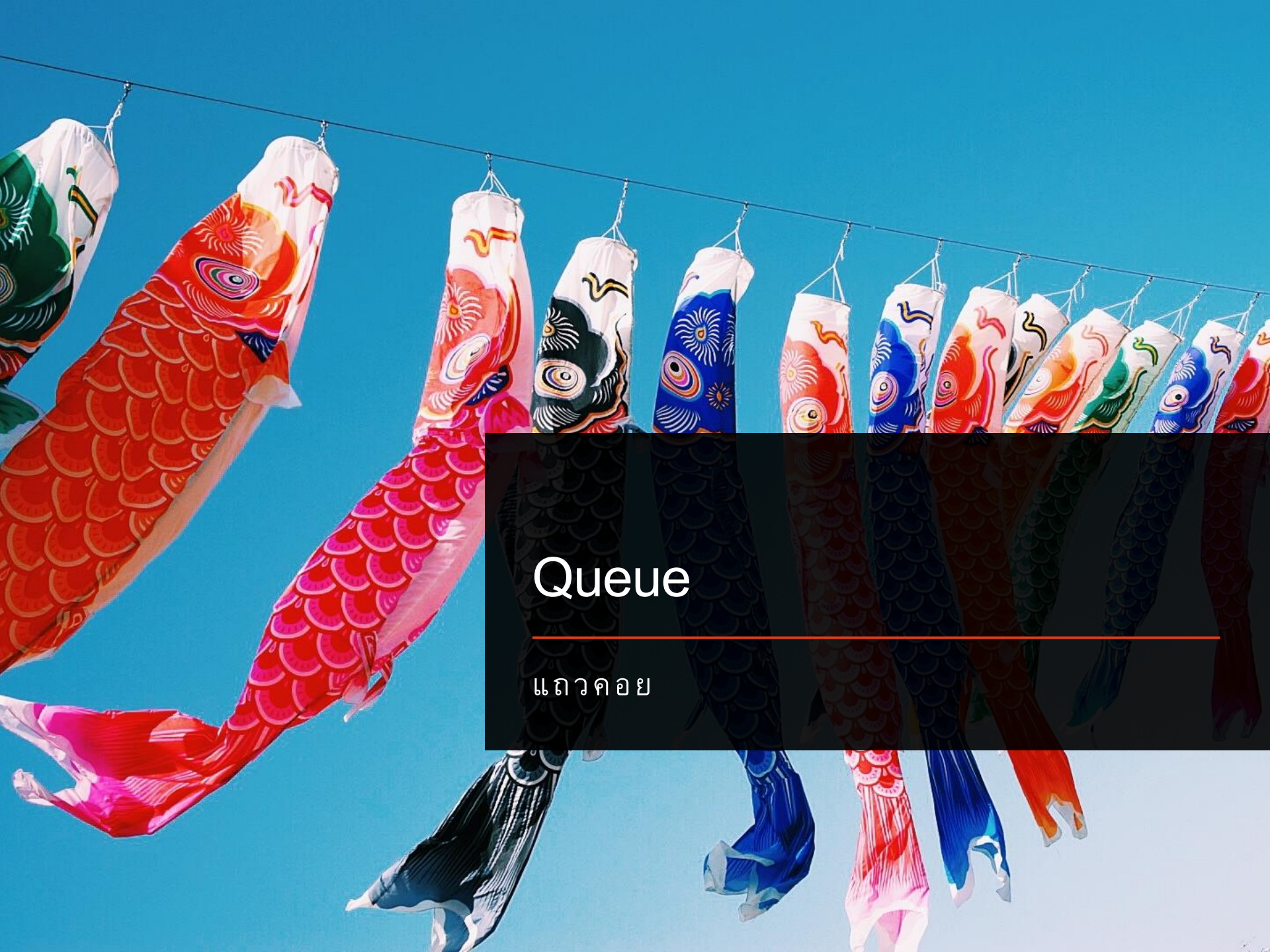


# Queue

แถวคอย

# Queue

คล้ายลิสต์

เพิ่มข้อมูลที่ปลายหนึ่ง (enqueue) / ลบข้อมูลที่อีกปลาย (dequeue)

First-In-First-Out (FIFO) เอาของที่เก็บเข้าก่อนออกมาก่อน

| Action | Queue q | x |
|---|---|---|
| Create queue | | |
| enqueue(1) | 1 | |
| enqueue(2) | 1 2 | |
| enqueue(3) | 1 2 3 | |
| x=dequeue() | 2 3 | 1 |
| enqueue(4) | 2 3 4 | |
| x=dequeue() | 3 4 | 2 |
| x=dequeue() | 4 | 3 |
| x=dequeue() | | 4 |

# ตัวอย่างการใช้ Queue

# ประโยชน์ของ Queue

ใช้เมื่อต้องการเอาของที่เก็บเข้าก่อนออกมาก่อน เช่น

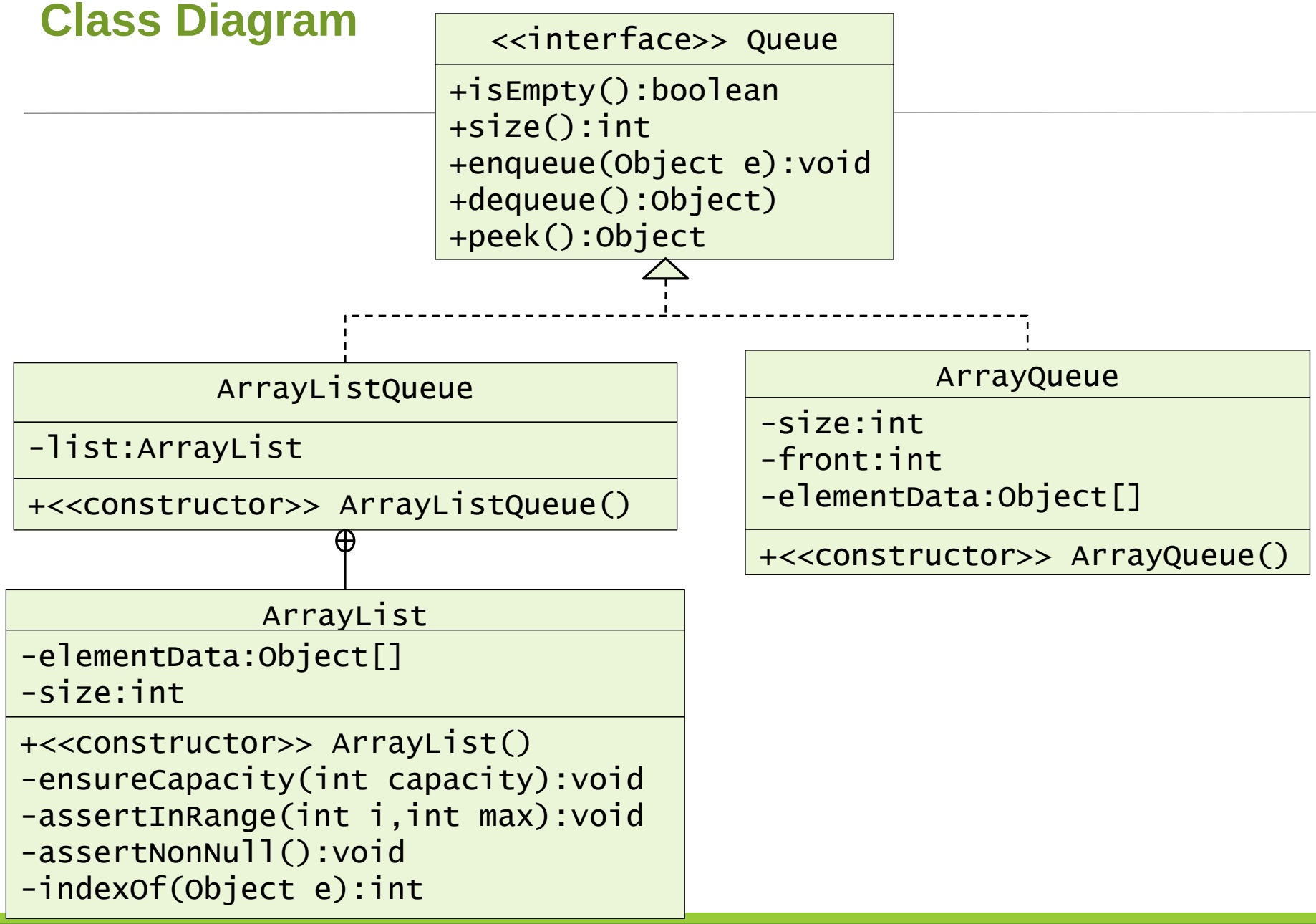
- การจัดลำดับของงานที่ processor ต้องทำ
- การเรียงลำดับแบบฐาน (Radix sort)
- การค้นตามแนวกว้าง (Breadth-first search)

# การสร้าง Queue

# Interface Queue

```
public interface Queue {
    public boolean isEmpty();          // queue ว่างหรือไม่
    public int size();                 // จำนวนข้อมูลใน queue
    public void enqueue(Object e);     // ใส่ e ต่อท้าย queue
    public Object dequeue();           // เอาข้อมูลออกจากหัว queue
    public Object peek();              // ดูข้อมูลที่หัว queue แต่ไม่เอาออก
}
```

# Class Diagram



# Class ArrayListQueue

สร้าง QUEUE ด้วย ARRAY LIST

# Class ArrayListQueue

```
public class ArrayListQueue implements Queue {
  private ArrayList list = new ArrayList();
  public boolean isEmpty()       { return list.isEmpty();    }
  public int size()              { return list.size();       }
  public void enqueue(Object e) { list.add(list.size(),e); }
  public Object peek() {
    if (isEmpty()) throw new IllegalStateException();
    return list.get(0);
  }
  public Object dequeue() {
    Object e = peek();
    list.remove(0);
    return e;
  }
}
```

# Using ArrayListQueue

```
public class test {
  public static void main() {
    Queue q = new ArrayListQueue();
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);
    System.out.println(q.dequque());
    q.enquque(4);
    System.out.println(q.dequque());
  }
}
```

| Queue q | output |
|---|---|
| | |
| 1 | |
| 1 2 | |
| 1 2 3 | |
| 2 3 | 1 |
| 2 3 4 | |
| 3 4 | 2 |

# Using ArrayListQueue

```
public class test {
  public static void main() {
    Queue q = new ArrayListQueue();
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);
    System.out.println(q.dequque());
    System.out.println(q.dequque());
    System.out.println(q.dequque());
  }
}
```

| Queue q | output |
|---|---|
| | |
| 1 | |
| 1 2 | |
| 1 2 3 | |
| 2 3 | 1 |
| 3 | 2 |
| | 3 |

# enqueue: Class ArrayListQueue

```
public class ArrayListQueue implements Queue {
  ...
  public void enqueue(Object e) {
    list.add(list.size(),e);
  }
  ...
}
```

```
public class ArrayList implements List {
  ...
  public void add(int i, Object e) {
    assertNonNull(e);
    assertInRange(i, size);
    ensureCapacity(size+1);
    for (int j=size-1; j>=i; j--)
      elementData[j+1] = elementData[j];
    elementData[i] = e;
    size++;
  }
  ...
}
```

```
public class ArrayList implements List
{
  ...
  private void ensureCapacity(int c) {
    if (capacity>elementData.length) {
      int s = Math.max(c,
              2*elementData.length);
      Object[] arr = new Object[s];
      for (int i=0;i<size; i++)
        arr[i]=elementData[i];
      elementData = arr;
    }
  }
...
}
```

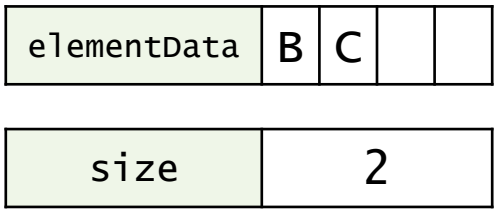
# dequeue: Class ArrayListQueue

```
public class ArrayListQueue
implements Queue {
  ...
  public Object dequeue() {
    Object e = peek();
    list.remove(0);
    return e;
  } ...
}
```

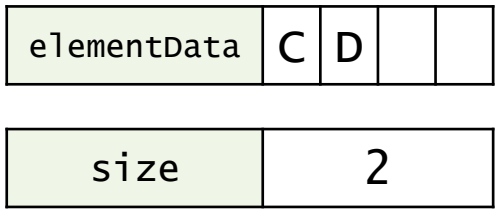
```
public class ArrayList implements List {
  ...

  public void remove(int i) {
    assertInRange(i, size-1);
    for (int j=i+1;j<size; j++)
      elementData[j-1] = elementData[j];
    elementData[--size] = null;
  }
 ...
}
```

# dequeue

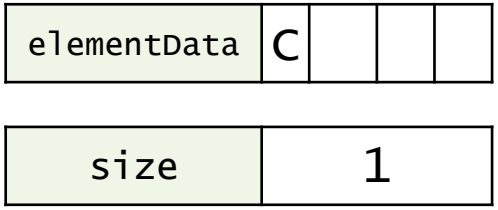
```
public Object dequeue() {
    Object e = peek();
    list.remove(0);
    return e;
  }
```



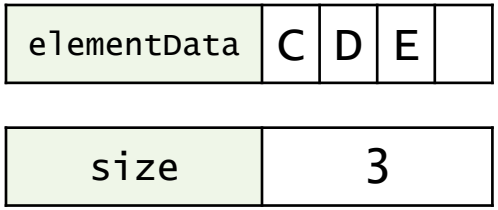

```
q.enqueue("D");
```




```
x=q.dequeue();
```




```
q.enqueue("E");
```



# peek

```
public Object peek() {
  if (isEmpty()) throw new IllegalStateException();
  return list.get(0);
}
```

| elementData | A | B | C | |
|---|---|---|---|---|

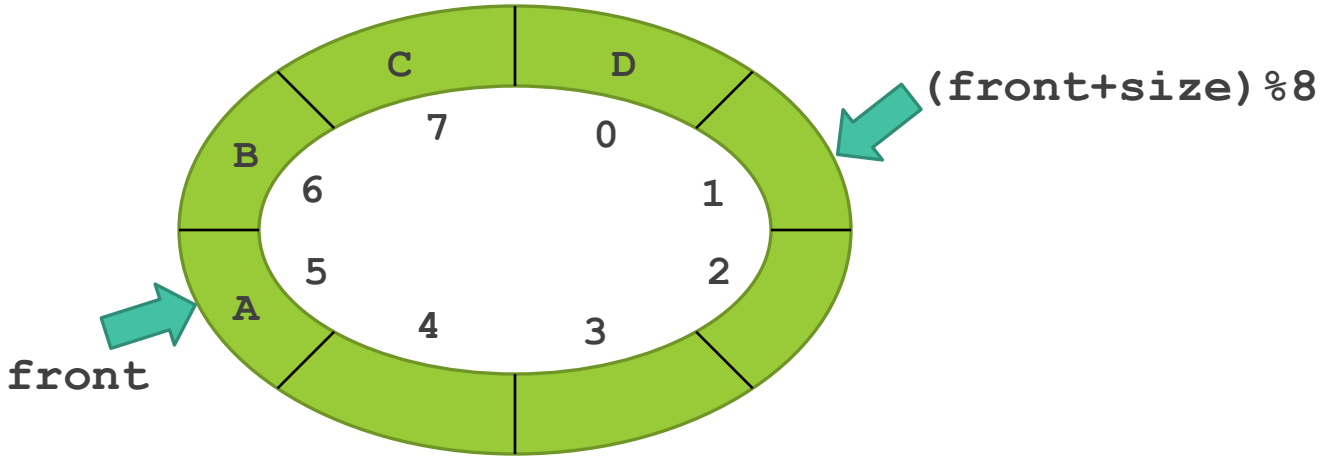
| size | 3 |
|---|---|

# Class ArrayQueue

สร้าง QUEUE ด้วย ARRAY

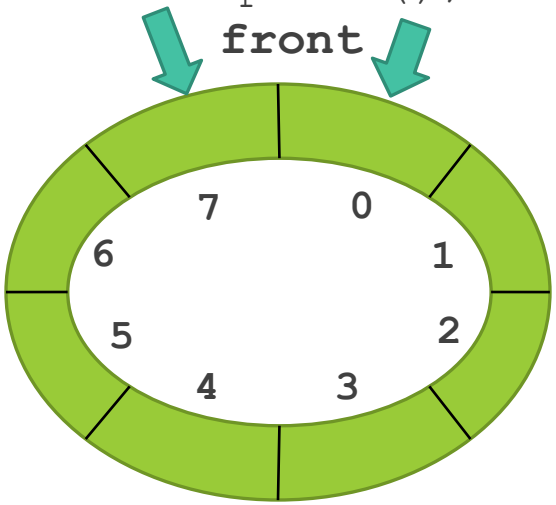
# Class ArrayQueue

```
public class ArrayQueue implements Queue {
  private Object[] elementData;
  private int front, size;
  public ArrayQueue()              {elementData = new Object[1];}
  public boolean isEmpty()         {return size==0;     }
  public int size()                {return size;     }
  public void enqueue(Object e)  { ... }
  private int inc(int i)           { ... }
  public Object peek()             { ... }
  public Object dequeue()          { ... }
}
```



# Class ArrayQueue

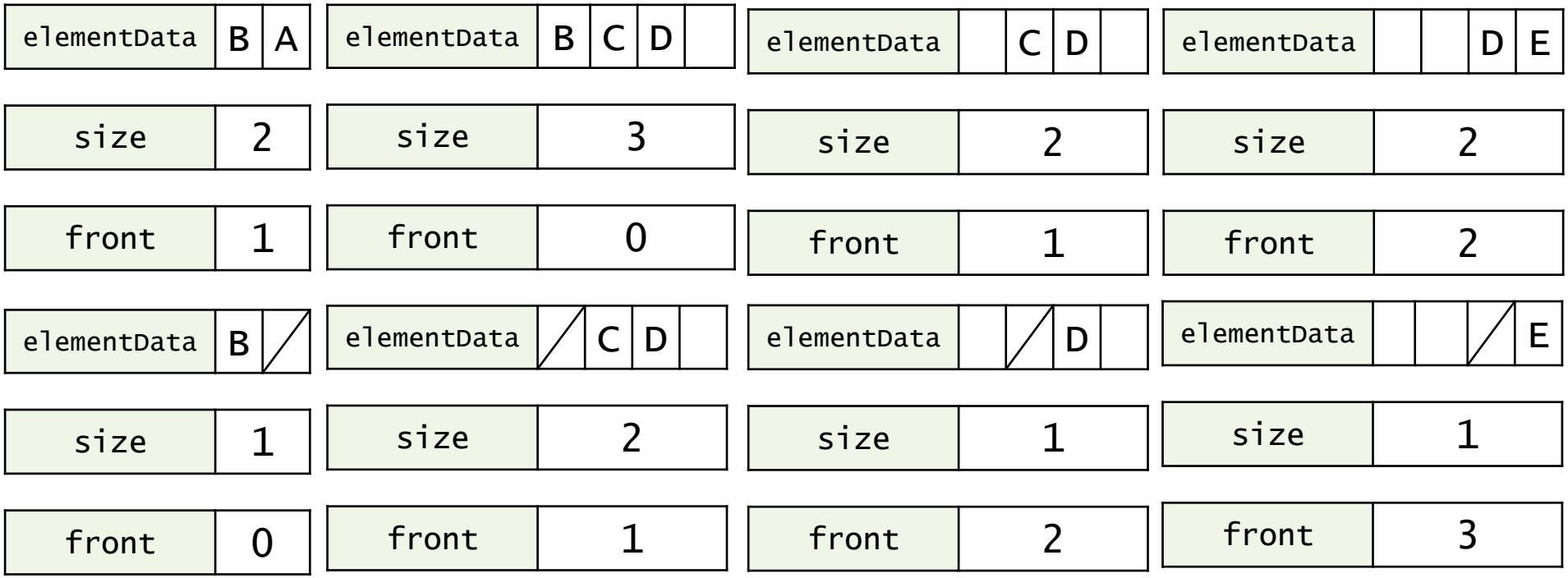
```
public class ArrayQueue implements Queue {
  private Object[] elementData;
  private int front, size;
  public ArrayQueue()              { ... }
  public boolean isEmpty()         { ... }
  public int size()                { ... }
  public void enqueue(Object e)    { ... }
  private int inc(int i)      {return (i+1)%elementData.length;}
  public Object peek() {
    if (isEmpty()) throw new IllegalStateException();
    return elementData[front];
  }
  public Object dequeue() {
    Object e = peek();
    front = inc(front);
    --size;
    return e;
 }
```



# dequeue

```
public Object dequeue() {
  Object e = peek();
  front = inc(front);
  --size;
  return e;
}
```

```
private int inc(int i) {
  return (i+1)%elementData.length;
}
```



# Class ArrayQueue

```
public class ArrayQueue implements Queue {
  private Object[] elementData;
  private int front, size;
  public ArrayQueue()              {elementData = new Object[1];}
  public boolean isEmpty()         {return size==0;}
  public int size()                {return size;}
  public void enqueue(Object e) {
    if (size==elementData.length) {
      Object[] arr = new Object[2*elementData.length];
      for (int i=0, j=front; i<size; i++, j=inc(j))
        arr[i] = elementData[j];
      front = 0;       elementData = arr;
    }
    int b = (front + size) % elementData.length;
    elementData[b] = e;          ++size;
  }
  private int inc(int i) { return (i+1)%elementData.length; }
  public Object peek() {…}
  public Object dequeue() {…}
```

# enqueue

```
public void enqueue(Object e) {
    if (size==elementData.length) {  // เพิ่มขนาดของ array
      Object[] arr = new Object[2*elementData.length];
      for (int i=0, j=front; i<size; i++, j=inc(j))
        arr[i] = elementData[j];
      front = 0;       elementData = arr;
     }
    int b = (front+size)%elementData.length; // หาท้ายคิว
    elementData[b] = e;
    ++size;
}
```

# enqueue

```
public void enqueue(Object e) {
   if (size==elementData.length) {  // เพิ่มขนาดของ array
     Object[] arr = new Object[2*elementData.length];
     for (int i=0, j=front; i<size; i++, j=inc(j))
       arr[i] = elementData[j];
     front = 0;       elementData = arr;
    }
   int b = (front + size) % elementData.length;
   elementData[b] = e;          ++ size;
  }
```

| elementData | B | A |
|---|---|---|

| size | 2 |
|---|---|

| front | 1 |
|---|---|

| elementData | A | B | C | |
|---|---|---|---|---|

| size | 3 |
|---|---|

| front | 0 |
|---|---|

# enqueue

```
public void enqueue(Object e) {
    if (size==elementData.length) {  // เพิ่มขนาดของ array
      Object[] arr = new Object[2*elementData.length];
      for (int i=0, j=front; i<size; i++, j=inc(j))
        arr[i] = elementData[j];
      front = 0;        elementData = arr;
    }
    int b = (front + size) % elementData.length;
    elementData[b] = e;          ++ size;
  }
```

| elementData | | B | C | |
|---|---|---|---|---|

| size | 2 |
|---|---|

| front | 1 |
|---|---|

| elementData | | B | C | D |
|---|---|---|---|---|

| size | 3 |
|---|---|

| front | 1 |
|---|---|

| elementData | E | B | C | D |
|---|---|---|---|---|

| size | 4 |
|---|---|

| front | 1 |
|---|---|

| elementData | B | C | D | E | F | | | |
|---|---|---|---|---|---|---|---|---|

| size | 5 |
|---|---|

| front | 0 |
|---|---|

# การนำ Queue ไปใช้

RADIX SORT

# การเรียงลำดับข้อมูลแบบฐาน

# Radix Sort

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 324 | 601 | 12 | 789 | 120 | 54 | 313 | 804 | 545 | 97 |

| q[0] | q[1] | q[2] | q[3] | q[4] | q[5] | q[6] | q[7] | q[8] | q[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 804 | | | | | |
| | | | | 54 | | | | | |
| 120 | 601 | 12 | 313 | 324 | 545 | | 97 | | 789 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 120 | 601 | 12 | 313 | 324 | 54 | 804 | 545 | 97 | 789 |

| q[0] | q[1] | q[2] | q[3] | q[4] | q[5] | q[6] | q[7] | q[8] | q[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 804 | 313 | 324 | | | | | | | |
| 601 | 12 | 120 | | 545 | 54 | | | 789 | 97 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 601 | 804 | 012 | 313 | 120 | 324 | 545 | 054 | 789 | 097 |

| q[0] | q[1] | q[2] | q[3] | q[4] | q[5] | q[6] | q[7] | q[8] | q[9] |
|---|---|---|---|---|---|---|---|---|---|
| 97 | | | | | | | | | |
| 54 | | | 324 | | | | | | |
| 12 | 120 | | 313 | | 545 | 601 | 789 | 804 | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 54 | 97 | 120 | 313 | 324 | 545 | 601 | 789 | 804 |

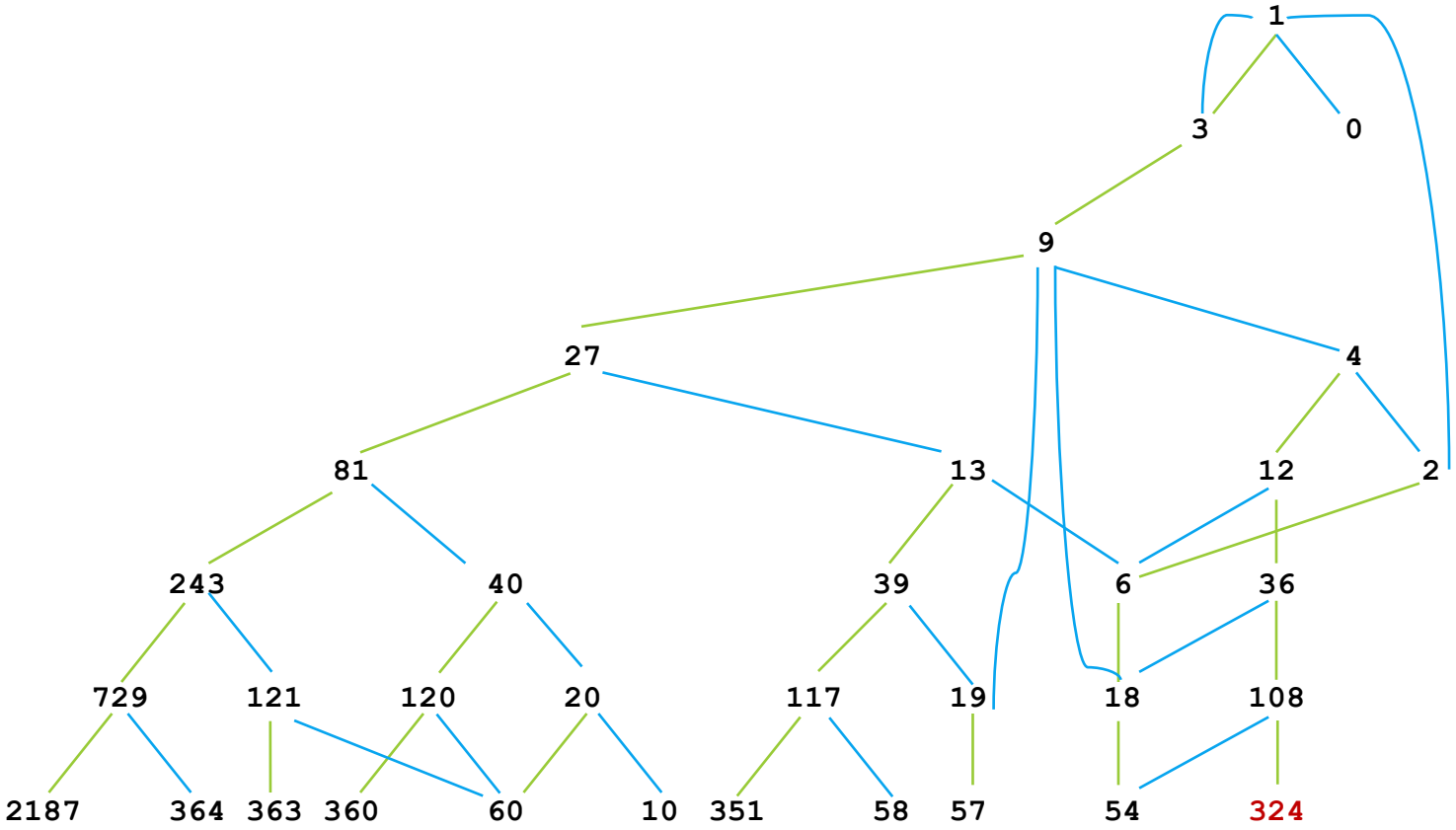
# Radix Sort

```
public class ArrayUtil {
  public static void radixSort(Integer[] data, int d) {
    Queue[] q = new ArrayQueue[10];  // one queue for one digit
    for (int i=0; i<q.length; i++)
      q[i] = new ArrayQueue();
    for (int k=0; k<d; k++) { // round k, d = max no. of digits
      for (int i=0; i<data.length; i++) { // get data
        int p = (data[i].intValue()//10^k)%10; // find kth digit
        q[p].enqueue(data[i]);           // add data in queue p
      }
      for (int i=0, j=0; i<q.length; i++)
        while (!q[i].isEmpty())       // move from q[i] to data
          data[j++] = (Integer) q[i].dequeue();
    }
  }
}
```

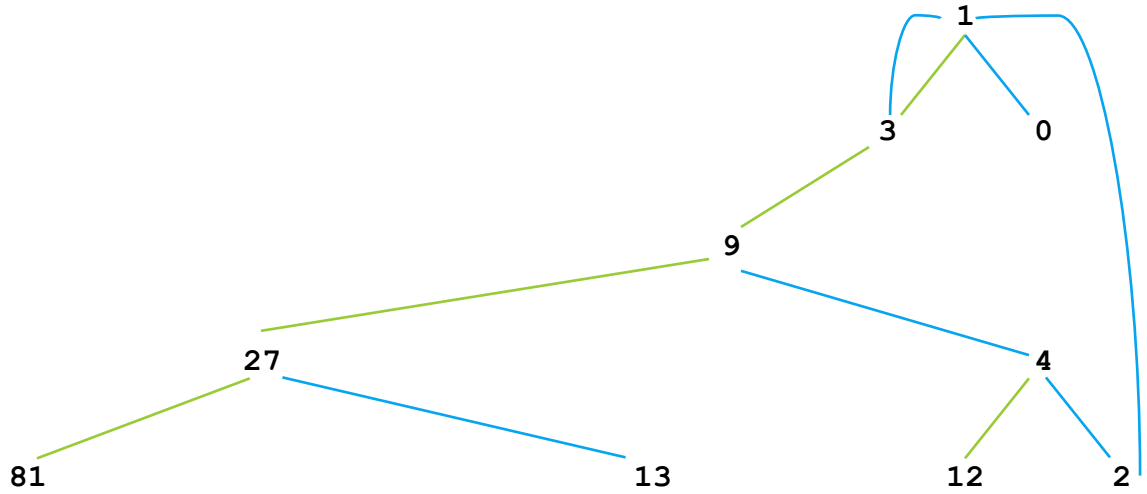
BREADTH FIRST SEARCH

# การค้นตามแนวกว้าง

# Puzzle: คูณ 3 หาร 2

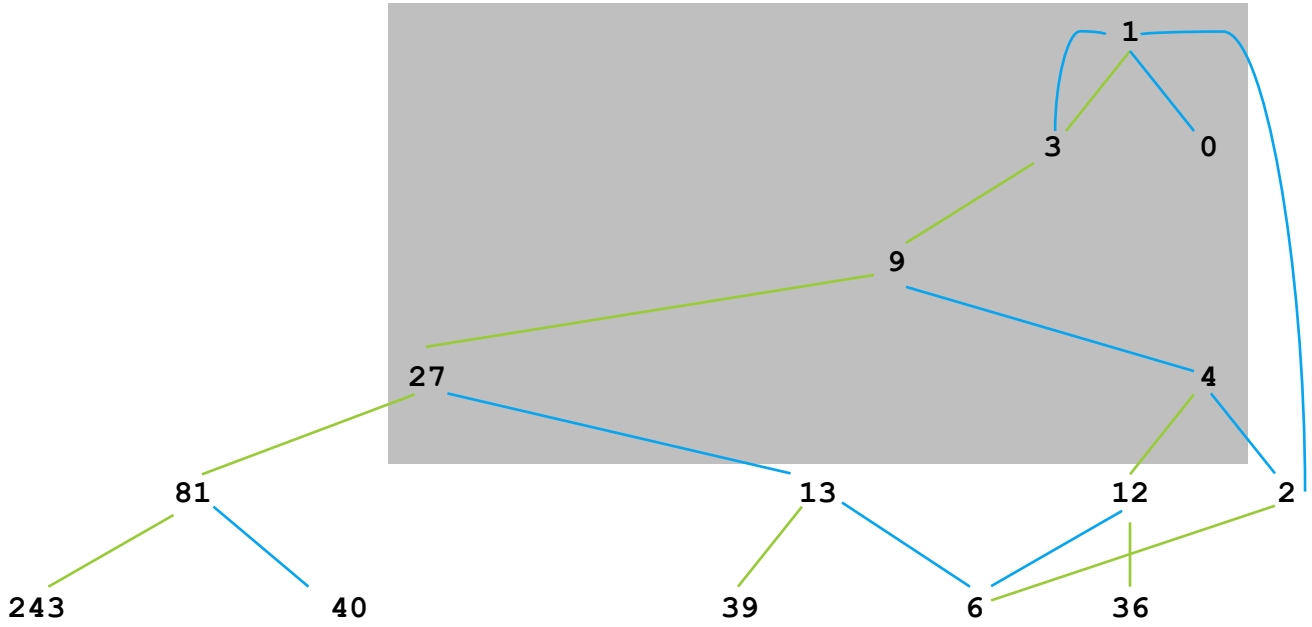


# ใช้ queue ในการค้นแบบกว้าง : คูณ 3 หาร 2



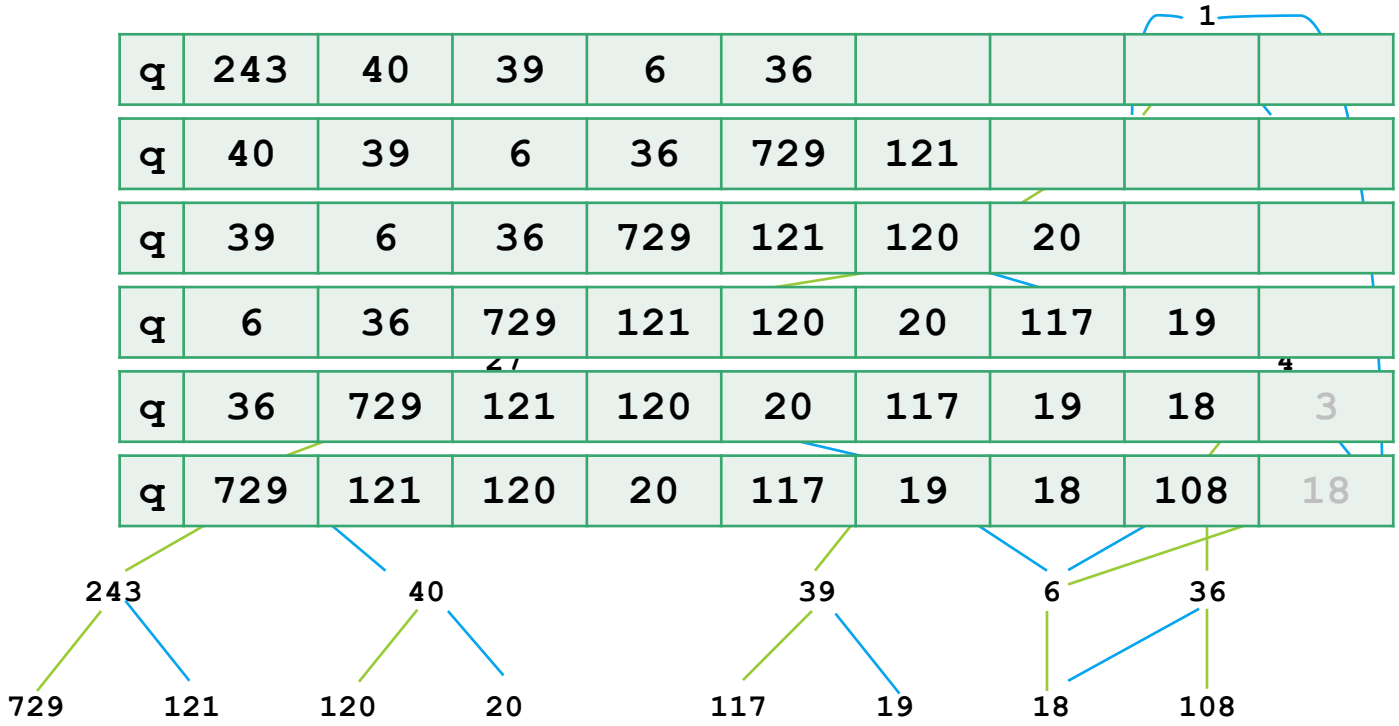

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| q | 1 | | | | | | | | | |
| q | 3 | 0 | | | | | | | | |
| q | 0 | 9 | 1 | | | | | | | |
| q | 9 | | | | | | | | | |
| q | 27 | 4 | | | | | | | | |
| q | 4 | 81 | 13 | | | | | | | |
| q | 81 | 13 | 12 | 2 | | | | | | |

# ใช้ queue ในการค้นแบบกว้าง : คูณ 3 หาร 2




| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| q | 81 | 13 | 12 | 2 | | | | | | |
| q | 13 | 12 | 2 | 243 | 40 | | | | | |
| q | 12 | 2 | 243 | 40 | 39 | 6 | | | | |
| q | 2 | 243 | 40 | 39 | 6 | 36 | | | | |
| q | 243 | 40 | 39 | 6 | 36 | | | | | |

# ใช้ queue ในการค้นแบบกว้าง : คูณ 3 หาร 2



# แก้ Puzzle คูณ 3 หาร 2

```
public class M3D2 {
  public static void main (String[] args) {
    m3d2(324);
  }
  private static class Node {
    int value;
    Node prev;
    Node(int v, Node p) { value=v; prev=p; }
    public boolean equals(Object e) { return value==((Node) e).value; }
  }

  public static String m3d2(int target) { ... }

  public static String solution(Node v) {
    if (v.prev==null) return "1";
    return solution(v.prev)+(v.prev.value/2 == v.value ? "/2" : "x3");
  }
}
```

# แก้ Puzzle คูณ 3 หาร 2

```
public static String m3d2(int target) {
  Queue q = new ArrayQueue();
  Set s = new ArraySet();
  Node v = new Node(1, null);
  q.enqueue(v); s.add(v);

  while (!q.isEmpty()) {
    v = (Node) q.dequeue();
    Node v1 = new Node(v.value/2, v);
    if (v1.value==target) break;
    if (!s.contains(v1)) {  q.enqueue(v1);   s.add(v1);   }

    Node v2 = new Node(v.value*3, v);
    if (v2.value==target) break;
    if (!s.contains(v2)) {  q.enqueue(v2);   s.add(v2);   }
  }
  if (v1.value == target) return solution(v1);
  else if (v2.value == target) return solution(v2);  else return("??");
}
```

# Set

```
public interface Set extends Collection {
  public void add(Object e);
}

public class ArraySet extends ArrayCollection
                      implements Set {

  public ArraySet(int cap) {
    super(cap);
  }

  public void add(Object element) {
    if (!contains(element)) {
      super.add(element);
    }
  }
}
```