แถวคอย

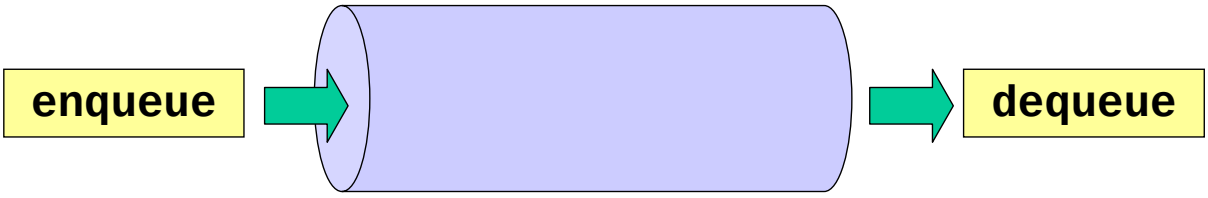
(Queue)

หัวข้อ

- นิยามแถวคอย และอินเตอร์เฟส Queue
- การสร้างแถวคอยด้วยอาเรย์
- ตัวอย่างการใช้งานแถวคอย
  - ที่พักข้อมูล
  - การเรียงลำดับแบบฐาน
  - การค้นคำตอบตามแนวกว้าง
  - การหาวิถีสั้นสุด

การเพิ่ม/ลบข้อมูลในแถวคอย

- ข้อมูล เข้าก่อน ออกก่อน (First-In First-Out)



แถวคอย : Queue

```
public interface Queue {
  public boolean isEmpty();
  public int size();
  public void enqueue(Object e);
  public Object peek();
  public Object dequeue();
}
```

A B C * * D E * * *

A B C D E

Queue คล้าย List

- Queue คือ list ที่เราเพิ่มปลายด้านหนึ่ง และลบที่ปลายอีกด้าน
- สร้าง queue ด้วย list แบบง่าย ๆ

```
public class ArrayListQueue implements Queue {
  private List list = new ArrayList(10);
  public boolean isEmpty() {return list.isEmpty();}
  public int size() {return list.size();}
  public void enqueue(Object e) {list.add(e);}
  public Object peek() {
    if (isEmpty()) throw new NoSuchElementException();
    return list.get(0);
  }
  public Object dequeue() {
    Object e = peek();
    list.remove(0);
    return e;
  }
}
```

enqueue เพิ่มท้าย list ใช้เวลา O(1)
แต่ dequeue ลบหัว list ใช้เวลา O(n)

สร้าง Queue ด้วย LinkedList

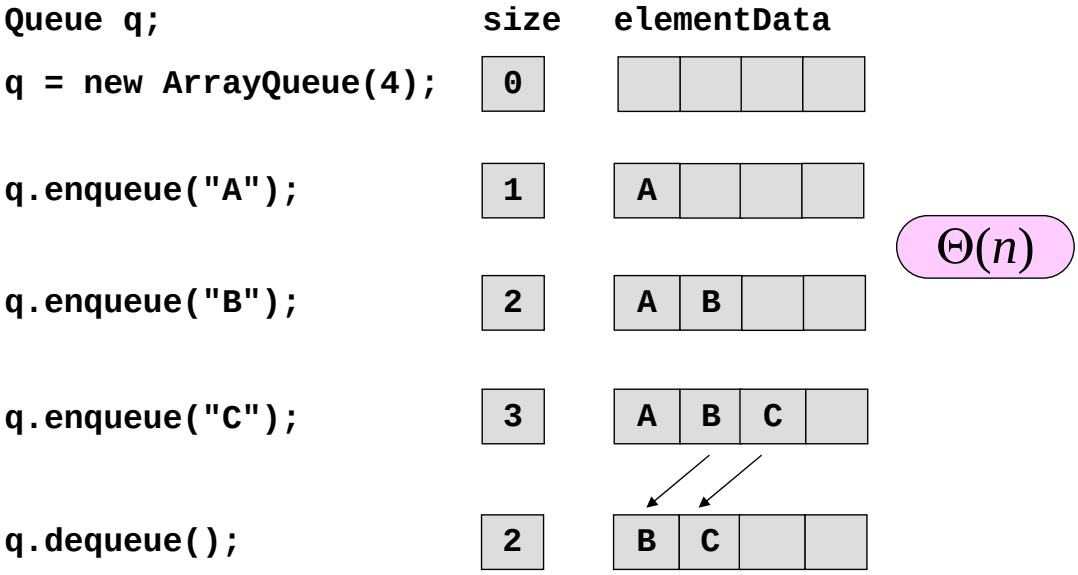
- เลือกใช้ Circular doubly linked list w/ header

```
public class LinkedListQueue implements Queue {
  private List list = new LinkedList(10);
  public boolean isEmpty() {return list.isEmpty();}
  public int size() {return list.size();}
  public void enqueue(Object e) {list.add(e);}
  public Object peek() {
    if (isEmpty()) throw new NoSuchElementException();
    return list.get(0);
  }
  public Object dequeue() {
    Object e = peek();
    list.remove(0);
    return e;
  }
}
```

enqueue เพิ่มท้าย list ใช้เวลา O(1)
dequeue ลบหัว list ใช้เวลา O(1)

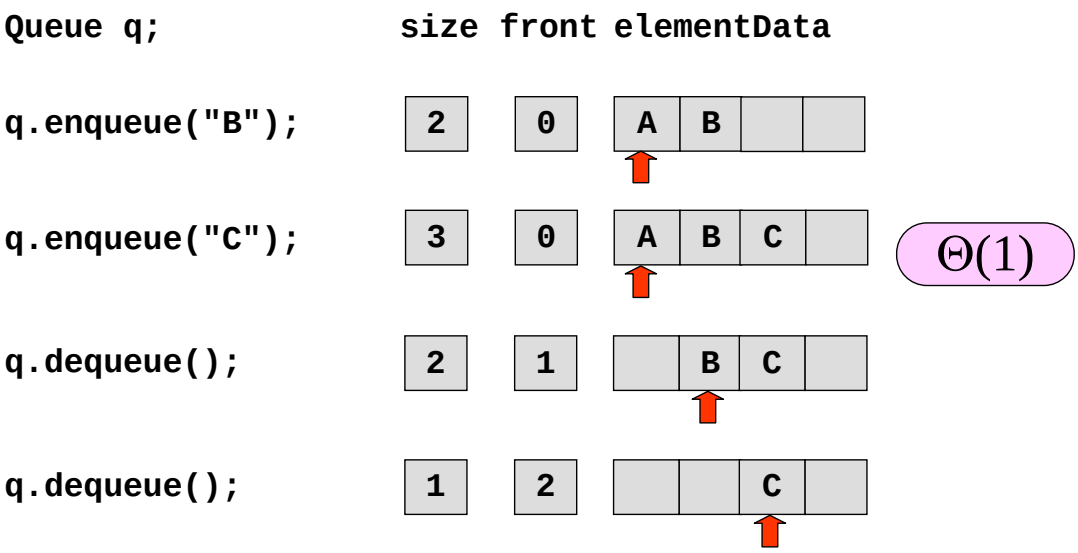
ArrayQueue : สร้าง Queue ด้วยอาเรย์

- เพิ่มที่ท้ายคิว ลบที่หัวคิว
- ให้หัวคิวอยู่ที่ index 0 เสมอ
- ตอนลบต้องใช้เวลา $\Theta(n)$



ArrayQueue : ตำแหน่งหัวคิวเปลี่ยนได้

- จำ index ของหัวคิว
- ลบ : อย่าย้ายข้อมูล แต่ใช้วิธีการเลื่อนตำแหน่งหัวคิว



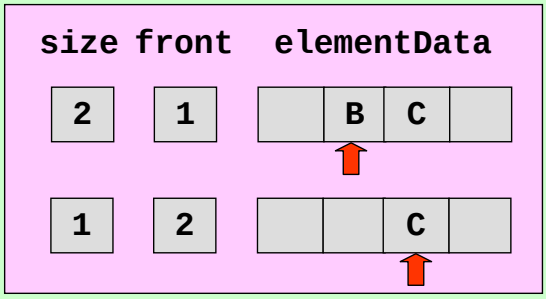
ArrayQueue

```
public class ArrayQueue implements Queue {
  private Object[] elementData;
  private int size;
  private int front;

  public ArrayQueue(int cap) {
    elementData = new Object[cap];
    size = front = 0;
  }
  public boolean isEmpty() {
    return size == 0;
  }
  public int size() {
    return size;
  }
  ...
```

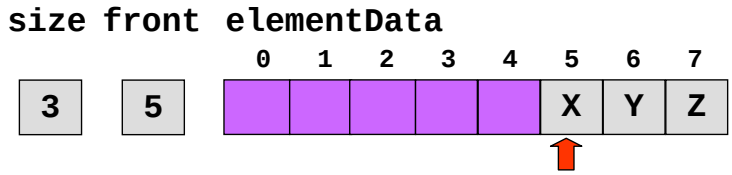

enqueue, peek, dequeue

```
public class ArrayQueue implements Queue {
  private Object[] elementData;
  private int size;
  private int front;
  ...
  public void enqueue(Object e) {
    // ...  ขยายอาเรย์ ถ้าเต็ม
    elementData[front + size] = e; size++;
  }
  public Object peek() {
    if (isEmpty()) throw new NoSuchElementException();
    return elementData[front];
  }
  public Object dequeue() {
    Object e = peek();
    elementData[front++] = null;
    size--;
    return e;
  }
  ...
```

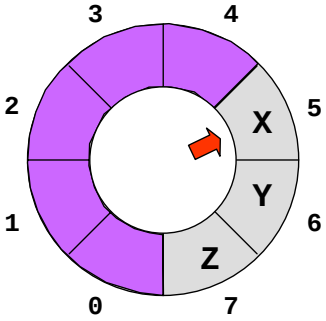


มองอาเรย์เป็นวงวน

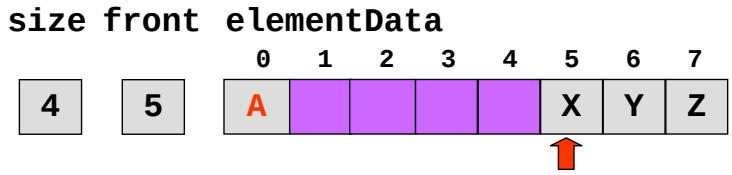
- ถ้าตัวท้ายคิวอยู่ท้ายอาเรย์ เติมตัวใหม่ไม่ได้
- มองอาเรย์ให้เป็นแบบวงวน จะใช้เนื้อที่ได้เต็มที่

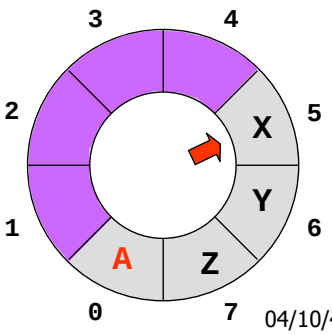



```
q.enqueue("A");
```



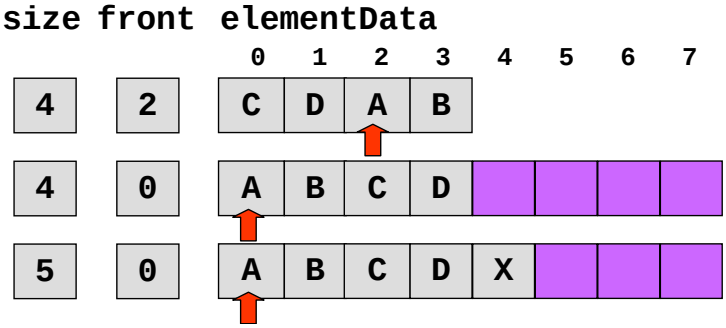

```
b = (front + size) % elementData.length;
elementData[b] = e; size++;
```





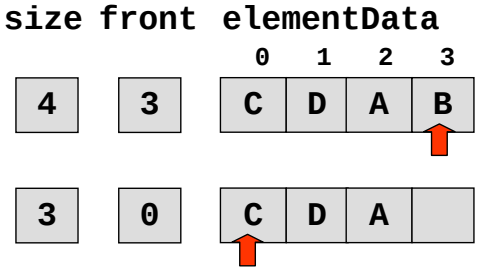
ArrayQueue : enqueue

```
public class ArrayQueue implements Queue {
  ...
  public void enqueue(Object e) {
    if (size == elementData.length) {
      Object[] a = new Object[2 * elementData.length];
      for (int i = 0, j = front; i < size;
           i++, j = (j+1)%elementData.length)
        a[i] = elementData[j];
      front = 0; elementData = a;
    }
    int b = (front + size) % elementData.length;
    elementData[b] = e; size++;
  }
```



ArrayQueue : dequeue

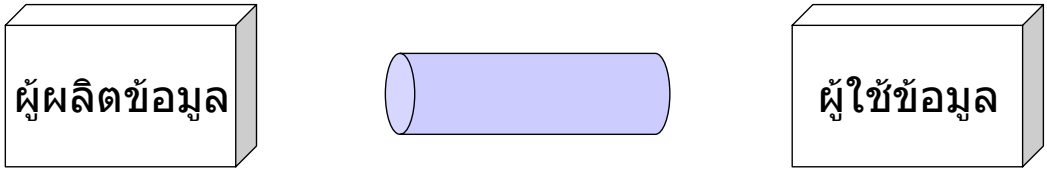
```
public Object dequeue() {
  Object e = peek();
  elementData[front] = null;
  front = (front + 1) % elementData.length;
  size--;
  return e;
}
```



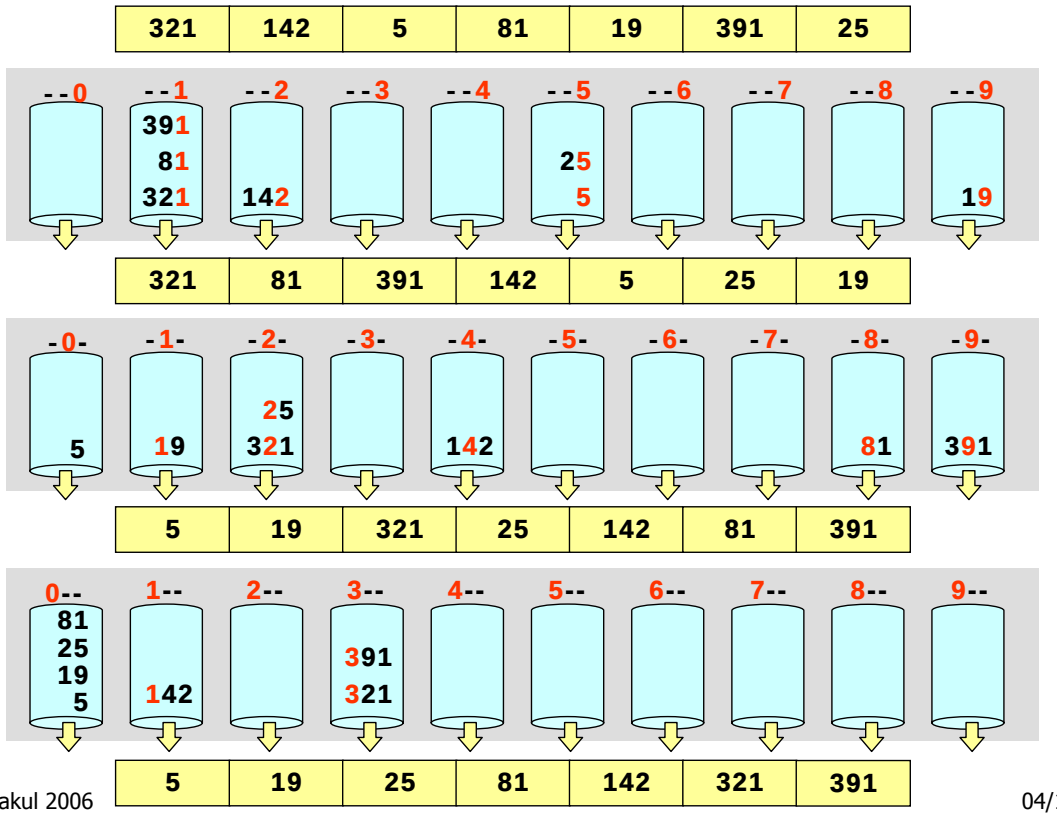
ตัวอย่างการใช้ queue

- เป็นที่พักข้อมูล
- การเรียงลำดับแบบฐาน
- การค้นคำตอบตามแนวกว้าง
- การหาวิถีสั้นสุด
- ...

การใช้แถวคอยเป็นที่พักข้อมูล



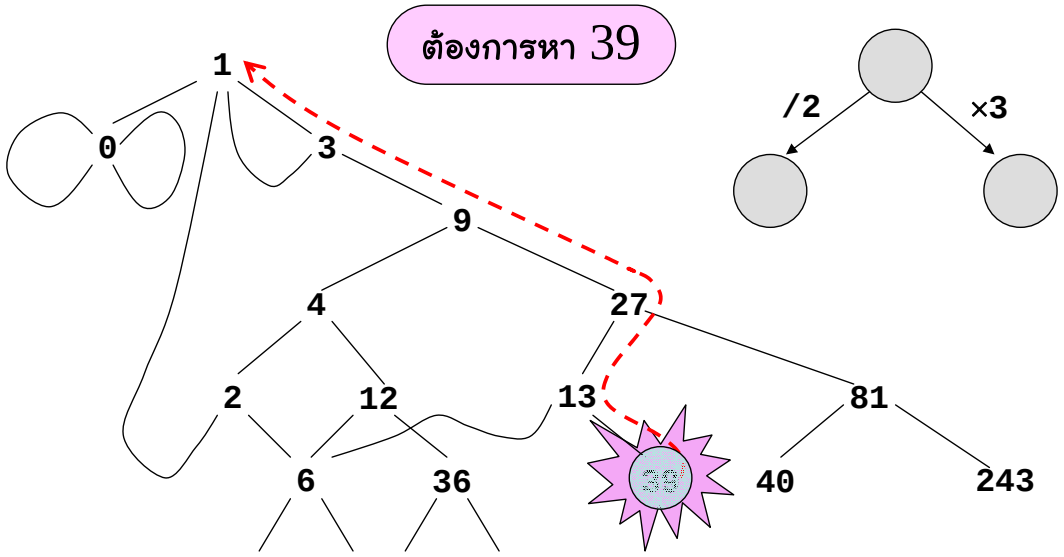
Radix Sort



ปัญหาคูณสามหารสอง

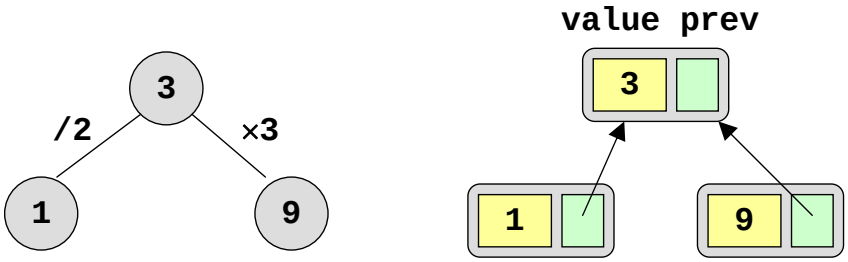
- ให้จำนวนเต็ม v
- เริ่มด้วย 1 จะต้องทำการ x3 และหรือ /2 (ปัดเศษทิ้ง) อย่างไร จึงมีค่าเท่ากับ v
- เช่น
  - v = 10 = 1x3x3x3/2/2/2
  - v = 31 = 1x3x3x3x3/2/2/2/2/2x3x3/2
- แก้ปัญหานี้อย่างไร ?
- ขอเสนอวิธีลุยทุกรูปแบบ

การค้นตามแนวกว้าง



$39 = 1 \times 3 \times 3 \times 3 / 2 \times 3$

ปมต่าง ๆ ระหว่างการค้น

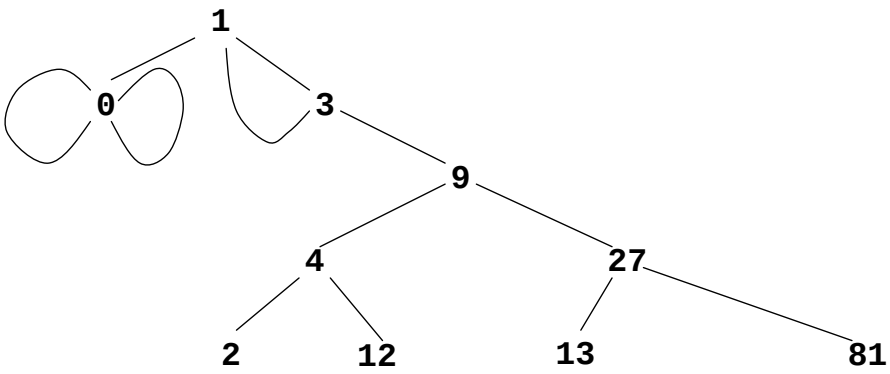



```
class Node {
  int value;
  Node prev;
  Node(int v, Node p) {
    this.value = v;
    this.prev = p;
  }
  public boolean equals(Object o) {
    if (!(o instanceof Node)) return false;
    return this.value == ((Node) o).value;
  }
}
```

สอง nodes เท่ากันก็เมื่อ values ทั้งสองเท่ากัน

ใช้ Set และ Queue

- ใช้ queue เก็บเฉพาะ nodes ที่ยังไม่แตกกิ่ง
- ใช้ set เก็บทุก nodes ที่เคยผลิต

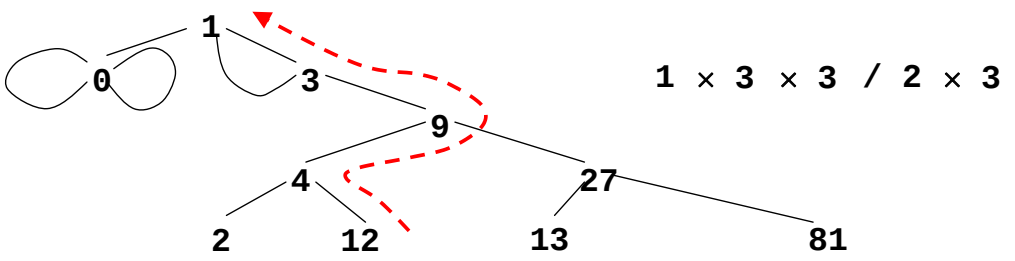


ตัวโปรแกรม

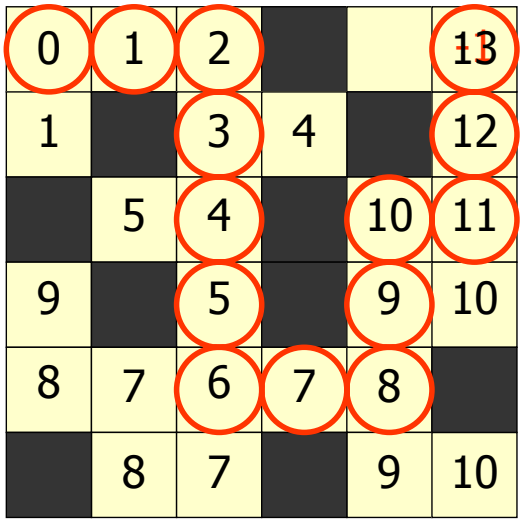
```
public static void bfsM3D2(int target) {
  Set set = new ArraySet(100);
  Queue q = new ArrayQueue(100);
  Node v = new Node(1,null);  // เริ่มด้วย 1
  q.enqueue(v); set.add(v);
  while( !q.isEmpty() ) {
    v = (Node) q.dequeue();
    if (v.value == target) break;
    Node v1 = new Node(v.value/2, v);  // ลองหารสอง (ปัดเศษ)
    Node v2 = new Node(v.value*3, v);  // ลองคูณสาม
    if (!set.contains(v1)) {q.enqueue(v1); set.add(v1);}
    if (!set.contains(v2)) {q.enqueue(v2); set.add(v2);}
  }
  if (v.value == target) showSolution(v);
}
```


การแสดงผลลัพธ์

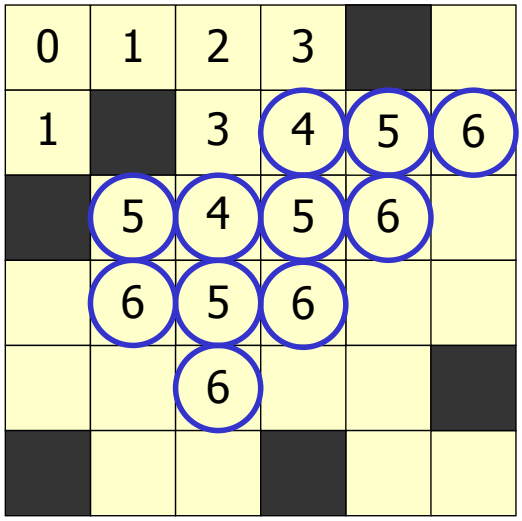
```
static void showSolution(Node v) {
  if (v.prev != null) {
    showSolution(v.prev);
    System.out.print((v.prev.value / 2 == v.value) ?
                     "/ 2" : "× 3");
  } else {
    System.out.print("1 ");
  }
}
```



การหาวิถีสั้นสุด



การใช้แถวคอยในการหาวิถีสั้นสุด



ใช้แถวคอยเก็บตำแหน่งของช่องที่รอขยาย

```
class Pos {
  int row, col;
  Pos(int r, int c) {row = r; col = c;}
}
```

โปรแกรมหาวิถีสั้นสุด

```
static void findPath(int[][] map, Pos source, Pos target) {
  map[source.row][source.col] = 0;   // ต้นทาง
  map[target.row][target.col] = -1;  // ปลายทาง
  Queue q = new ArrayQueue(map.length); q.enqueue(source);
  while (!q.isEmpty()) {
    Pos p = (Pos) q.dequeue();
    if (p.row == target.row && p.col == target.col) break;
    expand(map, q, p.row + 1, p.col, map[p.row][p.col]+1);
    expand(map, q, p.row - 1, p.col, map[p.row][p.col]+1);
    expand(map, q, p.row, p.col + 1, map[p.row][p.col]+1);
    expand(map, q, p.row, p.col - 1, map[p.row][p.col]+1);
  }
}
static void expand(int[][] map, Queue q,int r,int c,int k){
  if (r<0 || r>=map.length ||
      c<0 || c>=map[r].length || map[r][c] != 0) return;
  map[r][c] = k;
  q.enqueue(new Pos(r, c));
}
```


สรุป

- ประยุกต์แถวคอยในการแก้ปัญหาหลากหลาย
- การดำเนินการหลัก : enqueue / dequeue / peek
- สร้างแถวคอยได้ง่ายด้วยอาเรย์ (มองแบบวงวน)
- ถ้าจองขนาดให้เพียงพอ การทำงานทุกครั้งเป็น $\Theta(1)$