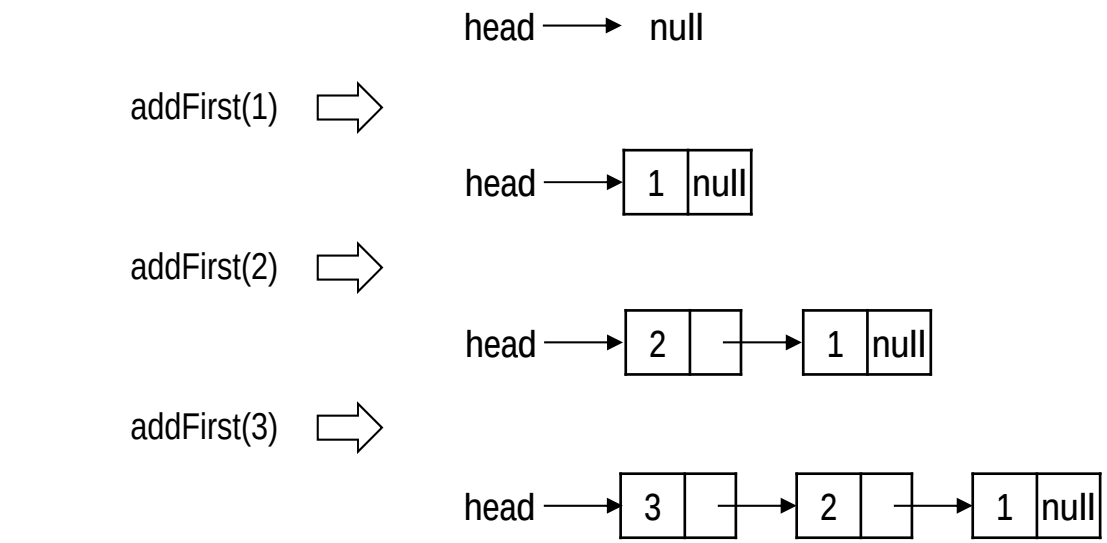
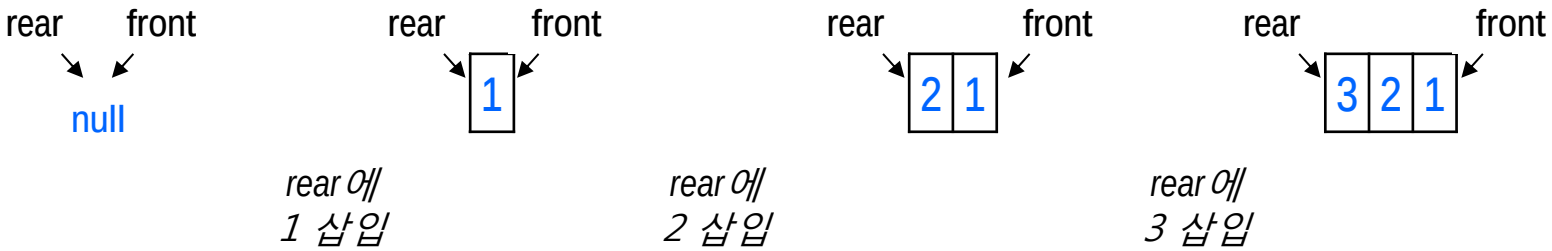
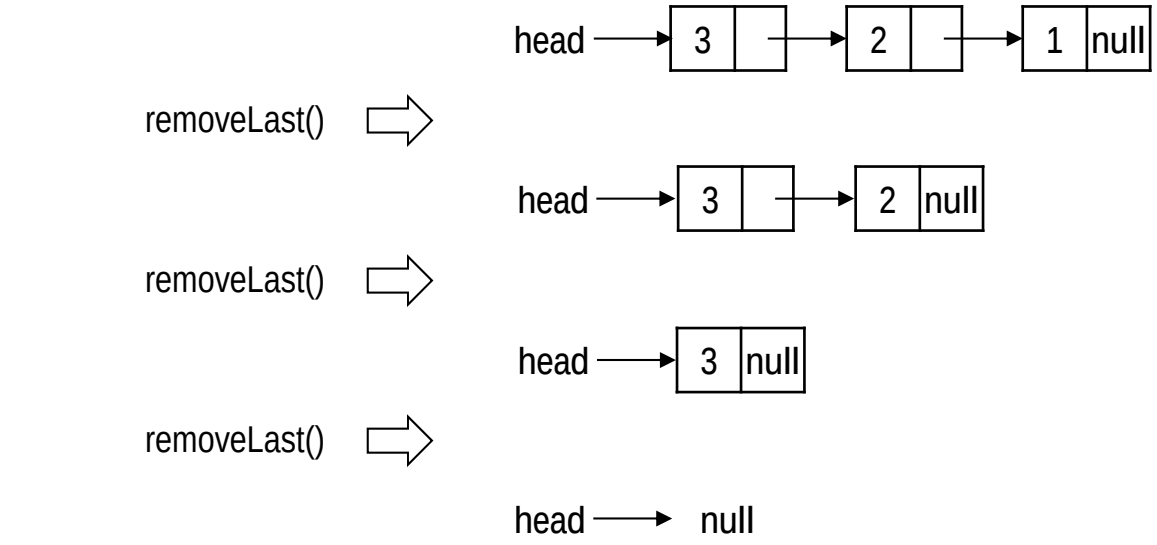
큐

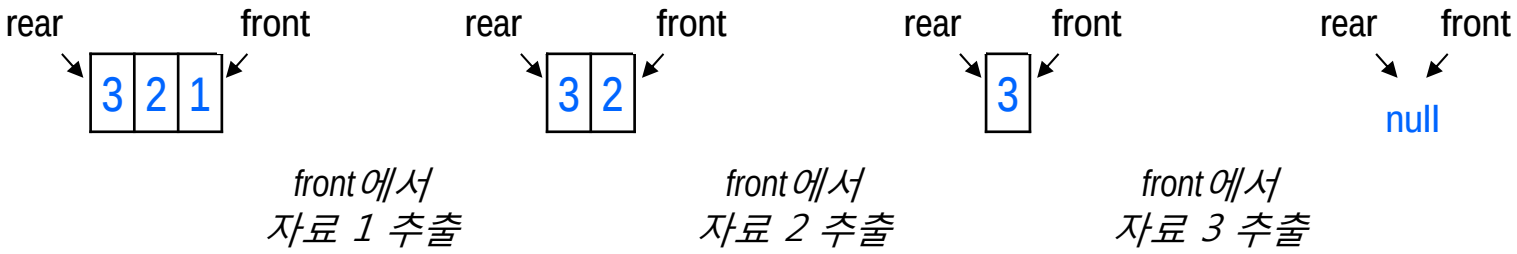
연결리스트와 큐


head
null
addFirst(1)
head
1
null
addFirst(2)
head
2
1
null
addFirst(3)
head
3
2
1
null


rear
front
null
rear
front
1
rear 에
1 삽입
rear
front
2
1
rear 에
2 삽입
rear
front
3
2
1
rear 에
3 삽입

연결리스트와 스택


head
3
2
1
null
removeLast()
head
3
2
null
removeLast()
head
3
null
removeLast()
head
null


rear
front
3 2 1
front 에서
자료 1 추출
rear
front
3 2
front 에서
자료 2 추출
rear
front
3
front 에서
자료 3 추출
rear
front
null

큐

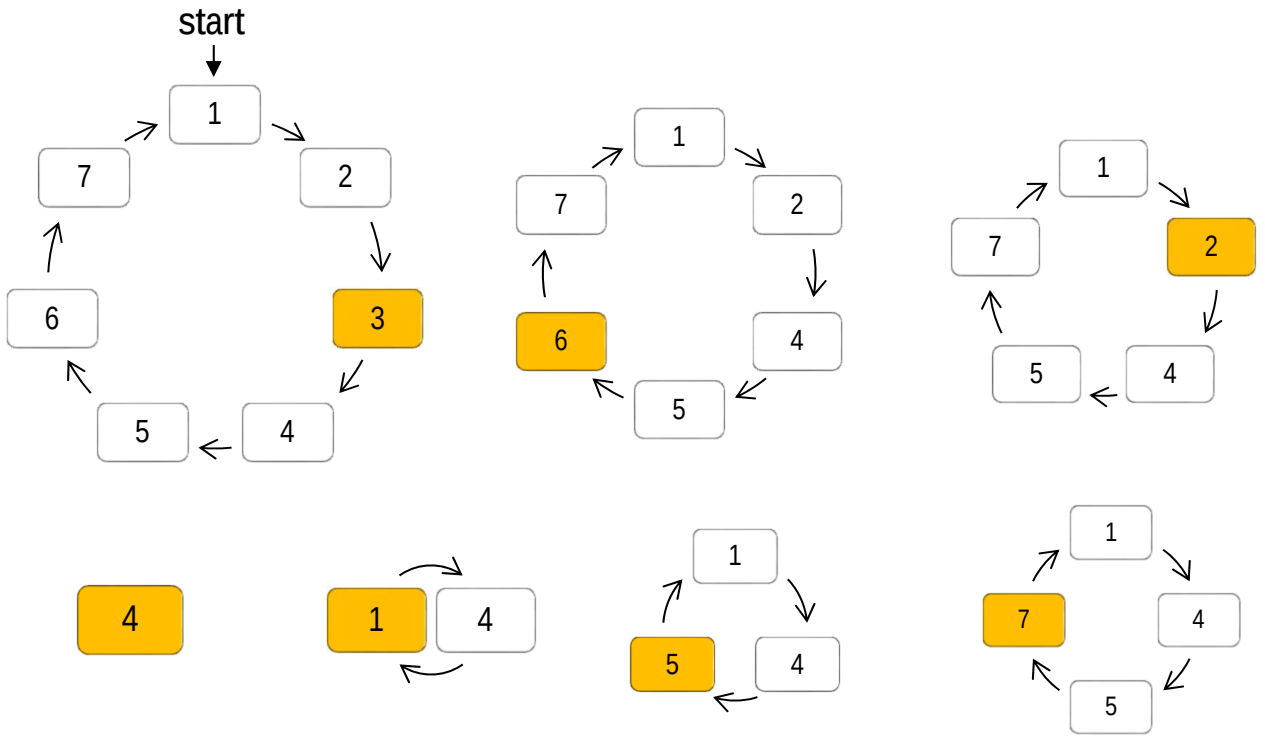
- 큐(queue)
  - 자료의 삽입과 삭제가 서로 다른 양쪽 끝에서 발생하는 리스트
  - 삽입, 삭제가 발생하는 위치를 각각 스택의 rear, front라고 부름
- 큐의 연산
  - enqueue
    - 큐의 rear에 새로운 자료를 삽입
  - dequeue
    - 큐의 front에 있는 자료를 삭제 (및 반환)
  - peek
    - 큐의 front에 있는 자료를 삭제하지 않고 반환
  - isEmpty
    - 큐가 비어 있는지 여부를 반환
  - isFull
    - 큐가 포화 상태인지 여부를 반환

Reference: https://en.wikipedia.org/wiki/Queue_(abstract_data_type)

큐 연습: 요세푸스 문제

Reference: https://ko.wikipedia.org/wiki/요세푸스_문제

- 요세푸스 문제
  - 원형 모임 n명에 대해 특정 위치부터 시작하여 k번째 사람을 제외하는 작업을 반복하여 최후 제외되는 사람의 최초 원형 위치 결정
  - 예
    - n=7, k=3



큐 연습: 요세푸스 문제

Reference:
- https://ko.wikipedia.org/wiki/요세푸스_문제
- https://rosettacode.org/wiki/Josephus_problem

- 요세푸스 문제 해결 (n=7, k=3)
  - 1 2 3 4 5 6 7 (최초 n개 자료 큐 삽입)
  - 3 4 5 6 7 1 2 (k-1개 자료 큐에서 삭제 후 다시 큐에 추가)
  - 4 5 6 7 1 2 (큐에서 자료 하나 삭제)
  - 6 7 1 2 4 5 (k-1개 자료 큐에서 삭제 후 다시 큐에 추가)
  - 7 1 2 4 5 (큐에서 자료 하나 삭제)
  - 2 4 5 7 1 (k-1개 자료 큐에서 삭제 후 다시 큐에 추가)
  - 4 5 7 1 (큐에서 자료 하나 삭제)
  - 7 1 4 5 (k-1개 자료 큐에서 삭제 후 다시 큐에 추가)
  - 1 4 5 (큐에서 자료 하나 삭제)
  - 5 1 4 (k-1개 자료 큐에서 삭제 후 다시 큐에 추가)
  - 1 4 (큐에서 자료 하나 삭제)
  - 1 4 (k-1개 자료 큐에서 삭제 후 다시 큐에 추가)
  - 4 (큐 크기 1)

큐 연습: 요세푸스 문제

Reference:
- https://ko.wikipedia.org/wiki/요세푸스_문제
- https://rosettacode.org/wiki/Josephus_problem

- 요세푸스 문제 해결 (n=7, k=3)
  - 1~n까지 노드로 구성된 연결리스트 생성
    - 1→2→3→4→5→6→7
  - 이 리스트를 원형리스트(끝 노드 다음이 첫 노드)라 가정하고
    - 첫 노드부터 시작하여 k번째 노드 삭제하는 작업을 리스트 크기가 1이 될 때까지 반복
    - 1→2→3→4→5→6→7
    - 1→2→4→5→6→7
    - 1→2→4→5→7
    - 1→4→5→7
    - 1→4→5
    - 1→4
    - 4

큐 연습: 요세푸스 문제

Reference:
- https://ko.wikipedia.org/wiki/요세푸스_문제
- https://rosettacode.org/wiki/Josephus_problem

- 요세푸스 문제 해결 (n=7, k=3)
  - 점화식(위치 번호 0부터 시작하는 것으로 가정)
    - G(1, k) = 0
    - g(n, k) = (g(n-1,k) + k) % n

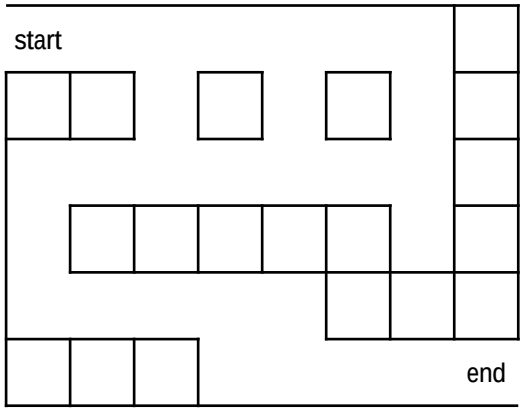
큐 응용: 미로 탐색

Reference: 프로그래밍 콘테스트 챌린징, Akiba 등 공저, 로드북, 2011, p51~53

미로 완전 탐색 (BFS)

① 시작 위치 (0,0)를 방문 표시(v) 후 큐에 삽입

② 공백 큐가 아닌 동안, 큐에서 위치 추출 후, 그 위치의 상하좌우 위치 중 이동 가능한 미방문 위치를 방문 표시 (v) 후 큐에 삽입하는 절차를 반복



v000000#	vv00000#	vvv0000#	vvvv000#	vvvvv00#	vvvvv00#	vvvvvv0#	vvvvvvv#
##0#0#0#	##0#0#0#	##0#0#0#	##v#0#0#	##v#0#0#	##v#0#0#	##v#v#0#	##v#v#v#
0000000#	0000000#	0000000#	0000000#	0000000#	00v0000#	00v0000#	vvvvvvv#
0#####0#	0#####0#	0#####0#	0#####0#	0#####0#	0#####0#	0#####0#	v#####v#
00000###	00000###	00000###	00000###	00000###	00000###	00000###	vvvvv###
###00000	###00000	###00000	###00000	###00000	###00000	###00000	###vvvvv
(0,0)	(0,1)	(0,2)	(0,3) , (1,2)	(1,2) , (0,4)	(0,4) , (2,2)	(2,2) , (0,5) , (1,4)	(5,7)

큐 응용: 미로 탐색

Reference: 프로그래밍 콘테스트 챌린징, Akiba 등 공저, 로드북, 2011, p51~53

- 미로 탐색 문제
  - 미로 탈출 경로 존재 여부 결정
    - BFS 방문 후 출구 위치가 방문 표시되어 있는가?
  - 미로 탈출 최단 거리 계산
    - BFS 방문 시 새로운 방문 위치까지의 거리는 직전 위치까지 거리 + 1
  - 미로 탈출 최단 경로 결정

배열 기반 큐 구현

- 배열 기반 큐 구현
  - rear는 큐에 삽입된 가장 최근 자료의 위치이다. (초기값 -1)
  - front는 큐에 삽입된 가장 오래된 자료 직전 위치이다. (초기값 -1)
  - 큐에 자료 삽입 시 rear를 1 증가한 후 rear 위치에 자료 저장
  - 큐에서 자료 추출 시 front를 1 증가한 후 front 위치 자료 추출
  - 공백 큐 판단
    - rear와 front의 값이 같으면 큐가 비어 있는 상태
  - 포화 큐 판단
    - rear가 최대 큐 크기보다 1 적은 값이면 큐 포화 상태
  - 큐 포화 시 front가 -1을 초과하는 경우 여분의 공간 확보 위해 배열 내 전체 자료를 왼쪽으로 이동시키고 front, rear 값 재설정 필요

배열 기반 큐 클래스 구현

```
public class SimpleQueue {
    int    rear=-1, front=-1, MaxSize=5;
    char queue[];
    public SimpleQueue() { queue=new char[MaxSize]; }
    public void add(char data) {
        if(full()) throw new RuntimeException("queue full");
        queue[++rear]=data;
    }
    public int remove() {
        if(empty()) throw new RuntimeException("queue full");
        return queue[++front];
    }
    private boolean full() { return rear==MaxSize-1; }
    private boolean empty() { return rear==front; }
    @Override
    public String toString() {
        return "front="+front+", rear="+rear+", "+Arrays.toString(queue);
    }
}
```

References

- C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 1993.
- 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데미. 2013.
- C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.
- 프로그래밍 콘테스트 챌린징, Akiba 등 공저, 로드북, 2011.
- https://introcs.cs.princeton.edu/java/assignments/
- https://en.wikipedia.org/wiki/Queue_(abstract_data_type)
- https://ko.wikipedia.org/wiki/요세푸스_문제
- https://rosettacode.org/wiki/Josephus_problem
- Introduction to Algorithms, Cormen et al., 3rd Edition (The MIT Press)