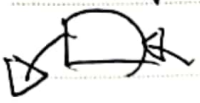
NO.
year month day ( )

큐 FIFO



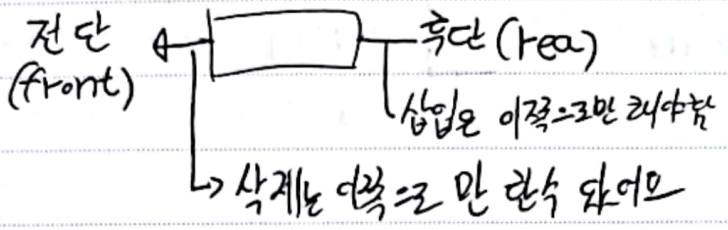
DFS(스택)

BFS(큐)

5. 큐와 덱

큐

- 삽입과 삭제가 양 끝에서 각각 수행되는 자료구조
- 선입선출의 자료구조
  → 먼저 들어온 데이터가 먼저 나가는 구조
- 새로운 데이터가 추가되고 앞에서 데이터가 하나씩 삭제되는 구조
- 삽입이 일어나는 곳을 후단이라고 하고 삭제가 일어나는 곳을 전단이라고한다.



ADT

데이터와 선입선출(FIFO)의 접근 방법을 위해서 필요한 항목들의 모음

주요연산

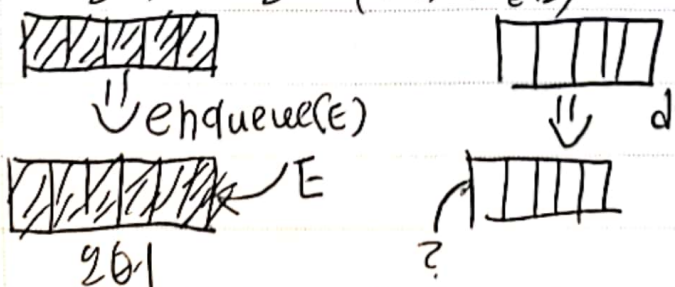
- isEmpty()
- enqueue(x) 맨뒤에 추가
- dequeue() 맨앞의 항목을 꺼내 반환

연산

- 삽입연산 enqueue() : 삽입은 후단을 통해서만 가능
- 삭제연산 dequeue() : 삭제는 전단을 통해서만 가능

고려할 점

오버플로, 언더플로 (스택과 같음)



큐의 사용

ㅇ 임시기억장치로 큐가 사용되며, 이를 버퍼라고 함

ㅇ 컴퓨터와 프린터사이: 프린터는 CPU에 비해 상대적으로 속도가 느림

ㅇ 실시간 비디오 스트리밍: 다운로드 된 데이터가 비디오를 재생하기에 충분하지 않을때 사용

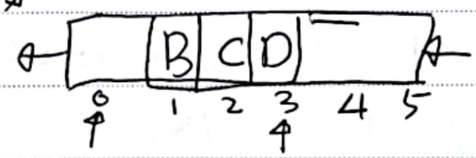
배열
연결리스트) 더 좋음

큐의 구조(배열)

용량이 고정됨

array[capacity] → 용량

↓ 요소들을 저장할 배열



응용 예)
ㅇ 이진트리의 레벨순회
ㅇ 기수 정렬에서 레코드 정렬을 위해
ㅇ 그래프의 너비우선탐색

전단요소 바로 앞의 위치 ← front   rear ← 맨 마지막 요소의 위치

ㅇ front → 첫 번째 요소 바로 앞의 위치

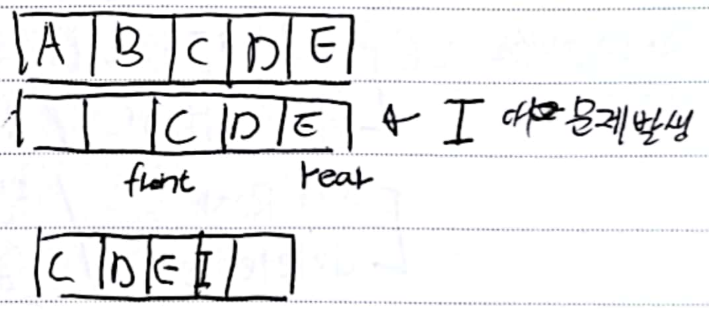
ㅇ rear → 맨 마지막 요소의 위치

문제점

많은 이동이 필요한 경우발생

후단에 빈공간이 없어 바로 삽입불가 →

모두 앞으로 이동해야 한다 →



개선 → 원형큐

전단회전
front ← (front+1)%cap

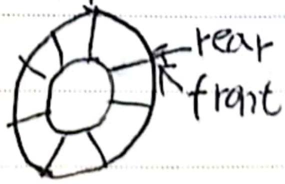
후단회전
rear ← (rear+1)%cap

ㅇ 선형 큐의 비효율성 보완

ㅇ 리스트에 항상 고정된 크기(MAX_SIZE)가 있다고 생각

ㅇ rear: 큐에 가장 최근에 삽입된 항목의 위치를 저장

ㅇ front: 가장 최근에 삭제된 항목의 위치를 저장



공백과 포화

ㅇ 공백상태 front == rear

ㅇ 포화상태: front == (rear+1)%MAX_SQSIZE

파이썬 queue 모듈

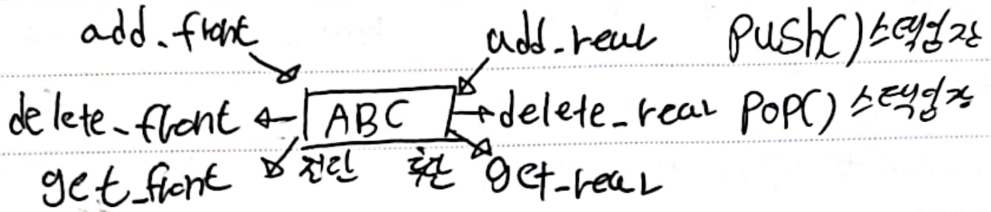
덱= 큐+스택

- 큐와 스택 클래스를 제공
- 사용하기 위해서는 먼저 queue 모듈을 import 해야함
- 삽입(put()) /삭제(get()) 연산 존재

덱

- 덱은 Double-Ended QUEue 의 줄임말
- 스택이나 큐보다는 입출력이 자유로운 구조
- 양쪽 끝에서 삽입과 삭제가 모두 가능한 구조
- 중간 삽입 or 삭제 불가능

덱에서는 삽입,삭제 양쪽에서 가능 그렇지만 중간에 넣거나 빼지는 못함



데이터 → 전단과 후단을 통한 접근을 허용하는 항목들의 모임

그리고 큐나 스택과 비슷한 연산이 많음

연산 [add Front, delete Front] / 스택과 큐의 연산들과 유사함
Rear과 front 반대방향 회전 연산 추가

[add Rear, delete Rear] / 원형큐를 상속하여 원형 덱 클래스를 구현
생성자는 상속되지 않음

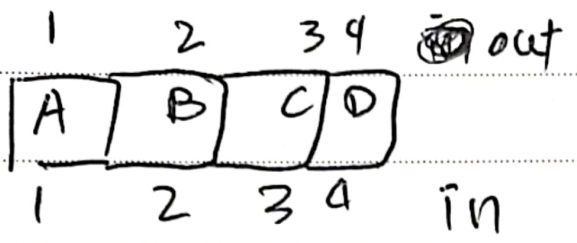
우선순위큐

- 우선순위의 개념을 큐에 도입한 자료구조
- 모든 데이터가 우선순위를 가짐
- 입력순서와 상관없이 우선순위가 높은 데이터가 먼저 출력
- 가장 일반적인 큐로 볼수 있음

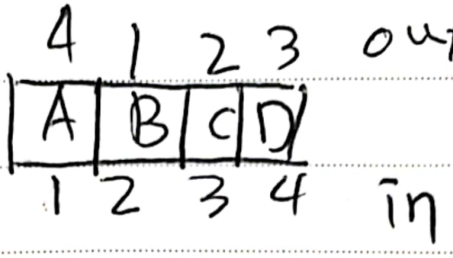
→ 우선순위 정하는 방식에 따라서 스택이나 큐로도 사용가능

응용분야 → 시뮬레이션, 네트워크 트래픽 제어, OS의 작업 스케줄링

일반적인 큐

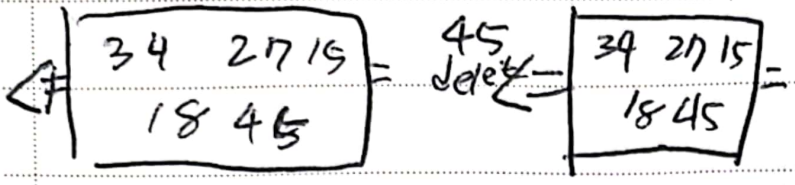


우선순위 큐



→ 전단과 후단을 통한 접근을 허용한 항목들의 모델

구현방법 → 배열 구조, 연결된 구조, 힙 트리 등



시간복잡도

정렬되지 않은 리스트 사용

- enqueue(): 대부분의 경우 O(1)
- findMaxIndex(): O(n)
- dequeue(), peek(): O(n)

정렬된 리스트 사용

- enqueue(): O(n)
- dequeue(), peek(): O(1)

힙 트리

- enqueue(), dequeue(): O(log n)
- peek(): O(1)