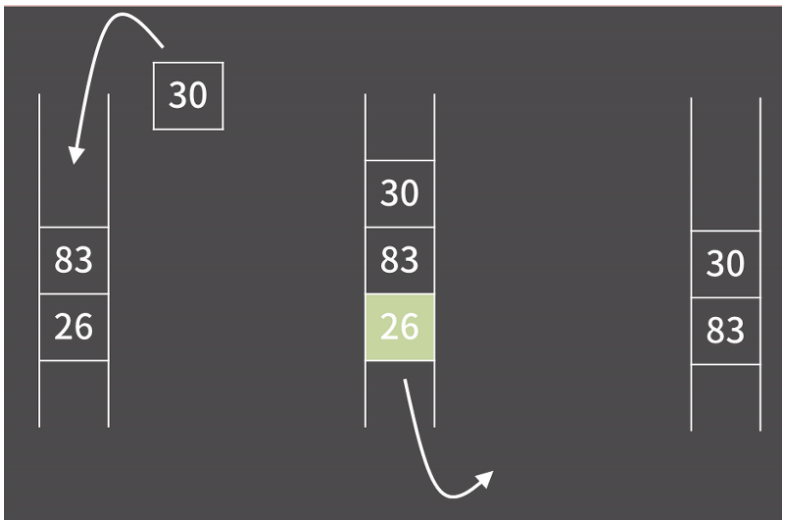
큐

강의 출처: 바킹독 실전 알고리즘 (https://blog.encrypted.gg/922?category=773649)

1. 정의와 성질



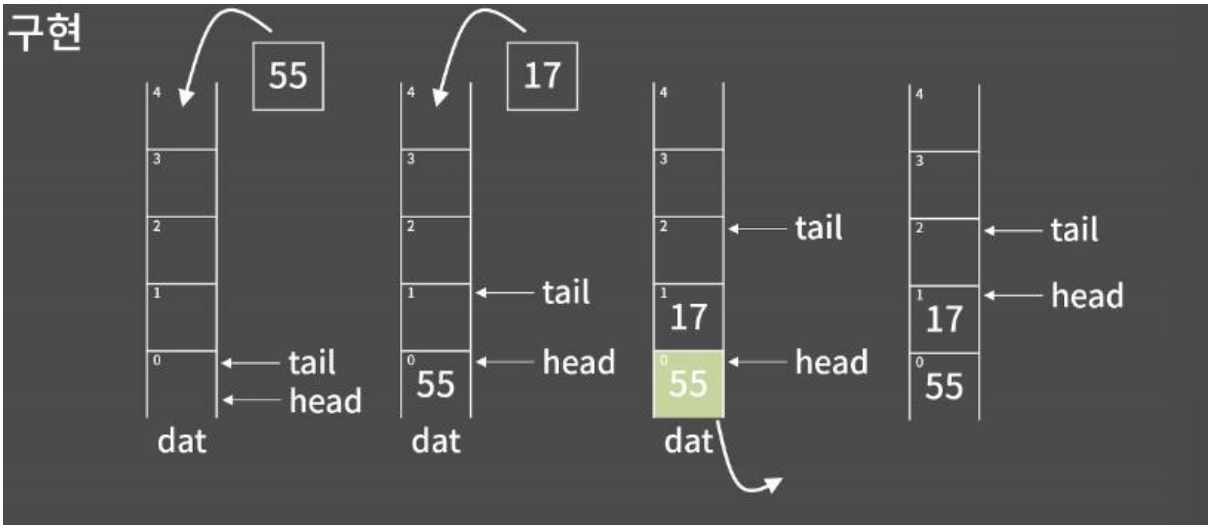
큐는 한쪽 끝에서 원소를 넣고 반대쪽 끝에서 원소를 뺄 수 있는 자료구조입니다. 스택에서는 먼저 들어간 원소가 나중에 나왔는데 큐에서는 먼저 들어간 원소가
먼저 나오게 됩니다. 공항에서 입국수속을 하는 줄과 같은 상황이라고 볼 수도 있습니다.

스택을 FILO(First In Last Out)이라고 한 것과 비슷하게 큐를 FIFO(First in First Out)이라고 하기도 합니다.

1. 원소의 추가가 O(1)
2. 원소의 제거가 O(1)
3. 제일 앞/뒤의 원소 확인이 O(1)
4. 제일 앞/뒤가 아닌 나머지 원소들의 확인/변경이 원칙적으로 불가능

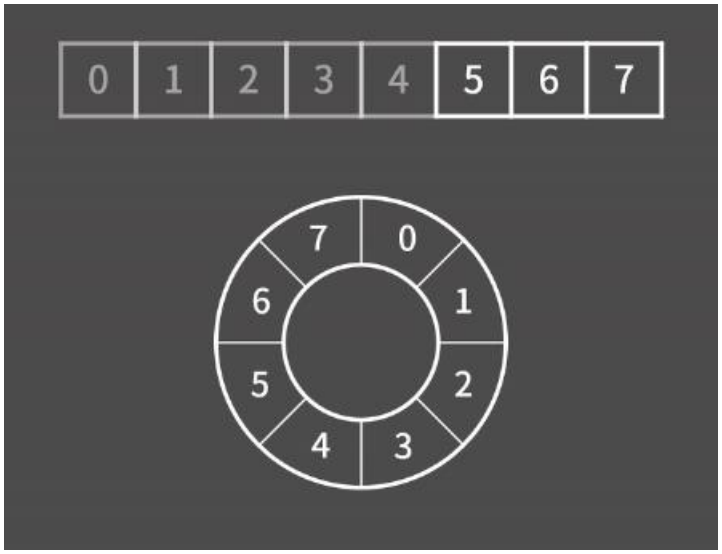
2. 구현

큐를 구현할 때는 원소를 담을 큰 배열 한 개와 앞쪽, 뒤쪽을 가리킬 변수 두 개가 필요합니다. 이 head와 tail을 어떻게 두는지는 예시를 보고 이해해보겠습니다.



이번에는 head가 1칸 올라갔습니다. 0번지에 있는 55는 굳이 다른 값으로 덮을 필요는 없습니다.

이런 식으로 head와 tail은 0번지에서 시작해 계속 증가하게 됩니다. 쭉 보셔서 알겠지만 dat 배열에서 dat[head]부터 dat[tail-1]번지가 바로 큐의 원소들이 들어있는 자리입니다. 그리고 큐의 크기는 tail - head로 쉽게 계산할 수 있습니다.



앞의 과정을 보면 push를 할 땐 tail이 증가하고, pop을 할 땐 head가 증가합니다. 그렇기 때문에 dat 배열에서 큐의 원소들이 들어있는 장소는 점점 오른쪽으로 밀립니다.

head가 5를 가리키고 있고 tail이 8을 가리키고 있어 dat의 5번지, 6번지, 7번지를 사용하고 있는 상황을 보면 앞쪽에 사용하지 않고 있는 공간이 많음에도 불구하고 더 이상 삽입을 할 수가 없습니다. 삽입을 하려면 dat[8]에 값을 써야 하는데 배열이 8칸이니 그럴 수가 없기 때문입니다.

스택과는 다르게 큐는 배열로 구현하면 삭제가 발생할 때 마다 앞쪽에 쓸모없는 공간이 계속 생기게 됩니다. 이 문제를 해결하는 방법은 의외로 간단한데 큐의 원소가 들어갈 배열을 원형으로 만드는 것입니다. 관념적으로는 배열이 원형인거고, 실제 구현을 할 땐 head나 tail이 7인 상태에서 1이 더해질 때 0번지로 다시 오도록 만들면 됩니다.

즉, dat의 5, 6, 7번지를 사용하는 상황에서 원소 1개가 추가되면 0번지를 점유하는 것입니다. 이렇게 원형의 배열을 가정하고 구현한 큐를 원형 큐(Circular Queue)라고 부릅니다.

3. STL 큐

reference : http://www.cplusplus.com/reference/queue/queue/

https://github.com/blisstoner/basic-algo-lecture-metarial/blob/master/0x06/queuee_example.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(void) {
  queue<int> Q;
  Q.push(10); // 10
  Q.push(20); // 10 20
  Q.push(30); // 10 20 30
  cout << Q.size() << '\n'; // 3
  if(Q.empty()) cout << "Q is empty\n";
  else cout << "Q is not empty\n"; // Q is not empty
  Q.pop(); // 20 30
  cout << Q.front() << '\n'; // 20
  cout << Q.back() << '\n'; // 30
  Q.push(40); // 20 30 40
  Q.pop(); // 30 40
  cout << Q.front() << '\n'; // 30
}
```

보통 큐는 BFS 랑 Flood Fill 를 할 때 쓰게 되는데 둘 다 코딩테스트 단골 문제여서 문제를 풀 때 STL queue 를 쓸 일이 아주 많을 것입니다. 그래서 나중 가면 적어도 STL queue 만큼은 외우기 싫어도 외워질 것입니다.

그리고 큐가 비어있는데 front 나 back 이나 pop 을 호출하면 런타임에러가 발생할 수 있다는 점은 주의를 하셔야 합니다.