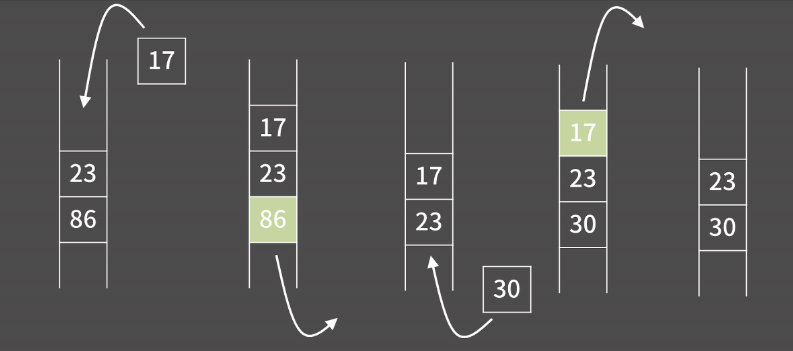
덱

1. 정의와 성질

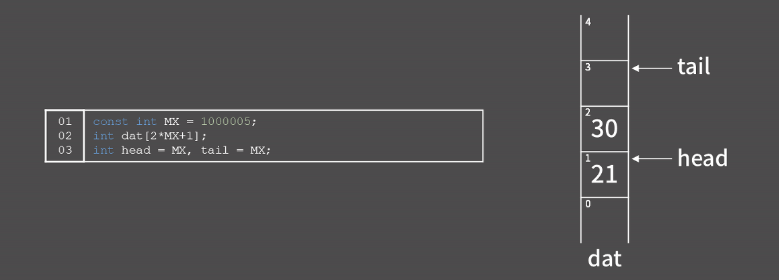
덱은 Restricted Structure의 끝판왕과 같은 느낌의 자료구조인데, 양쪽 끝에서 삽입과 삭제가 전부 가능합니다. 참고로 자료구조의 덱은 deque고 Double Ended Queue라는 뜻을 가지고 있습니다.

아무튼 덱은 양쪽 끝에서 삽입과 삭제가 전부 가능한 자료구조이니 스택과 큐를 덱의 특수한 예시라고 생각해도 괜찮겠습니다.

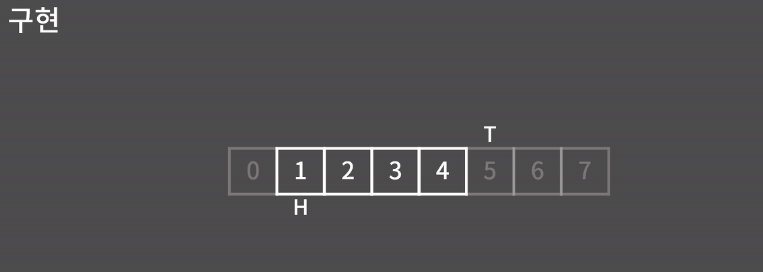
텍스트이(가) 표시된 사진

자동 생성된 설명

1. 구현



일단 필요한 변수는 큐랑 똑같이 원소를 담을 큰 배열 한 개와 앞쪽, 뒤쪽을 가리킬 변수 두 개입니다. 이 때 head와 tail을 두는 방식도 큐와 똑같습니다. head는 가장 앞에 있는 원소의 인덱스이고 tail을 가장 뒤에 있는 원소의 인덱스 + 1입니다. 그리고 head와 tail의 초기값이 0이 아니라 MX인데 이 부분을 좀 짚고 넘어가겠습니다.



 덱에서는 양쪽에서 모두 삽입이 가능합니다. 그렇기 때문에 마치 여의봉처럼 양쪽으로 확장해야 합니다. 그러면 별 생각 없이 시작 지점이 0번지로 뒀을 경우 왼쪽으로 확장을 할 수가 없게 됩니다. 대신에 시작 지점을 배열의 중간으로 둬야 합니다. 중간으로 두면 중앙에서 양쪽으로 확장이 가능합니다. 그래서 배열의 크기는 2\*MX+1이고 head와 tail의 초기값은 MX인 것입니다.

Stl 덱

텍스트이(가) 표시된 사진

자동 생성된 설명

물론 STL에 덱이 있어서 그냥 가져다쓰면 되는데, STL deque은 매우 독특하게도 double ended queue라는 느낌보다는 vector랑 비슷한데 front에서도 O(1)에 추가와 제거가 가능한 느낌이 강합니다. pop\_front, push\_front, pop\_back, push\_front 함수야 당연히 덱이니 있어야 정상인데, 이외에도 19-25번째 줄을 보면 insert도 있고 erase도 있고 인덱스로 원소에 접근도 할 수 있습니다.

이와 같이 STL vector에서 제공되는 기능을 STL deque에서도 다 제공해주고 심지어 deque은 front에서도 O(1)에 추가와 제거가 가능하니 deque이 vector보다 상위호환이 아닌가 하는 생각이 들 수 있겠지만, vector와 달리 deque은 모든 원소들이 메모리 상에 연속하게 배치되어 있지 않습니다. 구조에 대해 자세히 설명을 드리고 싶지만 강의의 난이도를 많이 벗어나는 내용인 것 같아서 궁금하시다면 스스로 c++ deque vs vector와 같은 검색으로 찾아보시면 되겠습니다.

다만 어떤 것 하나만 알고 가시면 되냐면, 앞쪽과 뒷쪽에서의 추가와 제거가 모두 필요하면 당연히 STL deque을 사용하는게 효율적이지만 굳이 앞쪽에서의 추가와 제거가 필요하지 않고 배열과 같은 느낌으로 쓰고싶을 땐 STL deque말고 STL vector를 쓰면 됩니다.

vector와 deque와의 큰 차이점 중 하나가, 연속의 유무라서  
vector의 경우 공간이 부족하면, memory reallocate 과정을 거쳐야 하는데  
deque의 경우 연속되지 않으니, 그냥 새로운 memory block 을 하나 할당하면 되니 평균적인 성능을 보장하네요.  
  
그래서 그런지 deque에는 capacity와 reserve 가 존재하지 않군요.  
  
또 원소에 어느방향이던 "iterate"은 가능하나, 연속적이지 않으니 당연히 vector간 포인터간 연산은 불가능할꺼구요.  
  
그리고 중간에서의 삽입 삭제 연산은 list 에 비해서 굉장히 구리네요.

deque이 vector/list에 비해 컨테이너 생성에 시간이 좀 많이 오래걸려요. 1000만번 생성하는 코드를 작성해보면 시간차이가 확실히 많이 나요. 그래서 문제풀 때 이 점을 조금 주의해야해요.