Lecture 7 : 스택(Stack)과 큐(Queue) - 기초

스택(stack)은 큐(queue)와 함께 탐색(searching)에서 가장 일반적으로 응용되는 대표적인 자료구조이다. 특히 그래프 관련 알고리즘을 구상할 때에는 이 둘 중 하나를, 또는 둘을 결합하여[1] 사용한다. Stack은 linear container(리스트나 array, vector)에 뭔가의 추가 장치를 더한 구조, 즉 adaptor 장치로 이해하면 좋다. 여러분은 바로 이 adaptor라는 개념에 주의해야 한다. 즉 container와 같이 어떤 자료를 저장하고 탐색하는 것이라기보다 특정한 동작만 가능한 자료구조라는 것이다. 왜 평범한 선형 컨테이너에 이런 제한을 두는가 하는 것에 의문을 가질 것이지만 이런 제한이 프로그래밍을 더 편하고 안전하게 도와준다는 것을 이해해야 한다. 적절한 제한은 시스템의 효율을 높여주는 주요한 장치이다. 예를 들어 쓰레기 처리를 할 때 특정한 요일에 특정한 쓰레기만 내놓을 수 있도록 제한하는 아파트를 흔히 볼 수 있는데 이런 제한을 가함으로써 쓰레기 처리 작업을 훨씬 더 효율적으로 할 수 있다.

이렇게 linear container에 제한을 가하는 방법에 따라서 stack과 queue, deque 또는 priority queue로 구분될 수 있으며 또는 응용문제에 따라서 새로운 adaptor 구조를 만들 수 있다. 예를 들면 양쪽으로 stack push pop 작업을 할 수 있는 double ended stack도 문제에 따라서는 활용할 수 있다.

스택과 큐는 매우 단순한 구조라 이것 자체에서 특별히 깊게 연구할 내용은 별로 없다. 중요한 것은 stack과 queue가 응용되는 다양한 문제를 접해보고 실제 구현해보는 것이다. stack과 queue는 이후 graph algorithm에서 DFS(Depth First Search)와 BFS(Breadth First Search)의 기본 자료구조가 되며 이 두 그래프 탐색 방법은 이후 Searching Model 중 대표적인 backtracking과 branch and bound 알고리즘의 핵심 방법론이 된다. 여러분은 이 연결된 구조를 항상 염두에 두고 stack과 queue를 다루어야 한다.

일반적인 많은 자료구조 교재에서는 stack과 queue를 직접 배열이나 리스트를 이용하여 구현하는 방법을 제시하고 있으나 이는 바람직하지 못한 형태이다. 특히 배열을 이용한 Stack과 Queue는 Overflow 문제에서 심각한 상황을 만들어 낼 수 있으므로 가능한 수제(?) stack과 queue는 현장에서 사용하지 않아야 한다.[2] 앞서 여러 번 언급하였지만, 자료구조를 직접 구현하는 것은 오류의 위험성 compatability과 같은 여러 문제를 일으킬 수 있으므로 매우 유의해야 한다. STL에서는 기본적인 stack과 queue, deque를 제공하고 있으며 또한 그 구현 방법까지 제공하고 있다. 즉 스택을 vector를 기본으로도 구현할 수 있고 또는 linked list로도 구현할 수 있다.[3]

7.1 스택(Stack)과 큐(Queue): 정의

Q) 코로나 시절이라 밖으로 나갈 때는 항상 마스크를 사용해야 한다. 마스크와 함께 안

---

1) 구성하고 활용하기에 복잡하지만 특별한 성능이 요구되는 고급자료구조에서 시도되는 방법이다.

2) 어떤 교재에 보면 이를 방지하기 위하여 환형 stack이나 queue를 제안하고 있는데 전혀 쓸데없는 짓이다. 이러한 circular 구조는 프로그램의 복잡도를 올릴 뿐만 아니라 크고 작은 오류의 원천이 된다.

3) 만일 stack이나 queue에 들어갈 원소의 개수가 문제가 되는 경우라면 list implementation으로 구현하는 것이 안정적인 시스템을 만들 수 있는 기본이 된다.

경, 이어폰을 사용하는 사람이 있다. 보통의 경우 안경을 쓰고, 이어폰을 꽂고 마지막에 마스크를 쓴다. 만일 이 사람이 다시 집으로 돌아왔다면 어떤 순서로 풀어야 제대로 {이어폰, 마스크, 안경}을 벗을 것인지 그 순서를 제시해보라.

- 일반적으로 살찔 때의 순서는 다음과 같다. < 뱃살, 허리, 다리, 팔, 얼굴 >
- 그러나 살이 빠지는 순서는 그 반대로 <얼굴, 팔, 다리, 허리, 뱃살 >이라고 한다.

이 과정을 stack을 이용해서 설명하시오.

Q) 스택이란 무엇인가?
A) push와 pop을 지원하는 선형구조입니다.
push는 스택에 하나의 자료를 추가하는 동작이며,
pop은 가장 최근에 들어간 자료를 선택하여 제거하는 동작입니다.

따라서 이 동작이 되도록 내부를 구현하면 된다. <u>어떻게 구현해도 상관없음</u>

- Linear 구조
- 입력과 출력은 특정하게 한정된 위치에서만 가능하게 함
- 기본함수 = { 생성, Push(S, data), Pop(S) }
- 보조함수 = { Is_Empty( ) , Is_Overflow( ), }

- 추상적 스택과 구체적 스택
- 추상형 자료구조와 구현 (implementation) 상의 차이를 이해
  eg) 추상적인 극장과 구체적인 O2 cinema의 차이점

7.2 STL에서 컨테이너와 어댑터(adaptor, 적응자)는 어떻게 다른가?

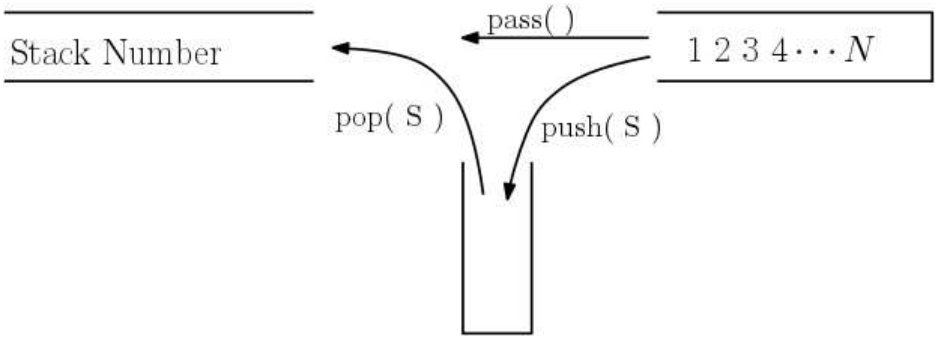
어댑터 - 어떤 자료구조에 뭔가를 “끼워서” 다른 목적에 합당하도록 사용하게 해주는 장치, 객체, 일종의 gadget이나 보조 젠더

- 컨테이너 = {Vector,Deque,List,Set,MultiSet,Map,MultiMap, }
- 어댑터 = { Stack, Queue, Priority Queue, BitSet }

7.3 Stack Sequence(스택 순서): 1개의 중간 스택을 이용하는 경우

입력 대기열이 숫자가 순서대로 1부터 N까지 대기하고 있다. 이 숫자는 중간의 대기 Stack으로 들어가거나 바로 output 줄로 나갈 수 있다. 또는 Stack top에 있는 원소가 pop( ) 되어 output 줄로 나갈 수 있다. 이 동작의 순서에는 아무런 제한이 없다.

위에서 설명한 동작을 실행했을 때 최종 output에 나타나는 수열을 Stack 수열, 또는 순서라고 한다. 예를 들어 N=5일 때 1,2,3,4,5 Stack Sequence이지만 1,5,2,3,4는 어떻게 해도 output 줄에 나타날 수 없는 순서이므로 이것은 stack sequence가 될 수 없다. 여러분은 주어진 순서가 Stack Sequence인지의 여부를 판단해야 한다.



다음 주 Stack Number가 아닌 것은 ? 그것을 알아내는 방법은 ?

a. 4 3 2 7 5 6 1
b. 7 5 6 4 3 1 2
c. 7 6 5 4 3 2 1
d. 1 2 3 7 6 5 4
e. 3 2 7 1 6 5 4
f. 4 5 6 7 3 1 2

7.4 만일 중간에 준비된 Stack이 1개가 아닌 2개인 경우를 생각해보자. 즉 입력 대기 중인 숫자는 Stack1, 또는 Stack2 중 아무 곳이나 들어갈 수 있다. 또는 출력 줄에는 Stack1 또는 Stack2의 top 원소 중 어떤 것이라도 들어갈 수 있다. 이렇게 해서 최종 output 줄(queue)에 생성되는 순서를 2-stack sequence라고 한다. N=6일인 경우에, 다음 순서 중에서 2-stack number인지의 여부를 밝히시오.

- [3, 4, 1, 6, 2, 5]
- [6, 5, 4, 1, 2, 3]
- [1, 6, 2, 5, 3, 4]
- [6, 2, 5, 3, 4, 1]
- [3, 1, 4, 6, 2, 5]

7.5 k-Stack Number 문제, 아랫단에 k개의 stack이 있다. 왼쪽 입력은 이 중 하나의 stack에 들어가거나 pass 한다. 또는 k개의 Stack 중에서 하나의 top 원소가 왼쪽으로 옮겨갈 수 있다. 이 문제를 위 7.3에 적용해보자.

7.5 STL 스택의 실제 구현구조를 지정하는 방법. (구현되는 형식을 사용자가 지정할 수 있다.)

- C/C++ int x[1000], top ;
- STL

stack <T>	default, deque가 사용됨
stack <T, vector<T> >	vector가 기본구조
stack <T, list<T> >	list가 기본
stack <T, dequeue<T> >	default와 동일함

7.6 STL에서 제공하는 STL basic function

a. void push(const T& x); 스택의 상단에 인수로 추가할 값을 전달. 기억 장소가 무한해서 추가는 항상 성공하므로 리턴(return)되는 데이터는 없다(주의!).

b. void pop( ); pop은 리턴 값이 없으므로 이런 에러를 처리할 수 없다

c. value_type& top( ); d. const value_type& top( ) const;

Stack의 recursive structure를 표현하고 계산하기 위한 전형적인 자료구조이다. Stack을 이용하여 recursion을 직접 제어하면 시스템을 사용하는 recursion에 비하여 효율을 훨씬 더 높일 수 있으며 debugging도 용이해진다. 그러나 재귀적 코딩에 수반하는 어려움 역시 감수해야 한다.

7.7 Permutation 출력하기 : S= { a, g, t, c }를 이용해서 모든 4-mer 출력하기
4개의 for loop을 이용하는 경우 - stupid

```
char s[4]={'a', 'c', 'g', 't'} ;

for ( i=1 ; i < N ; i++ ) {
    for ( j=1 ; j < N ; j++ ) {
        for ( k=1 ; k < N ; k++ ) {
            for ( m=1 ; m < N ; m++ ) {
                w = s[i]+s[j]+s[k]+s[m];
                print(w);
} } } }
```

만일 S에 5개의 원소가 이같이 S= { a, g, t, c, n } 존재한다면 source code 자체를 고쳐서 처리해야 한다.

S = { a, b, c, d, e } 의 모든 permutation 출력하기, 5개의 for loop ?

abcde → abced → abdec → ... → decab → decba

단 하나의 stack만을 사용하기

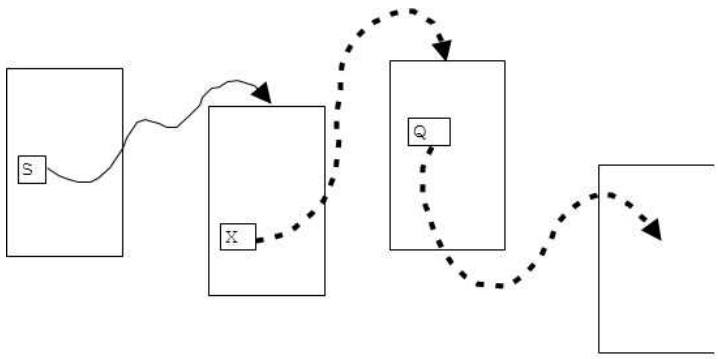
Stack을 사용하기 전에 먼제 recursion 구조를 파악해야 한다. N개 원소의 모든 permutation은 다음과 같다.

a) 우리는 1개 원소를 가진 집합의 permutation을 구할 수 있다.
b) 우리는 2개 원소를 가진 집합의 permutation을 구할 수 있다.
c) 만일 우리가 N-1개 원소의 모든 permutation을 구할 수 있다면
d) 이를 이용해서 N개 원소의 모든 permutation을 구할 수 있다.

소스 코드의 추가 수정 없이 하나의 스택만으로 해결하는 방법.

5	4	5	3	4	**3**	5	4		5	4
4	5	3	5	3	**4**	4	5		4	5
3	3	4	4	5	**5**	2	2		3	3
2	2	2	2	2	2	3	3		1	1
1	1	1	1	1	1	1	1		2	2
t=1	t=2	t=3	t=4	t=5	t=6			...		

7.8 Web-Browsing에서 Returning 하기: 돌아가야 할 페이지의 주소를 저장
(stack register를 이용하여 program interrupt를 수행함.)



		Q					
	X	X	X				
S	S	S	S	S			
t=1	*t=2*	*t=3*	*t=4*	*t=5*	*t=6*	*t=7*	*t=8*

7.9 어떤 식당에서는 정식 요리에 다음을 제공한다. 수프={배추 죽, 콩죽, 당근즙}, 메인={ 갈비, 족발, 생라면, 설렁탕, 시루떡 }, 후식={아이스크림, 마요네즈, 콜라, 들기름}. 따라서 한 식사는 <수프, 메인, 후식>으로 이루어진다. 이들의 가능한 모든 조합을 stack을 이용해서 출력하는 프로그램을 작성해보시오. for loop을 3개 "쌩으로“ 쓰면 안 됩니다.

배춧국	콩죽	당근즙		배춧국	콩죽	당근즙	
갈비	갈비	갈비	족발	족발	족발	족발	족발
아이스	아이스	아이스	아이스	아이스	아이스	아이스	아이스
t=1	*t=2*	*t=3*	*t=4*	*t=5*	*t=6*	*t=7*	*t=8*

7.10 다변수 일차방정식의 해 구하기 :

어떤 일차방정식을 만족하는 해를 모두 구하시오. 또는 몇 개의 주어진 방정식의 해를 모두 구하는 방법을 for loop을 여러 개 사용하지 말고 구해보시오.

$$11w + 3x + 4y + 7z = 100, \qquad \text{단 } 1 \le w,x,y,z$$

$$3p + 14q + 7r + 5s + 12t + u = 200$$
$$4p + 2q + 9r + 3s + 11t + 7u = 150$$

```
1               //식의 개수
4               // 변수의 개수
11 3 4 7 100
```

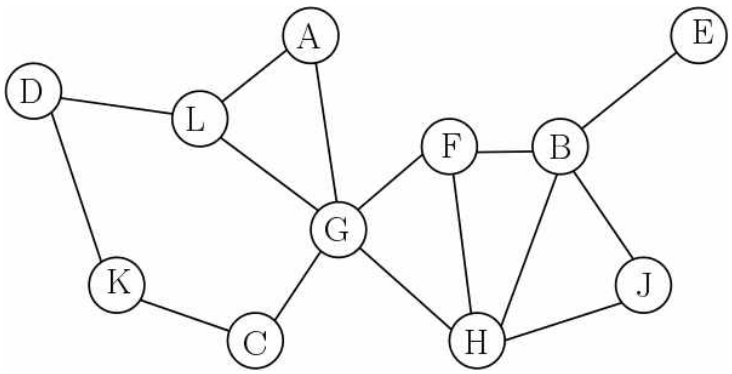
```
2               //식의 개수
6               // 변수의 개수
3 14 7 5 12 1 200
4  2 9 3 11 7 150
```

			L							
			D							
			K			B				
		C	C		F	F				
	G	G	G	G	G	G				
A	A	A	A	A	A	A				
t=1	*t=2*	*t=3*	*t=4*	*t=5*	*t=6*	*t=7*	*t=8*	*t=9*	*t=10*	*t=11*

v	A	B	C	D	E	F	G	H	J	L	K
visited( )											

7.11 그래프 탐색하기 – 가본 곳은 더 이상 가지 않고, 빠진 곳 없이 찾아보기

- 아래 그래프에서 A부터 방문할 때 그 순서를 표시하시오.
- 아래 그래프에서 G부터 방문할 때 그 순서를 표시하시오.



Stack을 이용한 DFS 방문 알고리즘

1) 시작점을 지정하고 그 위치로 간다.
2) 현재 위치에서 방문하지 않은 이웃을 선택한다. 그 지점을 x라고 하자.
3) 만일 그런 미방문 지역이 하나 이상이면 그중에서 가장 이름이 빠른 node로 진행한다. 그리고 Goto 2)
4) 만일 주위 이웃이 모두 이미 방문한 지역(node)라고 한다면 이전 방문 경로로 한 칸 back을 한다. 즉 x를 방문하기 직전에 방문한 노드를 기억했다가 다시 그 지점으로 되돌아간다. (backtrack)
5) 이렇게 하여 다시 시작점으로 되돌아오고 더 이상 방문할 지역이 없으면

탐색을 종료한다.

7.12 친구와 이야기하다 다시 원래 주제로 돌아가는 방법

1) . 학교에서 자료구조 수업을 듣는데. 과제물.....인터넷을 보다 보니
   새로 나온 신상 신발에 관한 pop-up 광고가 나옴
2) 신상 신발 이야기를 하다, 그 신발을 신고 나오는 아이돌 가수 이야기를 함.
3) 아이돌 가수 Q 이야기 중, 그 사람의 이전 연인 W 이야기가 나옴
4) W가 최근에 음주운전으로 구속되었다는 소식
5) 음주운전 처벌 법령이 너무 약하는 주장이 나옴
6) 요즘 일부 판사들이 ”개떡“같이 판결을 한다고 함.
7) 그건 판사의 문제가 아니라 국회에서 법을 제대로 못 만들어서 그렇다고 함.