

알고리즘과 코딩테스트 준비

2022/1/13~1/19

윤형기 (hky@openwith.net)

강의 소개

진행 순서

	1-13 (목)	1-14 (금)	1-17 (월)	1-18 (화)	1-19 (수)
오전 (2Hrs)	<ul style="list-style-type: none"> 강의 소개: 코딩테스트 최근 동향과 학습방법 도 구 / 언 어 Review (OOP, standard lib.) 	<ul style="list-style-type: none"> 주제(3): 탐욕알고리즘 <ul style="list-style-type: none"> Knapsack, ... Huffman, ... 예제실습 과제실습 	<ul style="list-style-type: none"> 주제(5): 탐색 <ul style="list-style-type: none"> 이진탐색, 선형... 예제실습 과제실습 	<ul style="list-style-type: none"> 주제(8): 동적프로그래밍 <ul style="list-style-type: none"> Memorization 예제실습 과제실습 	<ul style="list-style-type: none"> 주제(10): Graph(3) <ul style="list-style-type: none"> 기본형 review MSP (Kruskal, Bellman Ford, ...) 예제실습 과제실습
오후 (3Hrs)	<ul style="list-style-type: none"> 데이터구조 기초 <ul style="list-style-type: none"> :스택/큐 .. 주제(1): 알고리즘 개요와 복잡도 <ul style="list-style-type: none"> time, space,... 주제(2): 솔루션 전략과 완전탐색 <ul style="list-style-type: none"> 솔루션 전략 완전탐색 	<ul style="list-style-type: none"> 주제(4): 정렬 <ul style="list-style-type: none"> 이진정렬, ... 예제실습 과제실습 	<ul style="list-style-type: none"> 주제 6):Graph (1) <ul style="list-style-type: none"> DFS/BFS, ... 예제실습 과제실습 주제(7): 동적프로그래밍(1) <ul style="list-style-type: none"> Substructure/subproblem, 예제실습 과제실습 	<ul style="list-style-type: none"> 주제(9) Graph (2) <ul style="list-style-type: none"> 그래프 일반론 최단경로 (Dijkstra, ...) 예제실습 과제실습 	<ul style="list-style-type: none"> 개발형 코딩 <ul style="list-style-type: none"> 구현 Web/DB 데이터과학 등 확장 <ul style="list-style-type: none"> NP-Complete, Approximation, .. 마무리 <ul style="list-style-type: none"> 과정 review

- 교재:

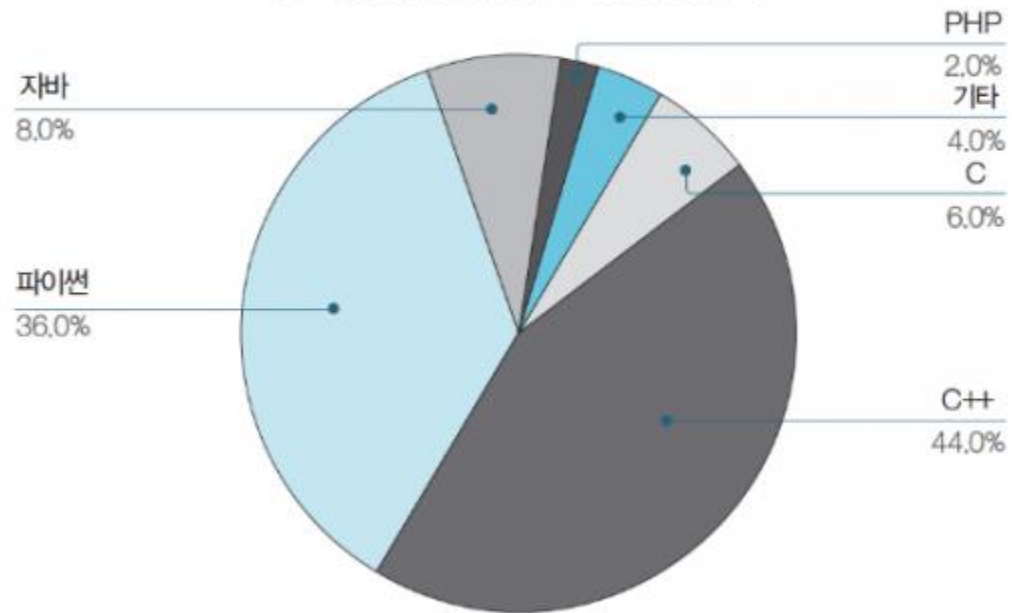
- 나동빈, 『이것이 취업을 위한 코딩 테스트다』, 한빛미디어, 2020
- 특기사항
 - 강의자료에는 위 교재의 내용과 문제가 일부 포함되어 있습니다.
 - (관련 저작권은 위 저자 및 출판사에 있습니다.)

코딩테스트

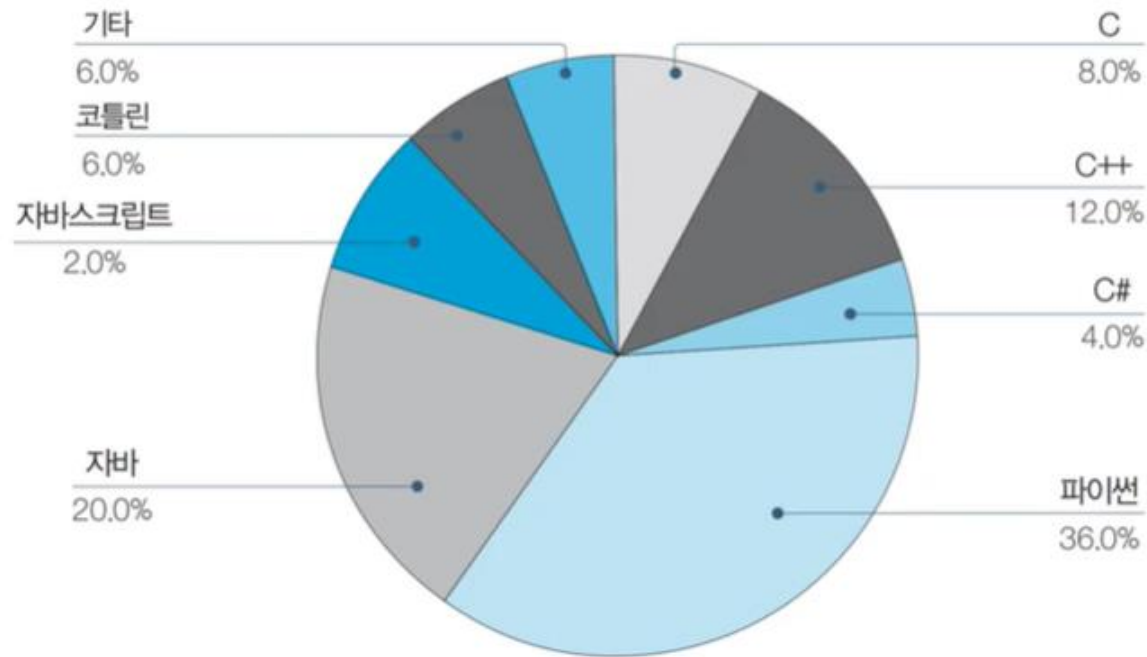
1 코딩 테스트 개념과 배경

- 해외
 - <https://codeforces.com/>
 - <https://www.topcoder.com/>
 - <https://leetcode.com/>
 - <https://www.codechef.com/>
- 국내
 - <https://www.acmicpc.net/>
 - <https://codeup.kr/index.php>
 - <https://programmers.co.kr/>
 - <https://swexpertacademy.com/main/main.do>
- 추가
 - <https://icpc.global/>
 - <http://icpckorea.org/>

알고리즘 문제 풀이 방식의 코딩 테스트에서 가장 유리한
프로그래밍 언어는 무엇이라고 생각하시나요?



프로그램 개발 방식의 코딩 테스트에서 가장 유리한
프로그래밍 언어는 무엇이라고 생각하시나요?



- 설문 대상: IT 계열 직군의 취업 준비생 및 경력자 50명 (2016 ~ 2019년 사이에 평균 3회 이상의 코딩 테스트 응시) / 설문 진행자: 나동빈

-
- https://www.hanbit.co.kr/channel/category/category_view.html?cms_code=CMS4385594264

- OJ (On-line Judge) 사이트

해외	http://codeforces.com
	https://www.topcoder.com/
	https://leetcode.com/
	https://www.codechef.com
	https://icpc.global/
	http://icpckorea.org/
국내	BOJ (백준 온라인 저지) https://www.acmicpc.net/
	https://codeup.kr
	https://programmers.co.kr
	SW Expert Academy https://swexpertacademy.com/main/main.do

2 실습 환경

- 온라인
 - C++
 - <https://www.onlinegdb.com/>
 - https://www.tutorialspoint.com/compile_cpp_online.php
 - Python
 - <https://replit.com/languages/python3>
 - Java
 - <https://www.online-java.com/>
 - <https://replit.com/languages/java10>
- 오프라인
 - C++
 - Python

실전

- 채용 프로세스



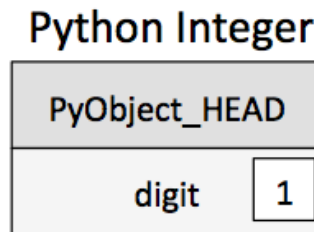
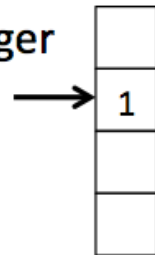
- 평상시 준비

- github 사용
- blog 관리

도구 (언어) REVIEW

- Python 특징

- ...
- VHLL (very high-level language) - dynamic typing, interpreter,
- OOP, FP
- 예: C Integer

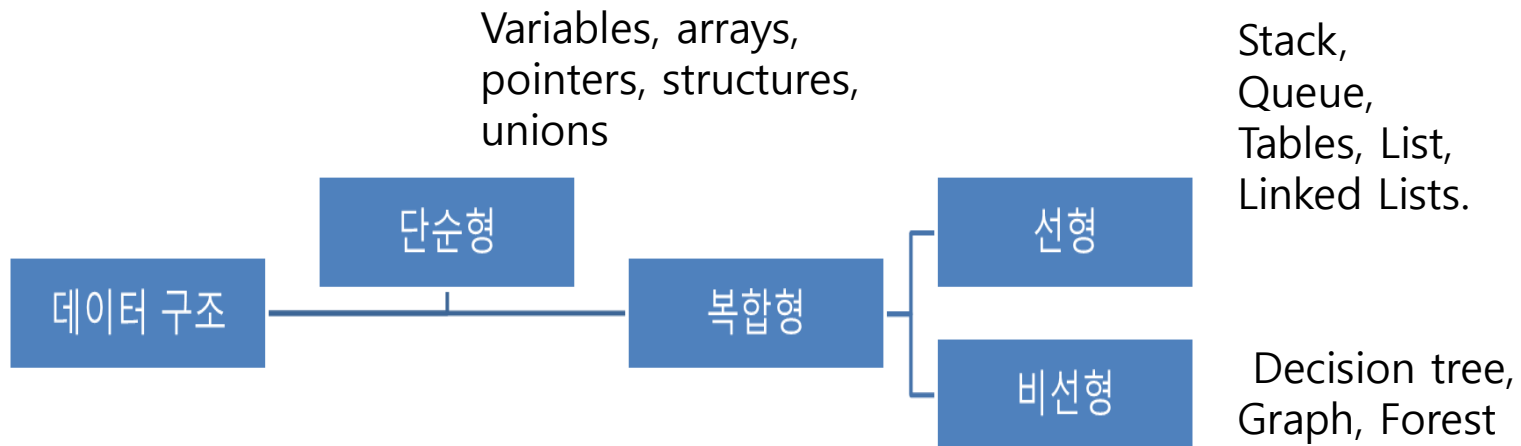


```
struct _longobject {  
    long ob_refcnt;  
    PyObject *ob_type;  
    size_t ob_size;  
    long ob_digit[1];  
};
```

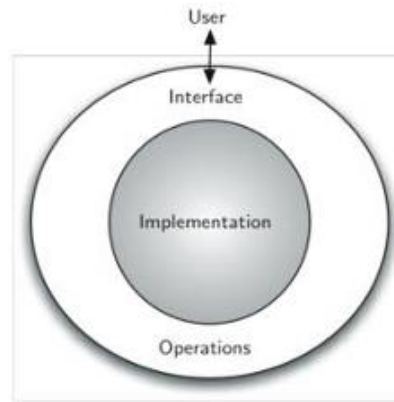
- A Python List Is More Than Just a List
- C++
 - MLL (middle-level language)

데이터 구조 기초

-
- 일반론
 - 개념
 - 종류



-
- Abstract Data Type (ADT)
 - 데이터 구조를 추상화한 것
 - 특정 데이터 구조가 준수해야 할 interface만을 규정



ADT	구현
List	Dynamic Array, Linked List
Queue	Linked List-based, Array-based, Stack-based Queue
Map	Tree Map, Hash map/ Hash Table
Vehicle	Golf cart, 자전거, 스마트 카

주요 데이터 구조

- Array
- Linked List
- Hash Table
- Stack
- Queue

Array


- 주식가격

```
stock_prices = [298,305,320,301,292]
```

```
stock_prices[0] ← 298 ← price on day 1
```

```
stock_prices[2] ← 320 ← price on day 3
```

stock_prices



0x00500

298

0x00504

305

0x00508

320

0x0050A


301

0x0050F

292

0x00500	298
0x00504	305
0x00508	320
0x0050A	301
0x0050F	292

stock_prices

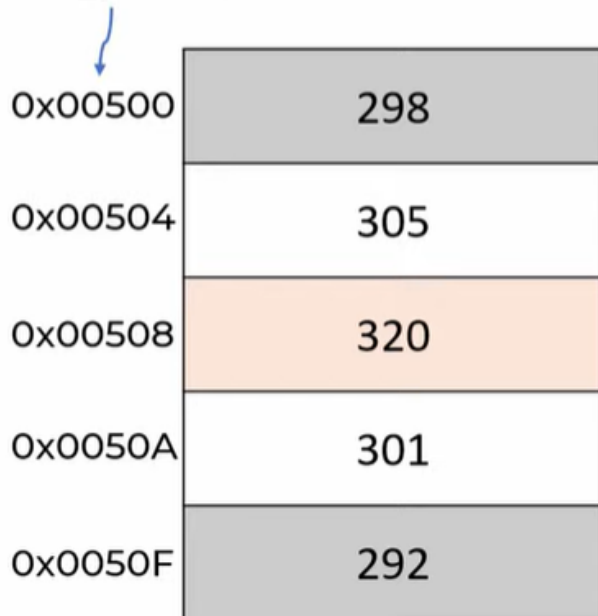


0x00500	298
0x00504	305
0x00508	320
0x0050A	301
0x0050F	292

0x00500	00000000	298
0x00501	00000000	
0x00502	00000001	
0x00503	00101010	
0x00504	00000000	305
0x00505	00000000	
0x00506	00000001	
0x00507	00110001	
	...	

-
- 상황 1: 12월3일의 주가?

stock_prices



0x00500	298
0x00504	305
0x00508	320
0x0050A	301
0x0050F	292

`stock_prices[2] → 320 (price on day 3)`

`stock_prices[0] → 0x00500`

`stock_prices[2] → 0x00500 + 2 * sizeof(integer)`


`stock_prices[2] → 0x00500 + 2 * 4`

`stock_prices[2] → 0x00508`

Lookup by index = 0(1)

-
- 상황 2: 언제 주가가 301?

stock_prices




0x00500	298
0x00504	305
0x00508	320
0x0050A	301
0x0050F	292

```
for i in range(len(stock_prices)):
    if stock_prices[i]==301:
        return i
```

Lookup by value = $O(n)$

-
- 상황 3: 주가 = 284를 index 1에 삽입

`stock_prices`



0x00500	298
0x00504	305
0x00508	320
0x0050A	301
0x0050F	292

`stock_prices.insert(1, 284)`

298
284
305
320
301
292

Array insertion = $O(n)$

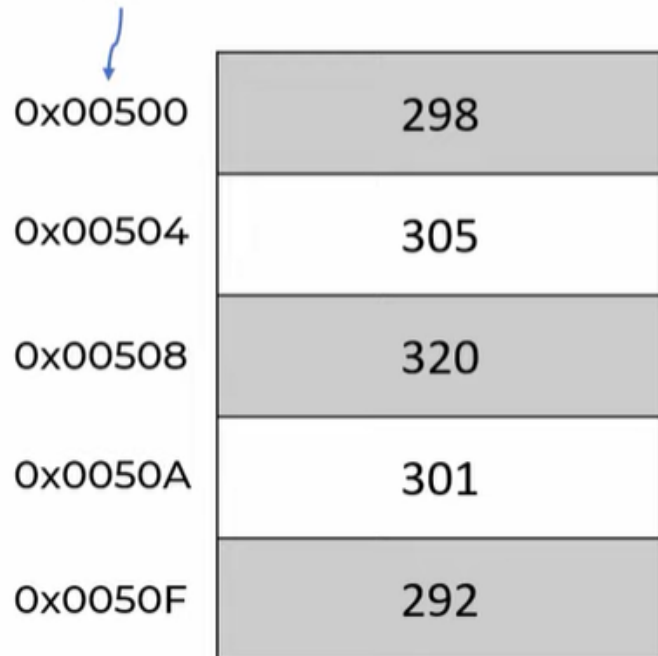
-
- Python에서는 list를 **dynamic array**로 구현
 - Java 및 C++에서는 **static** 및 **dynamic array**를 모두 가짐

Data Structure	Python	Java	C++
Static array		Native array	Native array
Dynamic array	List	ArrayList	std::vector

Linked List

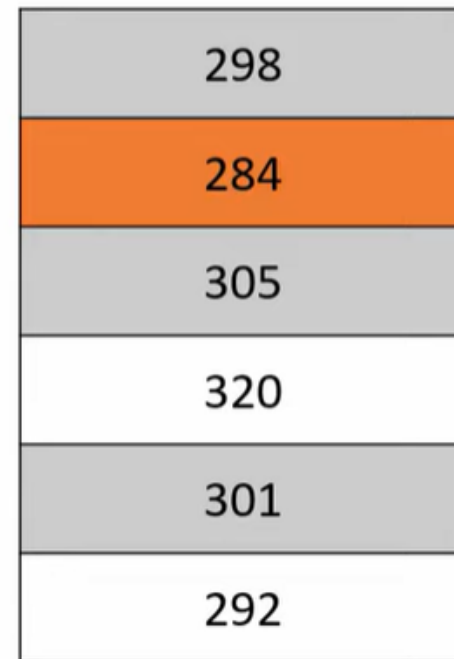
`stock_prices = [298,305,320,301,292]`

`stock_prices`



0x00500	298
0x00504	305
0x00508	320
0x0050A	301
0x0050F	292

`stock_prices.insert(1, 284)`



298
284
305
320
301
292

Array insertion = $O(n)$

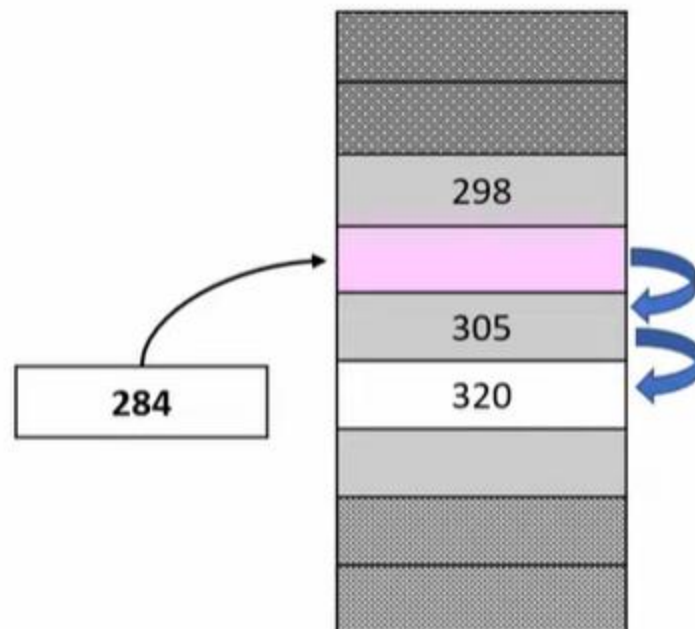
```
stock_prices = []
```

```
stock_prices.append(298)
```

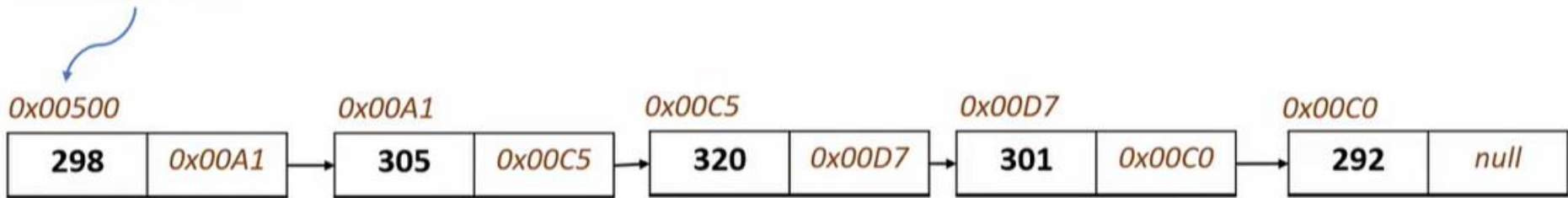
```
stock_prices.append(305)
```

```
stock_prices.append(320)
```

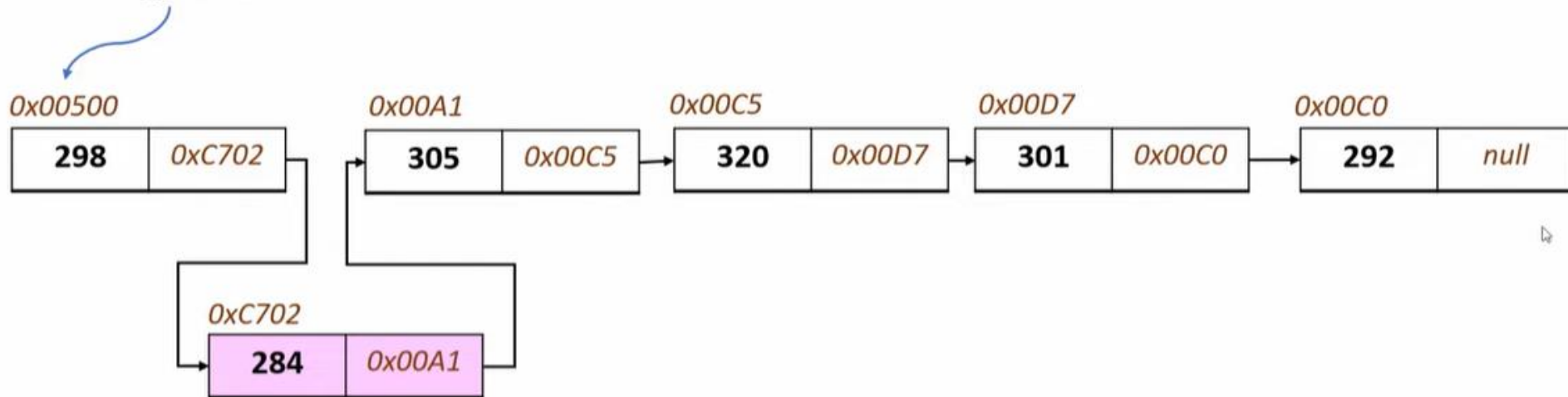
```
stock_prices.insert(1,284)
```



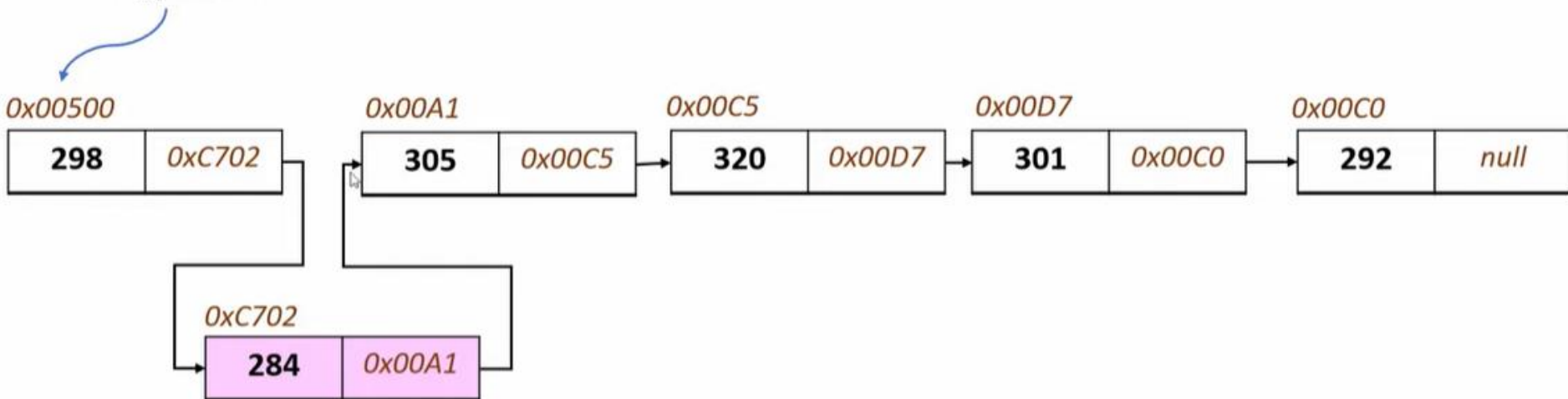
stock_prices



stock_prices

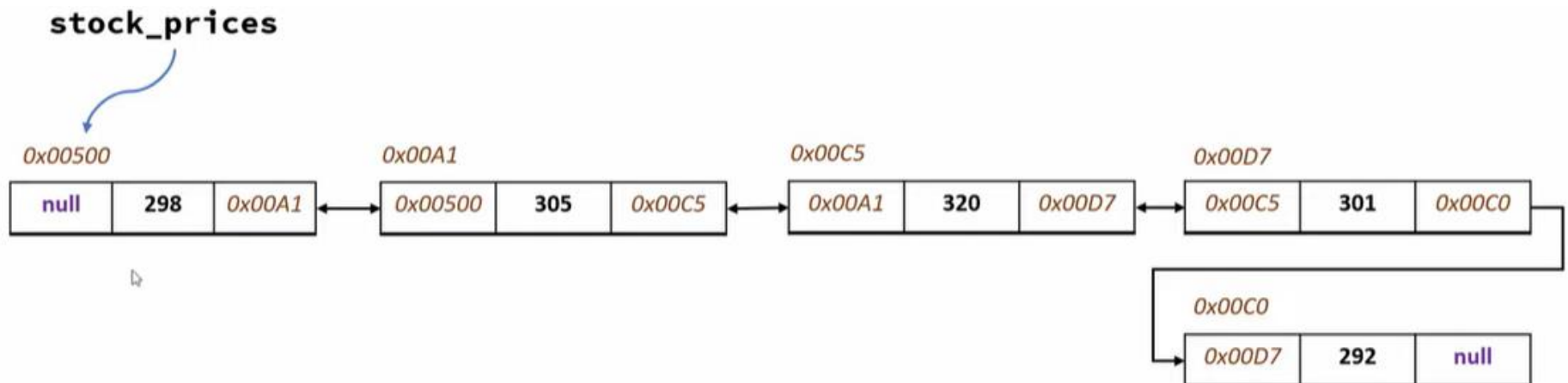


stock_prices



Insert Element at beginning = $O(1)$
Delete Element at beginning = $O(1)$
Insert/Delete Element at the end = $O(n)$

- Double Linked List

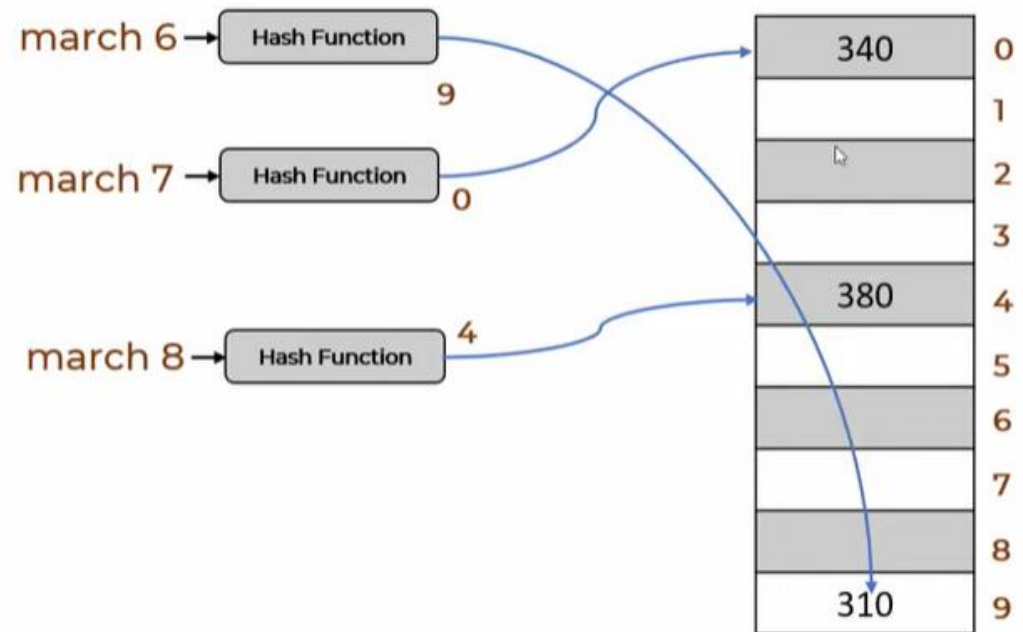


	Array	Linked List
Indexing	$O(1)$	$O(n)$
맨 앞에 Insert/Delete	$O(n)$	$O(1)$
맨 뒤에 Insert/Delete	$O(1)$ 0 (amortized)	$O(n)$
중간에 Insert Element	$O(n)$	$O(n)$

Hash Table

In [9]: stock_prices

Out[9]: {'march 6': 310.0,
'march 7': 340.0,
'march 8': 380.0,
'march 9': 302.0,
'march 10': 297.0,
'march 11': 323.0}



Look up by key is $O(1)$ on average
Insertion/Deletion is $O(1)$ on average

0	340
1	
2	310
3	
4	380
5	302
6	

stock_prices[0]

stock_prices[2]

march 6

march 7

march 8

340
310
380
302

stock_prices['march 6']

stock_prices['march 7']

march 6 → Hash Function → 9

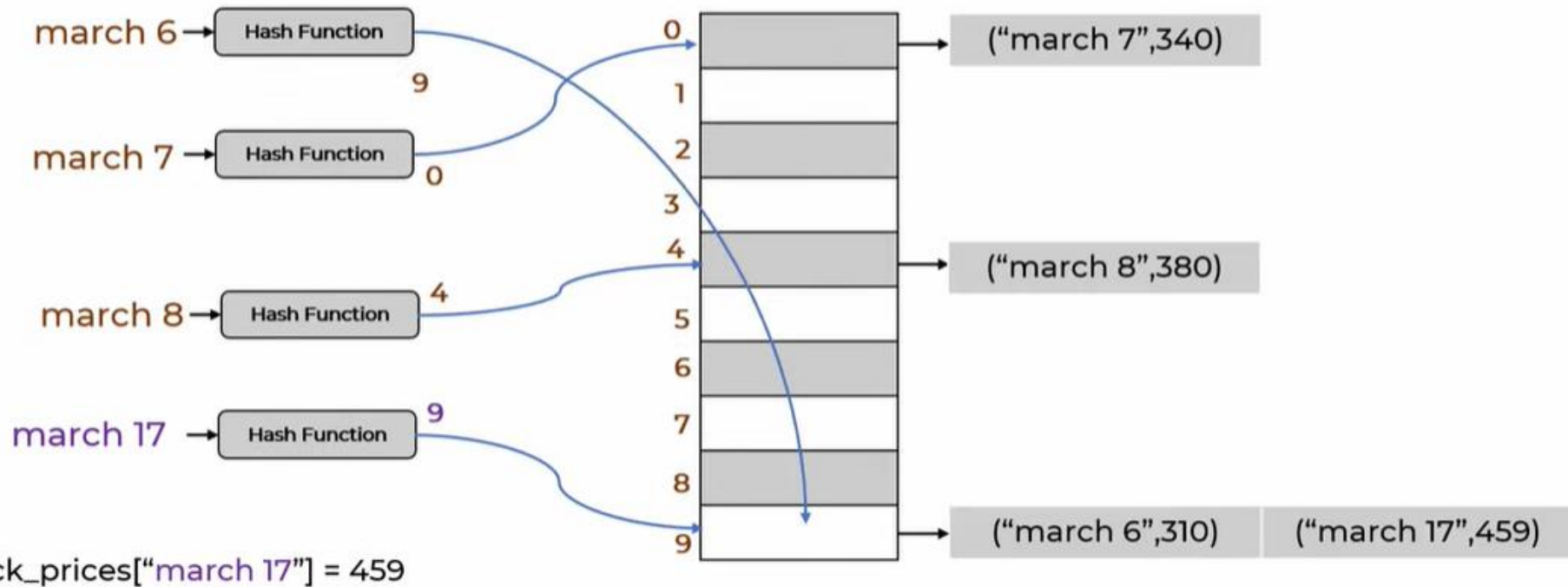
m	109
a	97
r	114
c	99
h	104
	32
6	54
SUM	609

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	Space	64	40	100	0	96	60	140	0	96	60	140
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	0	97	61	141
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	0	98	62	142
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	0	99	63	143
4	4	004	EOT (end of transmission)	36	24	044	\$	68	44	104	D	100	64	144	0	100	64	144
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	0	101	65	145
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	0	102	66	146
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	0	103	67	147
8	8	010	BS (backspace)	40	28	050	(72	48	110	H	104	68	150	0	104	68	150
9	9	011	TAB (horizontal tab)	41	29	051)	73	49	111	I	105	69	151	0	105	69	151
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	0	106	6A	152
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	0	107	6B	153
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	0	108	6C	154
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	0	109	6D	155
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	0	110	6E	156
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	0	111	6F	157
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	0	112	70	160
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	0	113	71	161
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	0	114	72	162
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	0	115	73	163
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	0	116	74	164
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	0	117	75	165
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	0	118	76	166
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	0	119	77	167
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	0	120	78	170
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	0	121	79	171
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	0	122	7A	172
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[123	7B	173	0	123	7B	173
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	0	124	7C	174
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135]	125	7D	175	0	125	7D	175
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	0	126	7E	176
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	0	127	7F	177

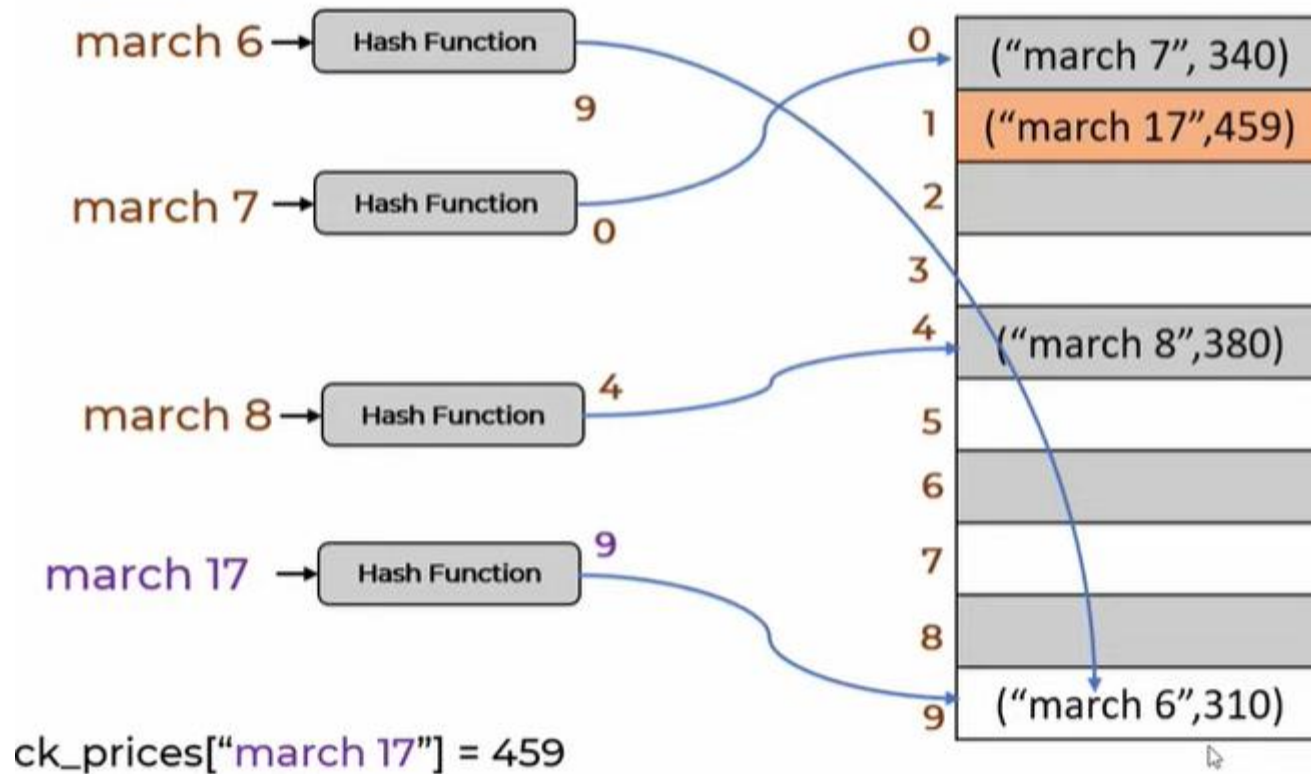
Source: www.LookupTables.com

	Class	Code Sample
Python	dictionary	<pre>prices = { 'march 6': 310, 'march 7': 430 }</pre>
JAVA	HashMap	<pre>HashMap<String, Integer> prices = new HashMap<String, Integer>(); prices.put("march 6",310); prices.put("march 7",430);</pre>
JAVA	LinkedHashMap	<pre>LinkedHashMap<String, Integer> prices = new LinkedHashMap<String, Integer>(); prices.put("march 6",310); prices.put("march 7",430);</pre>
C++	std::map	<pre>std::map<string,int> prices; prices['march 6']=310; prices['march 7']=430</pre>

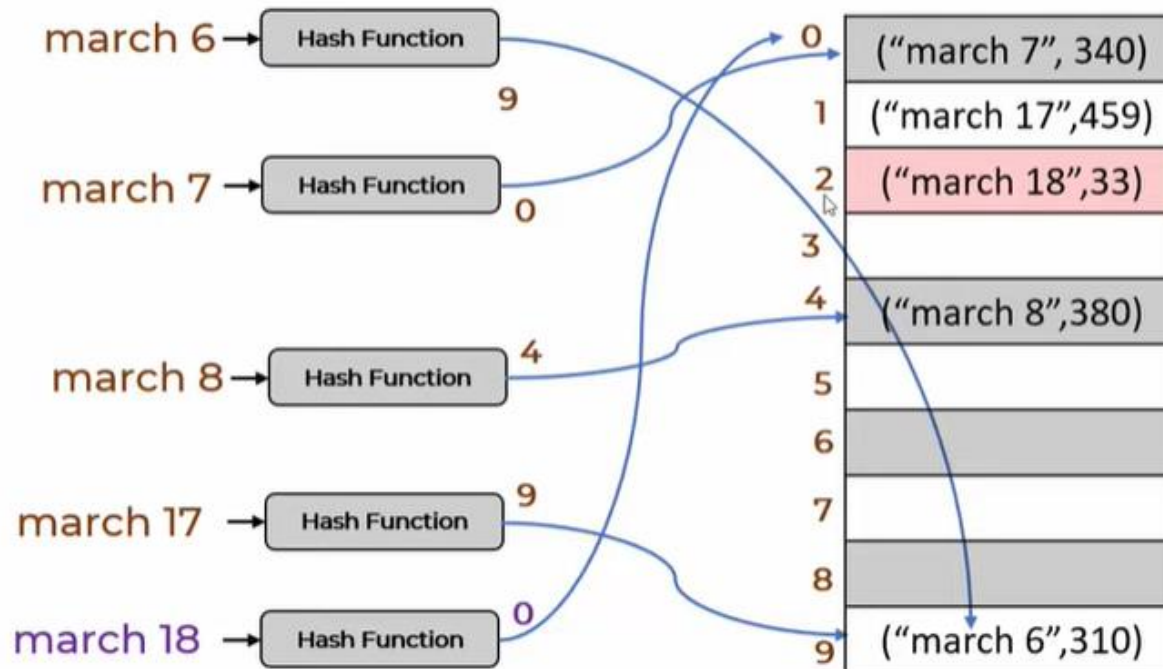
- Chaining을 이용한 Collision의 처리



- Linear Probing



- Linear Probing



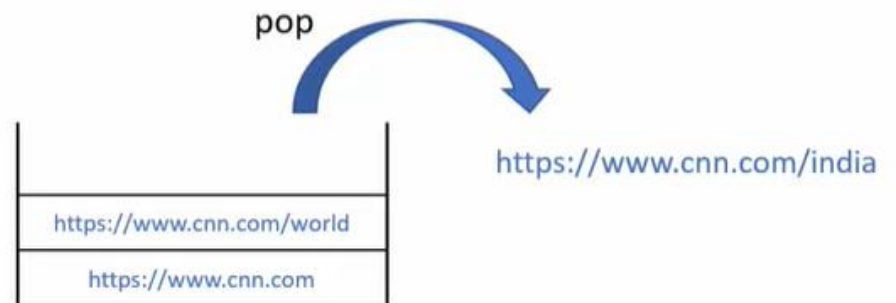
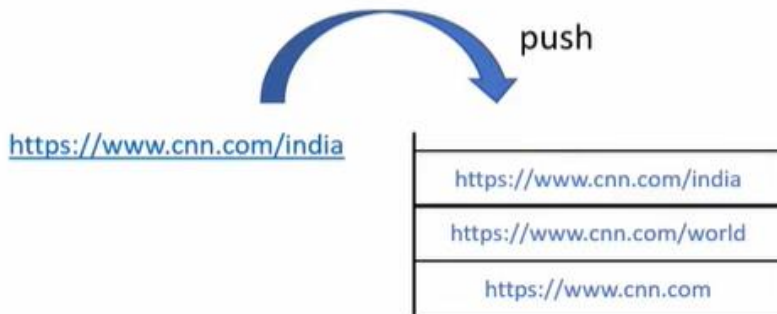
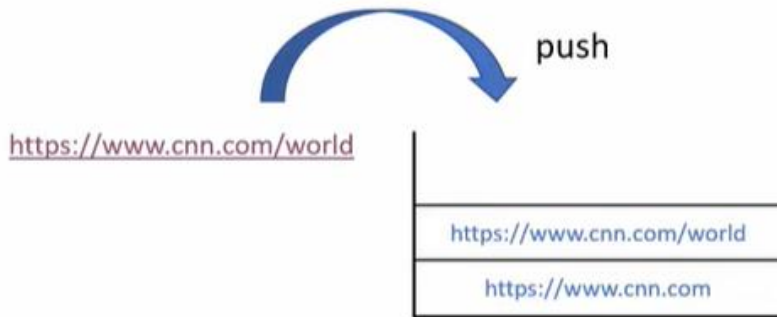
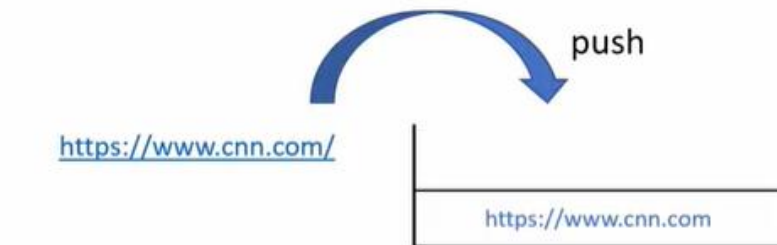
`stock_prices["march 18"] = 33`

Stack

Push/Pop element: $O(1)$
Search element by value: $O(n)$

Last In First Out

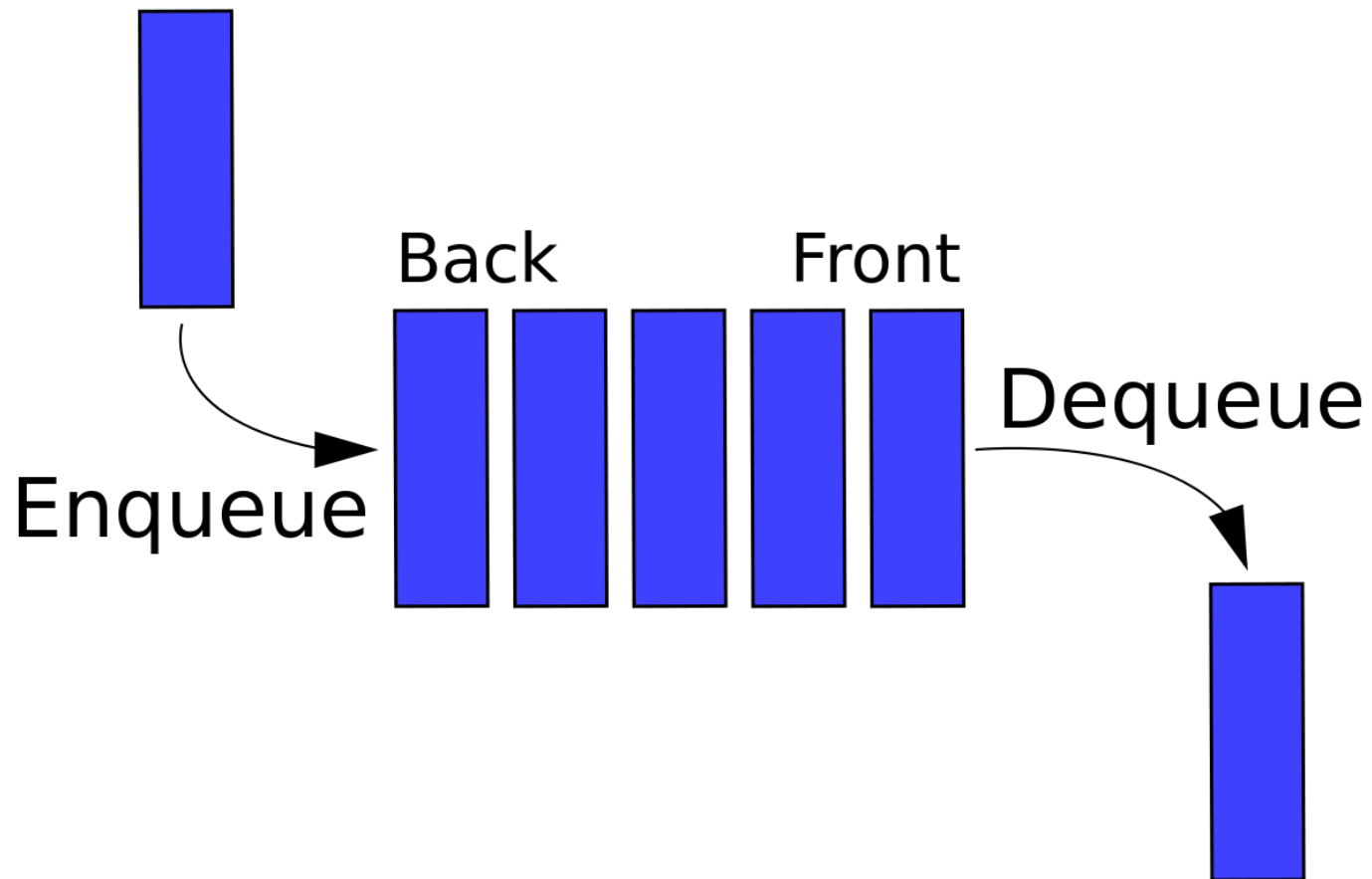
LIFO



	Class	Code Sample
Python	list collections.deque queue.LifoQueue	<pre>stk = deque() stk.append(5) stk.append(9) stk.pop() # returns 89</pre>
JAVA	Stack	<pre>Stack<Integer> stk = new Stack<>(); stk.push(5); stk.push(89); stk.pop(); // Returns 89</pre>
JAVA	Deque	<pre>Deque<Integer> stk = new ArrayDeque<>(); stk.push(5); stk.push(89); stk.pop(); // returns 89</pre>
C++	std::stack	<pre>std::stack<int> stk; stk.push(5); stk.push(89); stk.pop(); // Returns 89</pre>

Queue

- FIFO

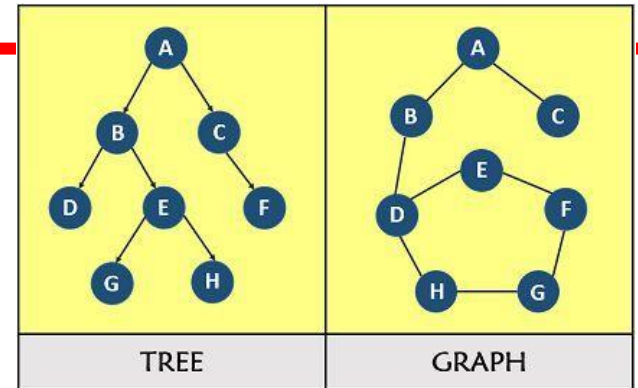


	Class	Code Sample
Python	list collections.deque queue.LifoQueue	<pre>q = deque() q.appendLeft(5) q.appendLeft(9) q.pop() # returns 5</pre>
JAVA	LinkedList	<pre>Queue<Integer> q = new LinkedList<>(); q.add(5); q.add(89); q.remove(); // Returns 5</pre>
C++	std::queue	<pre>std::queue<int> q; q.push(5); q.push(89); q.pop(); // Returns 5</pre>

비교

Data Structure	Python	Java	C++
Array	List	Native array Array List	Native array std::vector
Hash Table	Dictionary	HashMap LinkedHashMap	std::map
Linked List	N/A	LinkedList	std::list

Tree vs. Graph



비교 기준	트리	그래프
Path	Only one between two vertices.	More than one path allowed.
Root node	It has exactly one root node.	Graph doesn't have a root node.
Loops	No loops are permitted.	Graph can have loops.
Complexity	Less complex	More complex comparatively
Traversal techniques	Pre-order, In-order and Post-order.	Breadth-first search and depth-first search.
Number of edges	$n-1$ (단, n = number of nodes)	Not defined
Model type	Hierarchical	Network

알고리즘 개요와 복잡도

알고리즘 개요

- 알고리즘 개념
- 알고리즘 선택 (Solution) strategy
 - divide and conquer
 - 탐욕법 (greedy)
 - 퇴각전략 (Backtracking)
 - 분기와 한정 (Branch and bound)
 - 동적 프로그래밍 (Dynamic Programming 동태계획)

알고리즘 복잡도

복잡도	Class 명	Operation 예
$O(1)$	Constant	get item, set item, append
$O(\log n)$	Logarithmic	Array에서 element 검색
$O(n)$	Linear	insert, copy, iteration, delete
$n \log n$	Linear-logarithmic	merge-sort, sorting lists
n^2	Quadratic	Nested loops, Node간 최단경로 탐색
n^3	Cubic	max multiplication
2^n	Exponential	backtracking

Program Statements	Frequency Count
<p>.....</p> <p style="text-align: center;">$x = x + 2;$</p> <p>.....</p>	1
Total Frequency Count	1

Frequency Count of Program Segment B

Program Statements	Frequency Count
<p>.....</p> <p>for k =1 to n do</p> <p style="padding-left: 40px;">$x = x + 2;$</p> <p>end;</p> <p>.....</p>	<p>$(n+1)$</p> <p>n</p> <p>n</p>
Total Frequency Count	$3n+1$

Asymptotic Notation (점근표기)

- 관점
 - 시간복잡도 (time complexity): 실행시간 증가를 측정
 - 공간복잡도 (space complexity): 메모리 사용도

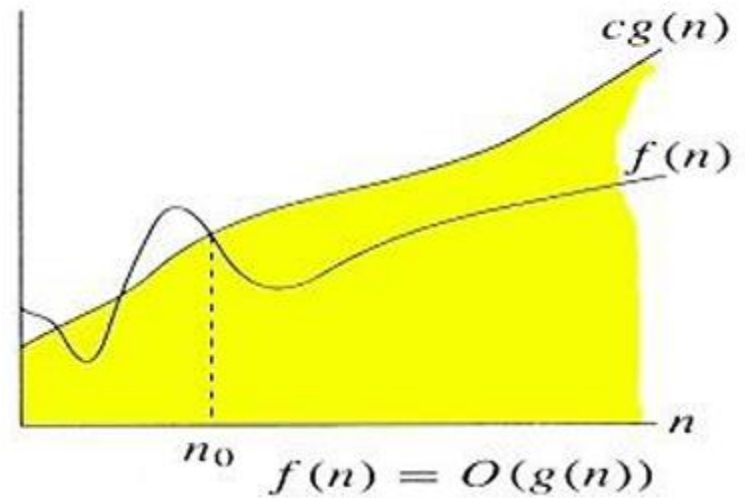
Program Statements	Frequency Count

for j = 1 to n do	(n+1)
for x = 1 to n do	n(n+1)
x = x + 2;	n^2
end	n^2
end;	n

Total Frequency Count	$3n^2 + 3n + 1$

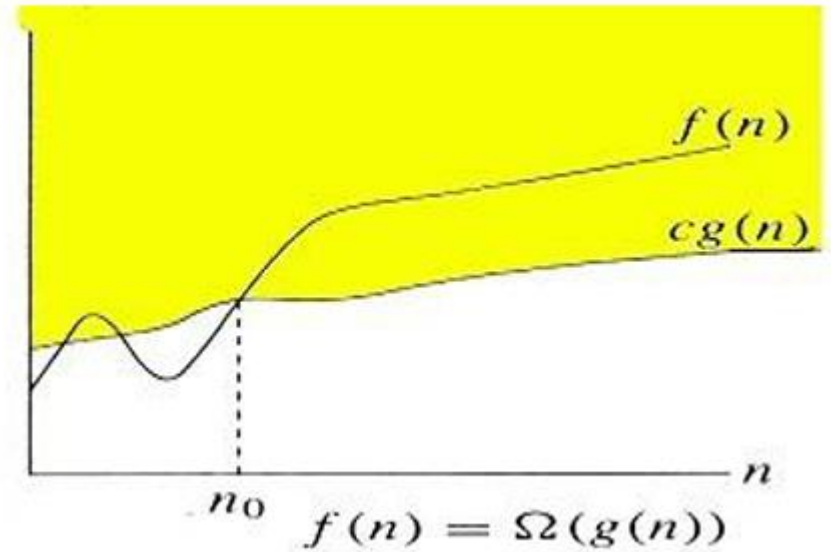
- Big-O

$f(n)$	$g(n)$	
$16n^3 + 45n^2 + 12n$	n^3	$f(n) = O(n^3)$
$34n - 40$	n	$f(n) = O(n)$
50	1	$f(n) = O(1)$



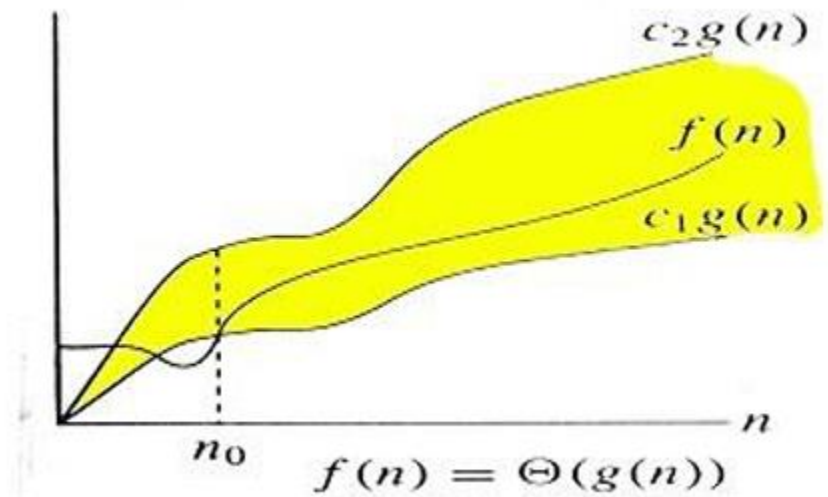
- Omega-O

$f(n)$	$g(n)$	
$16n^3 + 8n^2 + 2$	n^3	$f(n) = \Omega(n^3)$
$24n + 9$	n	$f(n) = \Omega(n)$



- Theta-O

$f(n)$	$g(n)$	
$16n^3 + 30n^2 - 90$	n^2	$f(n) = \Theta(n^2)$
$7 \cdot 2^n + 30n$	2^n	$f(n) = \Theta(2^n)$



- Numerical Comparison

S.No.	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
1.	0	1	1	1	1	2
2.	1	2	2	4	8	4
3.	2	4	8	16	64	16
4.	3	8	24	64	512	256
5.	4	16	64	256	4096	65536

Big-O

- 입력데이터 크기 변화에 따른 시간 및 공간 요구사항을 측정

- $O(n)$

```
def get_squared_numbers(numbers):  
    squared_numbers = []  
    for n in numbers:  
        squared_numbers.append(n * n)  
    return squared_numbers
```

```
numbers = [2,5,8,9]  
get_squared_numbers(numbers)
```

– $O(1)$

```
def find_first_pe(prices, eps, index):  
    pe = prices[index]/eps[index]  
    return pe
```

– $O(n^2)$

- $time = a * n^2 + b \rightarrow O(n^2)$

```
numbers = [3,6,2,4,3,6,8,9]  
  
for i in range(len(numbers)):J  
    for j in range(i+1, len(numbers)):  
        if numbers[i] == numbers[j]:  
            print(numbers[i] + " is a duplicate")  
            break
```

```
numbers = [3,6,2,4,3,6,8,9]
duplicate = None
```

```
for i in range(len(numbers)):
    for j in range(i+1, len(numbers)):
        if numbers[i] == numbers[j]:
            print(numbers[i] + " is a duplicate")
            break
```

n^2 iterations

```
for i in range(len(numbers)):
    if numbers[i] == duplicate:
        print(i)
```

n iterations

-
- Search for 68

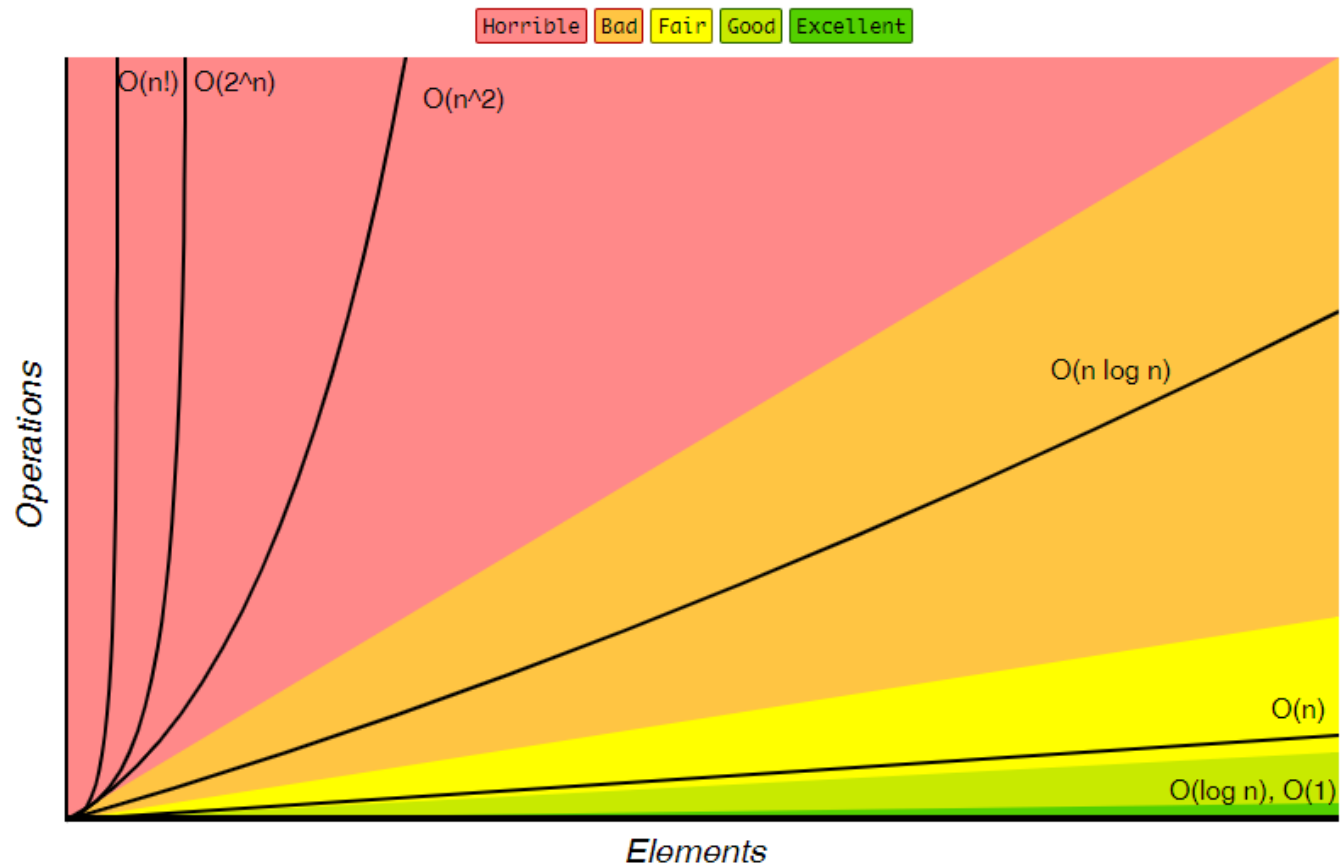
4	9	15	21	34	57	68	91
---	---	----	----	----	----	----	----

```
for i in range(len(numbers)):
    if numbers[i] == 68:
        print(i)
```

– $O(n)$

-
- 기타: 분할상환 분석 (Amortized analysis)
 - 개별 operation의 time complexity 보다는 average running time of sequence of operations의 평균 실행시간에 더 관심있을 때
 - 즉, 데이터구조의 state 변화를 감안.
 - 실행시간의 upper bound 결정 ← impose each operation in a sequence with an artificial cost. 이들 cost를 합산 계산

Big-O Complexity Chart



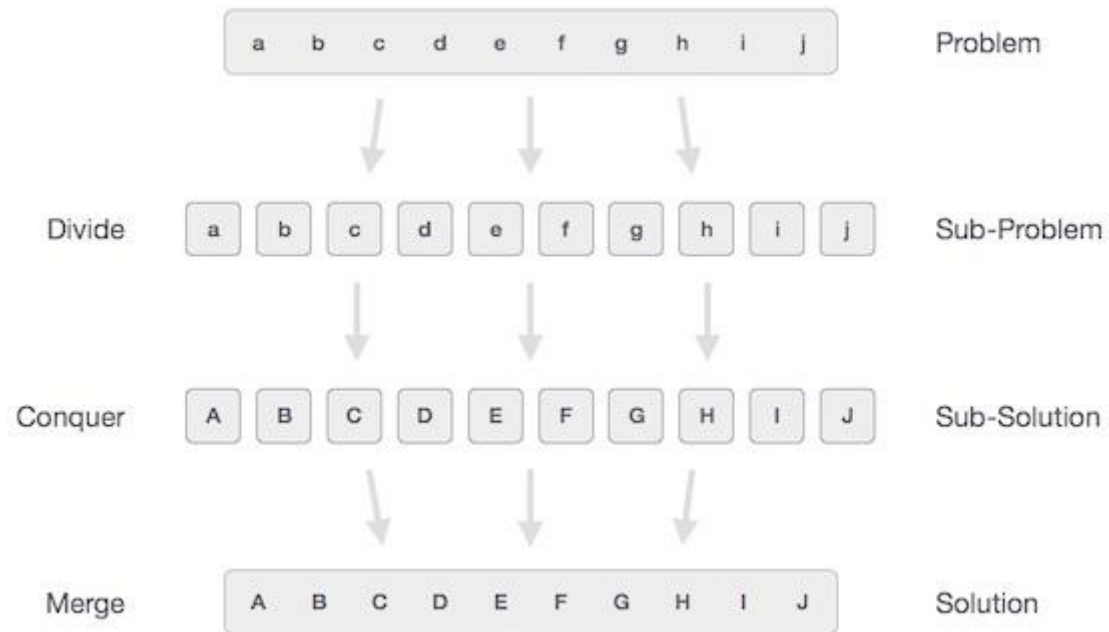
<https://www.bigocheatsheet.com/>

솔루션 전략과 완전탐색

솔루션 전략

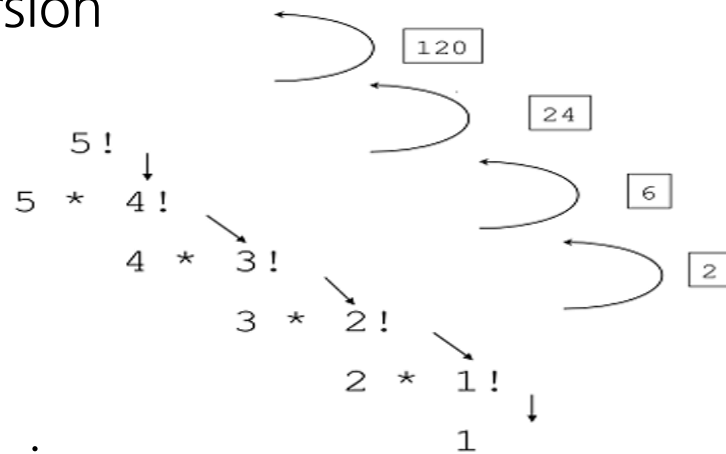
- 주요 전략
 - Brute Force
 - Divide-and-Conquer
 - Dynamic Programming
 - Recurrence

- Divide and Conquer

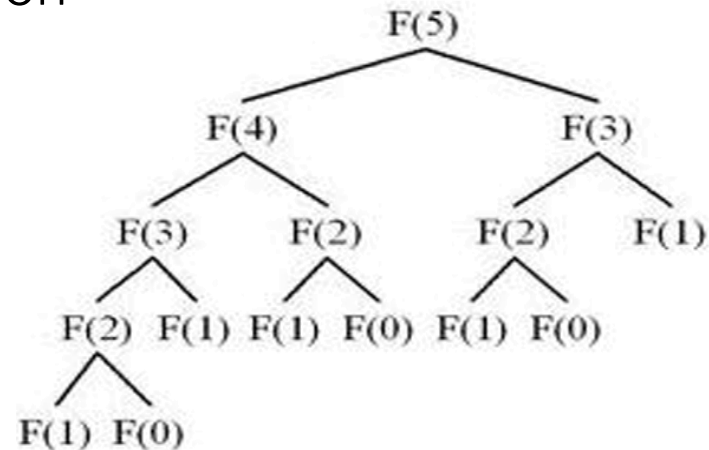


- Recursive Programming (Recurrence)

- Linear recursion



- Binary recursion



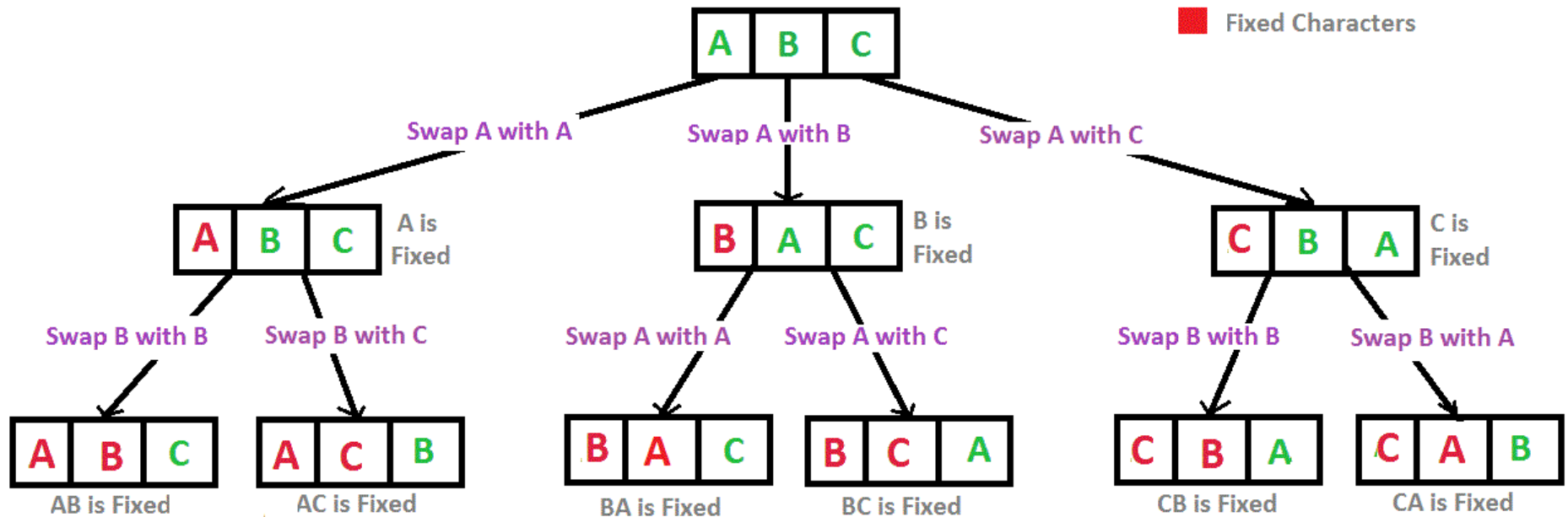
완전탐색

- 개념
 - 완전탐색(Exhaustive Search, Brute Force, 무식하게 ...)
- 적용여부
 - 문제해결 가능성
 - 효율성

- 이용방법

- 1) 문제해결 가능한 경우의 수를 대략적으로 계산.
- 2) 다양한 방법을 모색.
 - ① Brute Force 기법 - 반복 / 조건문을 활용해 모두 테스트하는 방법
 - ② 순열(Permutation)
 - ③ 재귀 호출
 - ④ 비트마스크
 - 비트(bit) 연산을 통해서 부분 집합을 표현
 - ⑤ BFS, DFS
- 3) 실제 적용.

- 예: permutation
 - Python:
 - C++



Recursion Tree for Permutations of String "ABC"

실습

예제 실습



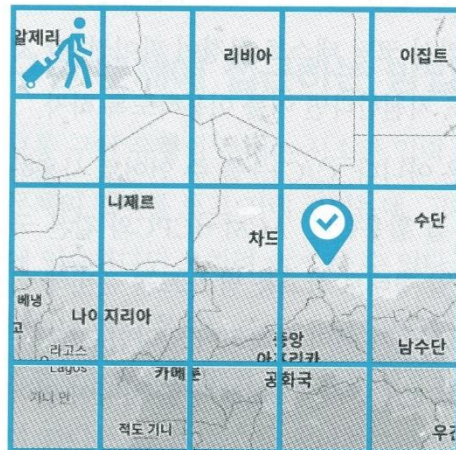
몸풀기

- 과제(1): 숫자암호 알아 맞추기
 - # 1~ 9999 사이의 숫자 암호를 입력한 후
 - # 완전탐색으로 맞춤
- 과제(변형)
 - ...

• 예제 4-1 상하좌우

난이도 ●○○ | 풀이 시간 15분 | 시간 제한 1초 | 메모리 제한 128MB

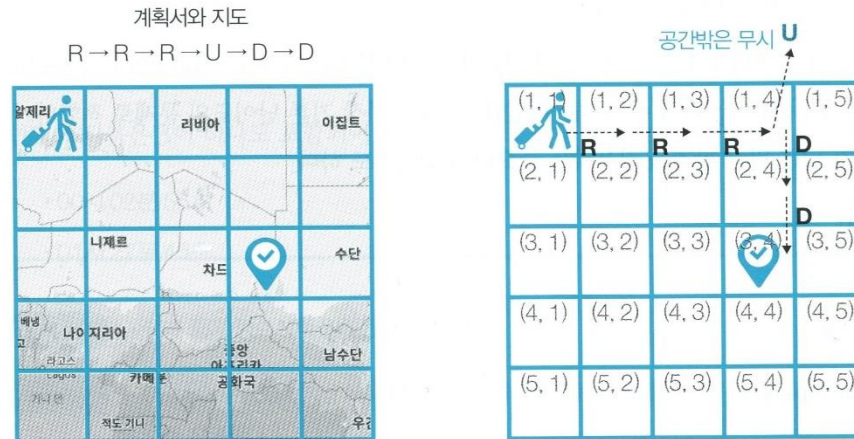
여행가 A는 $N \times N$ 크기의 정사각형 공간 위에 서 있다. 이 공간은 1×1 크기의 정사각형으로 나누어져 있다. 가장 왼쪽 위 좌표는 (1, 1)이며, 가장 오른쪽 아래 좌표는 (N, N)에 해당한다. 여행가 A는 상, 하, 좌, 우 방향으로 이동할 수 있으며, 시작 좌표는 항상 (1, 1)이다. 우리 앞에는 여행가 A가 이동할 계획이 적힌 계획서가 놓여 있다.



계획서에는 하나의 줄에 띄어쓰기를 기준으로 하여 L, R, U, D 중 하나의 문자가 반복적으로 적혀 있다. 각 문자의 의미는 다음과 같다.

- L: 왼쪽으로 한 칸 이동
- R: 오른쪽으로 한 칸 이동
- U: 위로 한 칸 이동
- D: 아래로 한 칸 이동

이때 여행가 A가 $N \times N$ 크기의 정사각형 공간을 벗어나는 움직임은 무시된다. 예를 들어 (1, 1)의 위치에서 L 혹은 U를 만나면 무시된다. 다음은 $N = 5$ 인 지도와 계획서이다.



이 경우 6개의 명령에 따라서 여행가가 움직이게 되는 위치는 순서대로 (1, 2), (1, 3), (1, 4), (2, 4), (3, 4)이므로, 최종적으로 여행가 A가 도착하게 되는 곳의 좌표는 (3, 4)이다. 다시 말해 3행 4열의 위치에 해당하므로 (3, 4)라고 적는다. 계획서가 주어졌을 때 여행가 A가 최종적으로 도착할 지점의 좌표를 출력하는 프로그램을 작성하시오.

- 입력 조건**
- 첫째 줄에 공간의 크기를 나타내는 N 이 주어진다. ($1 \leq N \leq 100$)
 - 둘째 줄에 여행가 A가 이동할 계획서 내용이 주어진다. ($1 \leq \text{이동 횟수} \leq 100$)

- 출력 조건**
- 첫째 줄에 여행가 A가 최종적으로 도착할 지점의 좌표 (X, Y)를 공백으로 구분하여 출력한다.

입력 예시

5
R R R U D D

출력 예시

3 4

Q 07 럭키 스트레이트

난이도 ●○○ | 풀이 시간 20분 | 시간 제한 1초 | 메모리 제한 256MB | 기출 핵심 유형

링크 <https://www.acmicpc.net/problem/18406>

게임의 아웃복서 캐릭터는 필살기인 '럭키 스트레이트' 기술이 있습니다. 이 기술은 매우 강력한 대신에 게임 내에서 점수가 특정 조건을 만족할 때만 사용할 수 있습니다.

특정 조건이란 현재 캐릭터의 점수를 N 이라고 할 때 자릿수를 기준으로 점수 N 을 반으로 나누어 왼쪽 부분의 각 자릿수의 합과 오른쪽 부분의 각 자릿수의 합을 더한 값이 동일한 상황을 의미합니다. 예를 들어 현재 점수가 123,402라면 왼쪽 부분의 각 자릿수의 합은 $1 + 2 + 3$, 오른쪽 부분의 각 자릿수의 합은 $4 + 0 + 2$ 이므로 두 합이 6으로 동일하여 럭키 스트레이트를 사용할 수 있습니다.

현재 점수 N 이 주어지면 럭키 스트레이트를 사용할 수 있는 상태인지 아닌지를 알려주는 프로그램을 작성하세요.

입력 조건 • 첫째 줄에 점수 N 이 정수로 주어집니다. ($10 \leq N \leq 99,999,999$) 단, 점수 N 의 자릿수는 항상 짝수 형태로만 주어집니다. 예를 들어 자릿수가 5인 12,345와 같은 수는 입력으로 들어오지 않습니다.

출력 조건 • 첫째 줄에 럭키 스트레이트를 사용할 수 있다면 "LUCKY"를, 사용할 수 없다면 "READY"를 출력합니다.

입력 예시 1

123402

출력 예시 1

LUCKY

입력 예시 2

7755

출력 예시 2

READY

과제 실습



난이도 ●○○ | 풀이 시간 50분 | 시간 제한 5초 | 메모리 제한 128MB | 기출 2020 카카오 신입 공채

링크 <https://programmers.co.kr/learn/courses/30/lessons/60061>

주의! 이 문제는 기본 코드가 제공되므로 상기 링크를 통해서 문제를 풀어야 합니다.

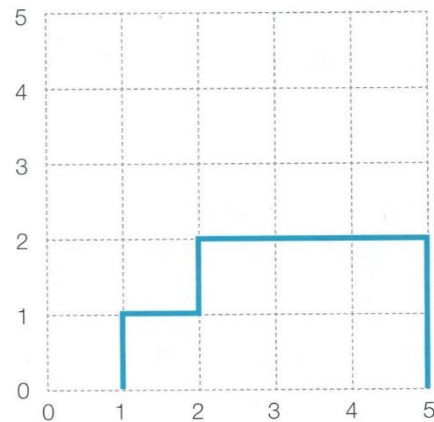
빙하가 깨지면서 스노우타운에 떠내려온 조르디는 인생 2막을 위해 주택 건축사업에 뛰어들기로 결심하였습니다. 조르디는 기둥과 보를 이용하여 벽면 구조물을 자동으로 세우는 로봇을 개발할 계획인데, 그에 앞서 로봇의 동작을 시뮬레이션 할 수 있는 프로그램을 만들고 있습니다.

프로그램은 2차원 가상 벽면에 기둥과 보를 이용한 구조물을 설치할 수 있는데, 기둥과 보의 길이가 1인 선분으로 표현되며 다음과 같은 규칙을 가지고 있습니다.

- 기둥은 바닥 위에 있거나 보의 한쪽 끝부분 위에 있거나, 또는 다른 기둥 위에 있어야 합니다.
- 보의 한쪽 끝부분이 기둥 위에 있거나, 또는 양쪽 끝부분이 다른 보와 동시에 연결되어 있어야 합니다.

단, 바닥은 벽면의 맨 아래 지면을 말합니다.

2차원 벽면은 $n \times n$ 크기 정사각 격자 형태이며, 각 격자는 1×1 크기입니다. 맨 처음 벽면은 비어 있는 상태입니다. 기둥과 보의 격자 선의 교차점에 걸치지 않고, 격자 간의 각 변에 정확히 일치하도록 설치할 수 있습니다. 다음은 기둥과 보를 설치해 구조물을 만든 예시입니다.



예를 들어, 이 그림은 다음 순서에 따라 구조물을 만들었습니다.

1. (1, 0)에서 위쪽으로 기둥을 하나 설치 후, (1, 1)에서 오른쪽으로 보를 하나 만듭니다.
2. (2, 1)에서 위쪽으로 기둥을 하나 설치 후, (2, 2)에서 오른쪽으로 보를 하나 만듭니다.
3. (5, 0)에서 위쪽으로 기둥을 하나 설치 후, (5, 1)에서 위쪽으로 기둥을 하나 더 설치합니다.
4. (4, 2)에서 오른쪽으로 보를 설치 후, (3, 2)에서 오른쪽으로 보를 설치합니다.

만약 (4, 2)에서 오른쪽으로 보를 먼저 설치하지 않고, (3, 2)에서 오른쪽으로 보를 설치하려 한다면 2번 규칙에 맞지 않으므로 설치가 되지 않습니다. 기둥과 보를 삭제하는 기능도 있는데 기둥과 보를 삭제한 후에 남은 기둥과 보 또한 위 규칙을 만족해야 합니다. 만약, 작업을 수행한 결과가 조건을 만족하지 않는다면 해당 작업은 무시됩니다.

벽면의 크기 n , 기둥과 보를 설치하거나 삭제하는 작업이 순서대로 담긴 2차원 배열 `build_frame`이 매개변수로 주어질 때, 모든 명령어를 수행한 후 구조물의 상태를 `return` 하도록 `solution` 함수를 완성해주세요.

제한 사항

- n 은 5 이상 100 이하인 자연수입니다.
- `build_frame`의 세로(행) 길이는 1 이상 1,000 이하입니다.
- `build_frame`의 가로(열) 길이는 4입니다.
- `build_frame`의 원소는 $[x, y, a, b]$ 형태입니다.
 - x, y 는 기둥, 보를 설치 또는 삭제할 교차점의 좌표이며, [가로 좌표, 세로 좌표] 형태입니다.
 - a 는 설치 또는 삭제할 구조물의 종류를 나타내며, 0은 기둥, 1은 보를 나타냅니다.
 - b 는 구조물을 설치할 지, 혹은 삭제할 지를 나타내며 0은 삭제, 1은 설치를 나타냅니다.
 - 벽면을 벗어나게 기둥, 보를 설치하는 경우는 없습니다.
 - 바닥에 보를 설치 하는 경우는 없습니다.
- 구조물은 교차점 좌표를 기준으로 보는 오른쪽, 기둥은 위쪽 방향으로 설치 또는 삭제합니다.
- 구조물이 겹치도록 설치하는 경우와 없는 구조물을 삭제하는 경우는 입력으로 주어지지 않습니다.
- 최종 구조물의 상태는 아래 규칙에 맞춰 `return` 해주세요.
 - `return` 하는 배열은 가로(열) 길이가 3인 2차원 배열로, 각 구조물의 좌표를 담고 있어야 합니다.
 - `return` 하는 배열의 원소는 $[x, y, a]$ 형식입니다.
 - x, y 는 기둥, 보의 교차점 좌표이며, [가로 좌표, 세로 좌표] 형태입니다.
 - 기둥, 보는 교차점 좌표를 기준으로 오른쪽, 또는 위쪽 방향으로 설치되어 있음을 나타냅니다.

- a는 구조물의 종류를 나타내며, 0은 기둥, 1은 보를 나타냅니다.
- return 하는 배열은 x 좌표 기준으로 오름차순 정렬하며, x 좌표가 같을 경우 y 좌표 기준으로 오름차순 정렬해주세요.
- x, y 좌표가 모두 같은 경우 기둥이 보보다 앞에 오면 됩니다.

입출력 예시

n	bulid_frame	result
5	[[1, 0, 0, 1], [1, 1, 1, 1], [2, 1, 0, 1], [2, 2, 1, 1], [5, 0, 0, 1], [5, 1, 0, 1], [4, 2, 1, 1], [3, 2, 1, 1]]	[[1, 0, 0], [1, 1, 1], [2, 1, 0], [2, 2, 1], [3, 2, 1], [4, 2, 1], [5, 0, 0], [5, 1, 0]]
5	[[0, 0, 0, 1], [2, 0, 0, 1], [4, 0, 0, 1], [0, 1, 1, 1], [1, 1, 1, 1], [2, 1, 1, 1], [3, 1, 1, 1], [2, 0, 0, 0], [1, 1, 1, 0], [2, 2, 0, 1]]	[[0, 0, 0], [0, 1, 1], [1, 1, 1], [2, 1, 1], [3, 1, 1], [4, 0, 0]]

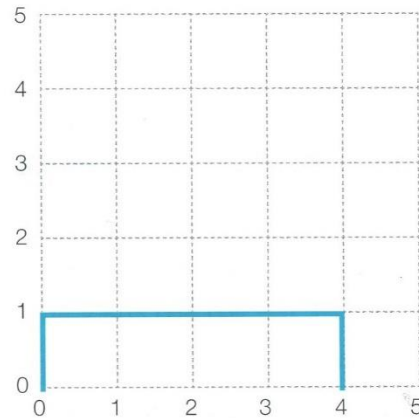
입출력 예시에 대한 설명

입출력 예시 #1

문제의 예시와 같습니다.

입출력 예시 #2

여덟 번째 작업을 수행 후 다음과 같은 구조물이 만들어집니다.



- 아홉 번째 작업의 경우, (1, 1)에서 오른쪽에 있는 보를 삭제하면 (2, 1)에서 오른쪽에 있는 보의 조건을 만족하지 않으므로 무시됩니다.
- 열 번째 작업의 경우, (2, 2)에서 위쪽 방향으로 기둥을 세울 경우 조건을 만족하지 않으므로 무시됩니다.

- 6.3 문제: 소풍 (문제 ID: PICNIC, 난이도: 하) 출처: 알고리즘 문제해결전략
- 1. 문제
 - 안드로메다 유치원 익스프레스반에서는 다음 주에 울동공원으로 소풍을 갑니다. 원석 선생님은 소풍 때 학생들을 두 명씩 짝을 지어 행동하게 하려고 합니다. 그런데 서로 친구가 아닌 학생들끼리 짝을 지어 주면 서로 싸우거나 같이 돌아다니지 않기 때문에, 항상 서로 친구인 학생들끼리만 짝을 지어야 합니다.
 - 각 학생들의 쌍에 대해 이들이 서로 친구인지 여부가 주어질 때, 학생들을 짝 지을 수 있는 방법의 수를 계산하는 프로그램을 작성하세요. 짝이 되는 학생들이 일부만 다르더라도 다른 방법이라고 봅니다. 예를 들어 다음 두 가지 방법은 서로 다른 방법입니다.
 - (태연,제시카)(써니,티파니)(효연,유리)
 - (태연,제시카)(써니,유리)(효연, 티파니)
- 2. 시간 및 메모리 제한
 - 프로그램은 1초 내에 실행되어야 하고, 64MB 이하의 메모리만을 사용해야 합니다.
- 3. 입력
 - 입력의 첫 줄에는 테스트 케이스 수 $C(C \leq 50)$ 가 주어집니다. 각 테스트 케이스의 첫 줄에는 학생의 수 $n(2 \leq n \leq 10)$ 과 친구 쌍의 수 $m(0 \leq m \leq n \cdot (n-1)/2)$ 이 주어집니다. 그 다음 줄에 m 개의 정수 쌍으로 서로 친구인 두 학생의 번호가 주어집니다. 번호는 모두 0부터 $n-1$ 사이의 정수이고, 같은 쌍은 입력에 두 번 주어지지 않습니다. 학생들의 수는 짝수입니다.
- 4. 출력
 - 각 테스트 케이스마다 한 줄에 모든 학생을 친구끼리만 짝지어줄 수 있는 방법의 수를 출력합니다.
- 예제입력
 - 3
 - 2 1
 - 0 1
 - 4 6
 - 0 1 1 2 2 3 3 0 0 2 1 3
 - 6 10
 - 0 1 0 2 1 2 1 3 1 4 2 3 2 4 3 4 3 5 4 5
- 예제출력
 - 1
 - 3
 - 4
- 예제 입출력 설명
 - 첫 번째 입력 (2 1)에는 두 학생밖에 없으며 이들은 서로 친구입니다. 따라서 딱 한 가지의 경우가 있게 되죠.
 - 두 번째 입력 (4 6)에는 네 명의 학생이 모두 서로 친구 입니다. 이들을 보라돌이, 뚜비, 나나, 뽀라고 할 때 다음과 같은 세 가지 방법이 있습니다.
 - (보라돌이, 뚜비)(나나, 뽀)
 - (뚜비, 뽀)(보라돌이, 나나)
 - (나나, 뚜비)(뽀, 보라돌이)