

삼성 청년 SW 아카데미

APS 기본

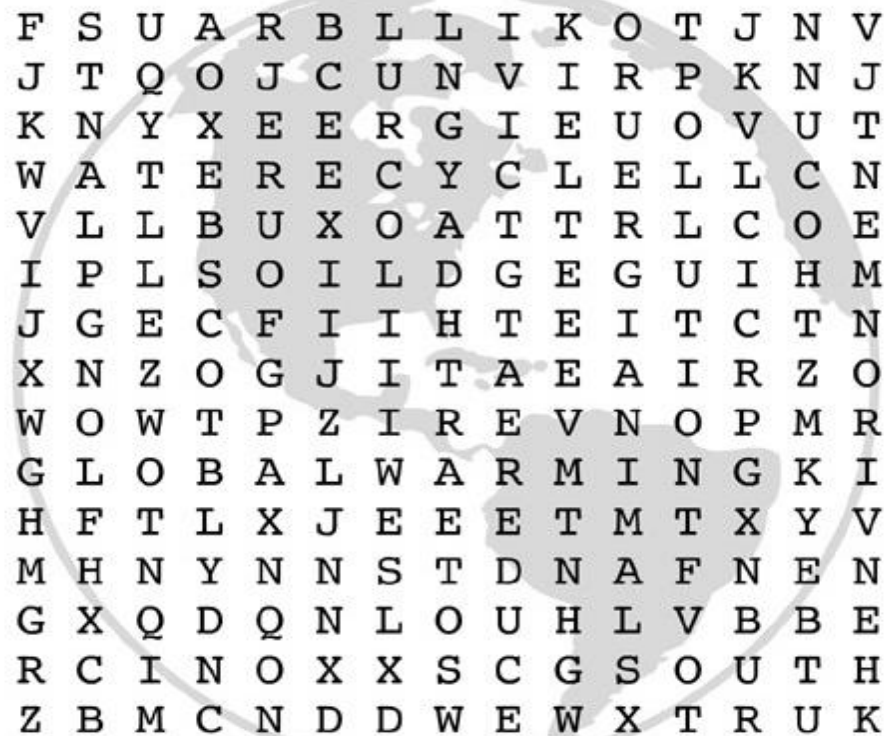
문자열 (String)

- 문자열
- 패턴 매칭
- (부록) 문자열 암호화
- (부록) 문자열 압축

문자열

Earth Day Word Search

Try to find all of the hidden Earth Day words in the word puzzle below.
Remember, words can be diagonal, vertical, horizontal, forward or backwards.



F	S	U	A	R	B	L	L	I	K	O	T	J	N	V
J	T	Q	O	J	C	U	N	V	I	R	P	K	N	J
K	N	Y	X	E	E	R	G	I	E	U	O	V	U	T
W	A	T	E	R	E	C	Y	C	L	E	L	L	C	N
V	L	L	B	U	X	O	A	T	T	R	L	C	O	E
I	P	L	S	O	I	L	D	G	E	G	U	I	H	M
J	G	E	C	F	I	I	H	T	E	I	T	C	T	N
X	N	Z	O	G	J	I	T	A	E	A	I	R	Z	O
W	O	W	T	P	Z	I	R	E	V	N	O	P	M	R
G	L	O	B	A	L	W	A	R	M	I	N	G	K	I
H	F	T	L	X	J	E	E	E	T	M	T	X	Y	V
M	H	N	Y	N	N	S	T	D	N	A	F	N	E	N
G	X	Q	D	Q	N	L	O	U	H	L	V	B	B	E
R	C	I	N	O	X	X	S	C	G	S	O	U	T	H
Z	B	M	C	N	D	D	W	E	W	X	T	R	U	K

Earth Day
Environment
Conservation

Reduce
Reuse
Recycle

Air
Soil
Water

People
Plants
Animals

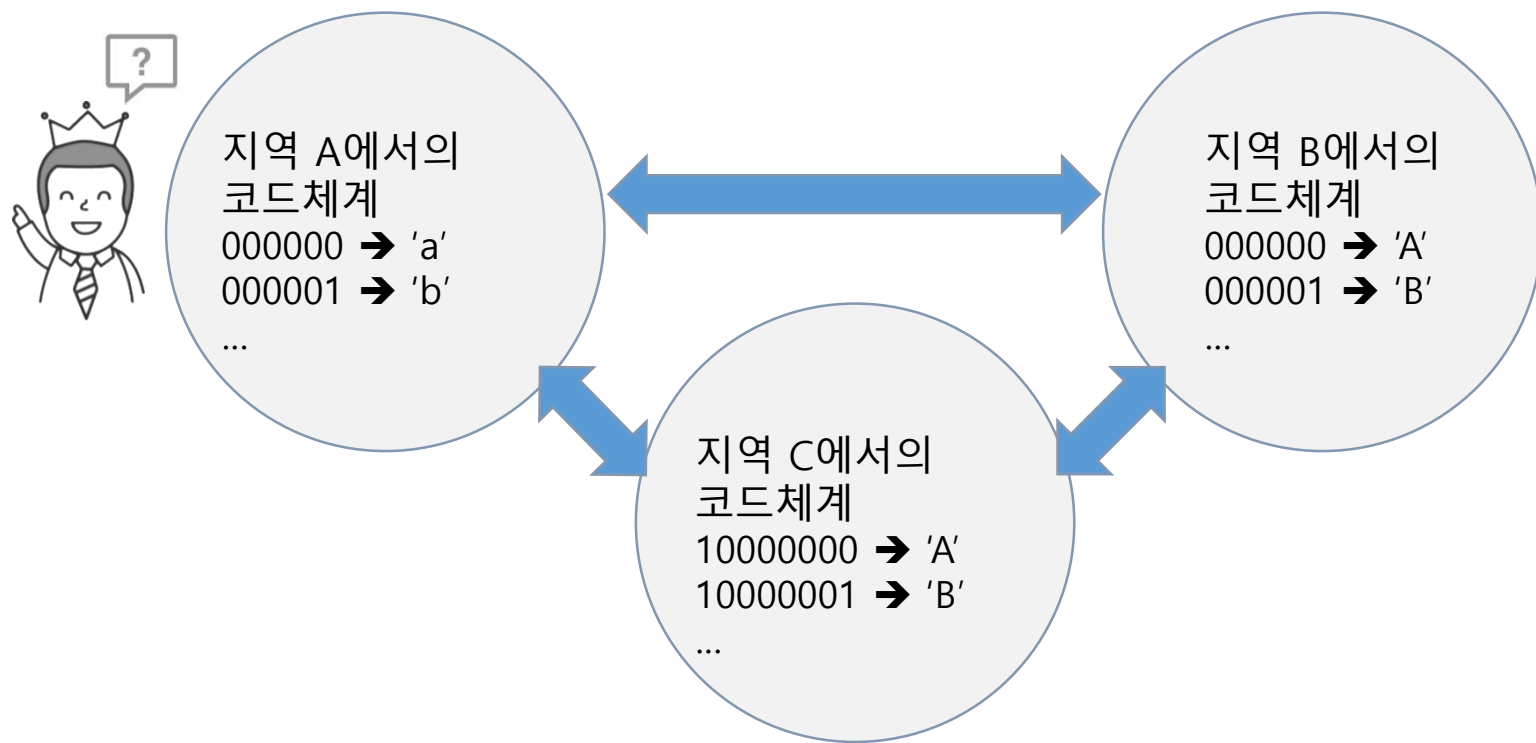
Litter
Pollution
Global Warming

✓ 컴퓨터에서의 문자표현

- 글자 A를 메모리에 저장하는 방법에 대해서 생각해보자
- 물론 칼로 A라는 글자를 새기는 방식은 아닐 것이다. 메모리는 숫자만을 저장할 수 있기 때문에 A라는 글자의 모양 그대로 비트맵으로 저장하는 방법을 사용하지 않는 한(이 방법은 메모리 낭비가 심하다) 각 문자에 대해서 대응되는 숫자를 정해 놓고 이것을 메모리에 저장하는 방법이 사용될 것이다.
- 영어가 대소문자 합쳐서 52 이므로 6(64가지)비트면 모두 표현할 수 있다.
이를 코드체계라고 한다.
- 000000 → 'a', 000001 → 'b'

문자의 표현

- ✔ 그런데 네트워크가 발전되기 전 미국의 각 지역 별로 코드체계를 정해 놓고 사용했지만
- ✔ 네트워크(인터넷 : 인터넷은 미국에서 발전했다)이 발전하면서 서로간에 정보를 주고 받을 때 정보를 달리 해석한다는 문제가 생겼다.



문자의 표현

- ✓ 그래서 혼동을 피하기 위해 표준안을 만들기로 했다.
- ✓ 바로 이러한 목적으로 1967년, 미국에서 ASCII(American Standard Code for Information Interchange)라는 문자 인코딩 표준이 제정되었다.
- ✓ ASCII는 7bit 인코딩으로 128문자를 표현하며 33개의 출력 불가능한 제어 문자들과 공백을 비롯한 95개의 출력 가능한 문자들로 이루어져 있다.

문자의 표현

✓ 출력 가능 아스키 문자 (32 ~126)

ASCII		ASCII		ASCII		ASCII		ASCII		ASCII	
32		48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(56	8	72	H	88	X	104	h	120	x
41)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	=	77	M	93]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o		

- ✓ 확장 아스키는 표준 문자 이외의 악센트 문자, 도형 문자, 특수 문자, 특수 기호 등 추가적인 문자를 128개 추가할 수 있게 하는 부호이다.
 - 표준 아스키는 7bit를 사용하여 문자를 표현하는 데 비해 확장 아스키는 1B 내의 8bit를 모두 사용함으로써 추가적인 문자를 표현할 수 있다.
 - 컴퓨터 생산자와 소프트웨어 개발자가 여러 가지 다양한 문자에 할당할 수 있도록 하고 있다. 이렇게 할당된 확장 부호는 표준 아스키와 같이 서로 다른 프로그램이나 컴퓨터 사이에 교환되지 못한다.
 - 그러므로 표준 아스키는 마이크로컴퓨터 하드웨어 및 소프트웨어 사이에서 세계적으로 통용되는 데 비해, 확장 아스키는 프로그램이나 컴퓨터 또는 프린터가 그것을 해독할 수 있도록 설계되어 있어야만 올바르게 해독될 수 있다.

문자의 표현

✓ 확장 아스키 예

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
129	81	ü	161	A1	í	193	C1	ł	225	E1	β
130	82	é	162	A2	ó	194	C2	Ṭ	226	E2	Γ
131	83	â	163	A3	ú	195	C3	ṭ	227	E3	π
132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	†	229	E5	σ
134	86	ã	166	A6	ª	198	C6	‡	230	E6	μ
135	87	ç	167	A7	º	199	C7	‡	231	E7	ι
136	88	ê	168	A8	¿	200	C8	Ł	232	E8	Φ
137	89	ë	169	A9	ƒ	201	C9	Ṛ	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	Ł	234	EA	Ω
139	8B	ÿ	171	AB	½	203	CB	Ṛ	235	EB	ϑ
140	8C	î	172	AC	¼	204	CC	‡	236	EC	∞
141	8D	ï	173	AD	ı	205	CD	=	237	ED	∞
142	8E	Ä	174	AE	«	206	CE	‡	238	EE	ε
143	8F	Å	175	AF	»	207	CF	Ł	239	EF	Π
144	90	É	176	B0	☐	208	D0	Ł	240	FO	≡
145	91	æ	177	B1	☐	209	D1	Ṛ	241	F1	±
146	92	Æ	178	B2	☐	210	D2	Ṛ	242	F2	≥
147	93	ô	179	B3		211	D3	Ł	243	F3	≤
148	94	ö	180	B4	†	212	D4	Ł	244	F4	[
149	95	ò	181	B5	‡	213	D5	Ṛ	245	F5]
150	96	û	182	B6	‡	214	D6	Ṛ	246	F6	÷
151	97	ù	183	B7	Ṛ	215	D7	‡	247	F7	≈
152	98	ý	184	B8	Ṛ	216	D8	‡	248	F8	•
153	99	Ö	185	B9	‡	217	D9	Ṛ	249	F9	•
154	9A	Û	186	BA		218	DA	Ṛ	250	FA	•
155	9B	÷	187	BB	Ṛ	219	DB	■	251	FB	√
156	9C	£	188	BC	Ṛ	220	DC	■	252	FC	∞
157	9D	¥	189	BD	Ṛ	221	DD	■	253	FD	z
158	9E	℔	190	BE	Ṛ	222	DE	■	254	FE	■
159	9F	ƒ	191	BF	Ṛ	223	DF	■	255	FF	□

문자의 표현

- ✓ 오늘날 대부분의 컴퓨터는 문자를 읽고 쓰는데 ASCII형식을 사용한다.
- ✓ 그런데 컴퓨터가 발전하면서 미국 뿐 아니라 각 나라에서도 컴퓨터가 발전했으며 각 국가들은 자국의 문자를 표현하기 위하여 코드체계를 만들어서 사용하게 되었다.
 - 우리나라도 아주 오래된 이야기지만 한글 코드체계를 만들어 사용했고 조합형, 완성형 두 종류를 가지고 있었다.

문자의 표현

- ✓ 인터넷이 전 세계로 발전하면서 ASCII를 만들었을 때의 문제와 같은 문제가 국가간에 정보를 주고 받을 때 발생했다.
- ✓ 자국의 코드체계를 타 국가가 가지고 있지 않으면 정보를 잘못 해석 할 수 밖에 없었다.
- ✓ 그래서 다국어 처리를 위해 표준을 마련했다 이를 유니코드라고 한다.

문자의 표현

✓ 유니코드의 일부

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0004	DC4 0014	\$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074

	B30	B31	B32	B33	B34	B35	B36	B37
0	대 B300	데미 B310	단 B320	닷 B330	델 B340	덱 B350	덜 B360	테 B370
1	दै B301	दै B311	दै B321	दै B331	दै B341	दै B351	दै B361	दै B371
2	दै B302	दै B312	दै B322	दै B332	दै B342	दै B352	दै B362	दै B372
3	दै B303	दै B313	दै B323	दै B333	दै B343	दै B353	दै B363	दै B373
4	दै B304	दै B314	दै B324	दै B334	दै B344	दै B354	दै B364	दै B374

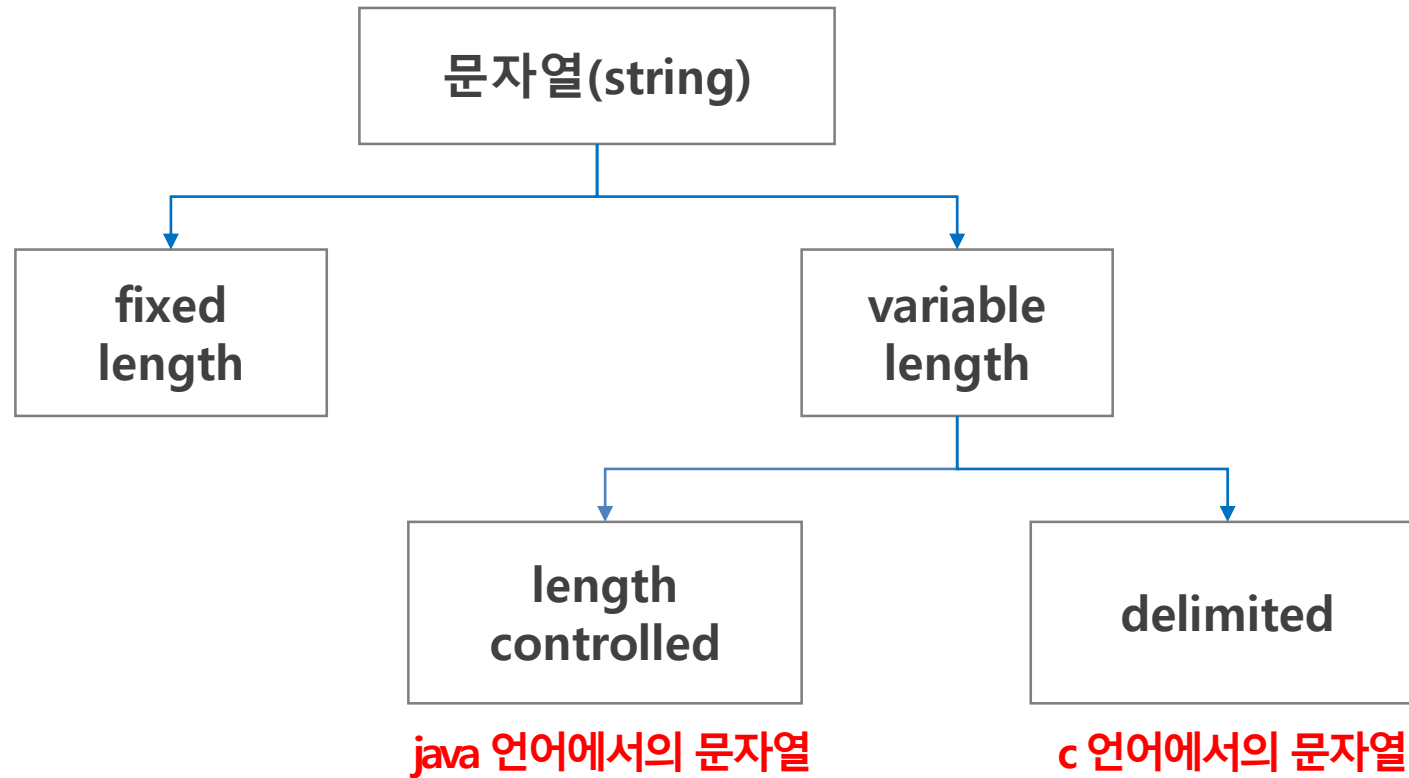
✓ 유니코드도 다시 Character Set으로 분류된다.

- UCS-2(Universal Character Set 2)
- UCS-4(Universal Character Set 4)
- 유니코드를 저장하는 변수의 크기를 정의
- 그러나, 바이트 순서에 대해서 표준화하지 못했음.
- 다시 말해 파일을 인식 시 이 파일이 UCS-2, UCS-4인지 인식하고 각 경우를 구분해서 모두 다르게 구현해야 하는 문제 발생
- 그래서 유니 코드의 적당한 외부 인코딩이 필요하게 되었다.

✓ 유니코드 인코딩 (UTF : Unicode Transformation Format)

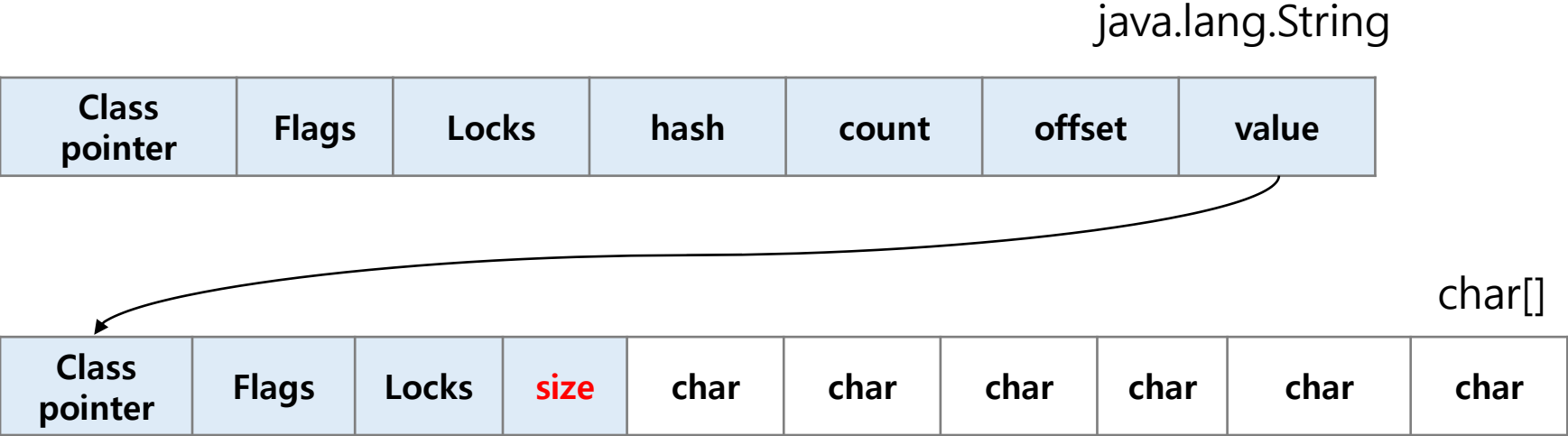
- UTF-8 (in web)
 - MIN : 8bit, MAX: 32bit(1 Byte * 4)
- UTF-16 (in windows, java)
 - MIN : 16bit, MAX: 32bit(2 Byte * 2)
- UTF-32 (in unix)
 - MIN : 32bit, MAX: 32bit(4 Byte * 1)

✓ 문자열의 분류



✓ java에서 String 클래스에 대한 메모리 배치 예

- 그림에서 보이듯, java.lang.String 클래스에는 기본적인 객체 메타 데이터 외에도 네 가지 필드들이 포함되어 있는데, hash값(hash), 문자열의길이(count), 문자열 데이터의 시작점(offset), 그리고 실제 문자열 배열에 대한 참조(value)이다.



✓ C언어에서 문자열 처리

- 문자열은 문자들의 배열 형태로 구현된 응용 자료형
- 문자배열에 문자열을 저장할 때는 항상 마지막에 끝을 표시하는 널문자('\0')를 넣어줘야 한다.

```
char ary[]={ 'a', 'b', 'c', '\0' }; // 또는 char ary[]="abc";
```

- 문자열 처리에 필요한 연산을 함수 형태로 제공한다.

```
strlen(), strcpy(), strcmp(),...
```

✓ Java(객체지향 언어)에서의 문자열 처리

- 문자열 데이터를 저장, 처리해주는 클래스를 제공한다.
- String클래스를 사용한다.

```
String str="abc"; //또는 String str = new String("abc")
```

- 문자열 처리에 필요한 연산을 연산자, 메소드 형태로 제공한다.
 - +, length(), replace(), split(), substring(), ...
 - 보다 풍부한 연산을 제공한다.

✓ C와 Java의 String 처리의 기본적인 차이점

- c는 아스키 코드로 저장한다.
- java는 유니코드(UTF16, 2byte)로 저장한다.
- 파이썬은 유니코드(UTF8)로 저장

C

```
char * name = "홍길동";  
int count = strlen(name);  
printf("%d", count);
```

6이 출력된다.

Java

```
String name = "홍길동";  
System.out.println(name.length());
```

3이 출력된다.

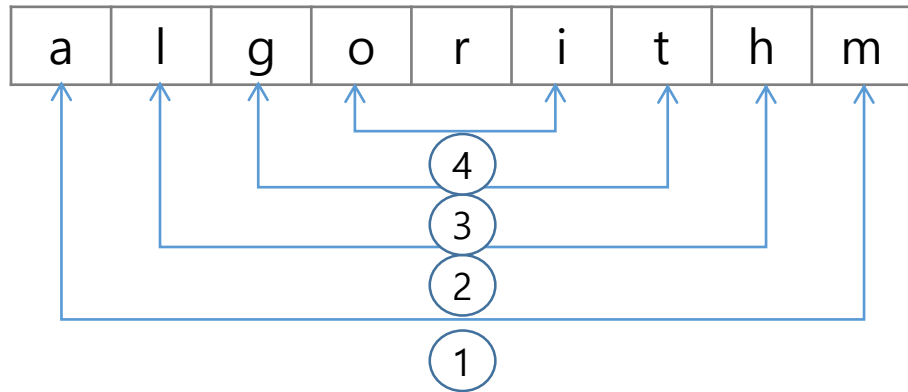
Python

```
name = "홍길동"  
print(len(name))
```

3이 출력된다.

문자열 뒤집기

- ✓ 자기 문자열에서 뒤집는 방법이 있고 새로운 빈 문자열을 만들어 소스의 뒤에서부터 읽어서 타겟에 쓰는 방법이 있겠다.
- ✓ 자기 문자열을 이용할 경우는 swap을 위한 임시 변수가 필요하며 반복 수행을 문자열 길이의 반만을 수행해야 한다.



문자열 길이 9
 $9 / 2 = 4.5$
4회 반복

연습문제 1

- ✓ c에서는 앞의 알고리즘 대로 구현해야 한다.
- ✓ java에서는 StringBuffer 클래스의 reverse() 메소드를 이용하면 된다.
- ✓ 구현해봅시다.

문자열 비교

- ✓ c strcmp() 함수를 제공한다.
- ✓ Java에서는 equals() 메소드를 제공한다.
 - 문자열 비교에서 == 연산은 메모리 참조가 같은지를 묻는 것
- ✓ python에서는 == 연산자와 is 연산자를 제공한다.
 - == 연산자는 내부적으로 특수 메서드 __eq__()를 호출

java

```
String s1 = "Hi";
String s2 = "Hello";
if(s1.equals(s2)==true){
    System.out.println("Equal");
} else if(s1.equals(s2)==false){
    System.out.println("Not Equal");
}
```

문자열 숫자를 정수형으로 변환

- ✓ c 언어에서는 `atoi()` 함수를 제공한다. 역 함수로는 `itoa()`가 있다.
- ✓ java에서는 숫자 클래스의 `parse` 메소드를 제공한다.
 - 예 : `Integer.parseInt(String)`
 - 역함수로는 `toString()` 메소드를 제공한다.
- ✓ python에서는 숫자와 문자변환 함수를 제공한다.
 - ex) `int("123")`, `float("3.14")`, `str(123)`, `repr(123)`

문자열 숫자를 정수형으로 변환

✓ atoi()

Java

```
int atoi(char[] charArr) {  
    int value = 0;  
    int digit;  
    for (int i = 0; i < charArr.length; i++) {  
        if (charArr[i] >= '0' && charArr[i] <= '9')  
            digit = charArr[i] - '0';  
        else {  
            break;  
        }  
        value = (value * 10) + digit;  
    }  
    return value;  
}
```

연습문제 2

✓ itoa()를 구현해 봅시다.

- 양의 정수를 입력 받아 문자열로 변환하는 함수
 - 입력 값 : 변환할 정수 값, 변환된 문자열을 저장할 문자배열
 - 반환 값 : 없음
-
- 음수를 변환할 때는 어떤 고려 사항이 필요한가요?

패턴 매칭

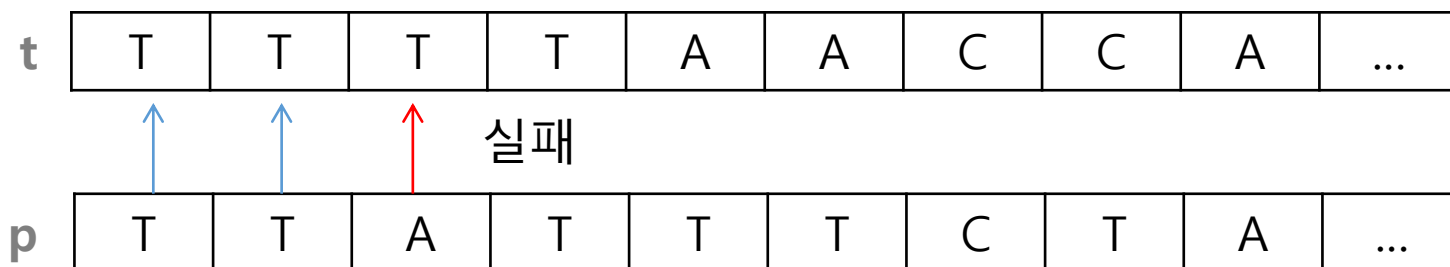
✓ 패턴 매칭에 사용되는 알고리즘들

- 고지식한 패턴 검색 알고리즘
- 카프-라빈 알고리즘
- KMP 알고리즘
- 보이어-무어 알고리즘

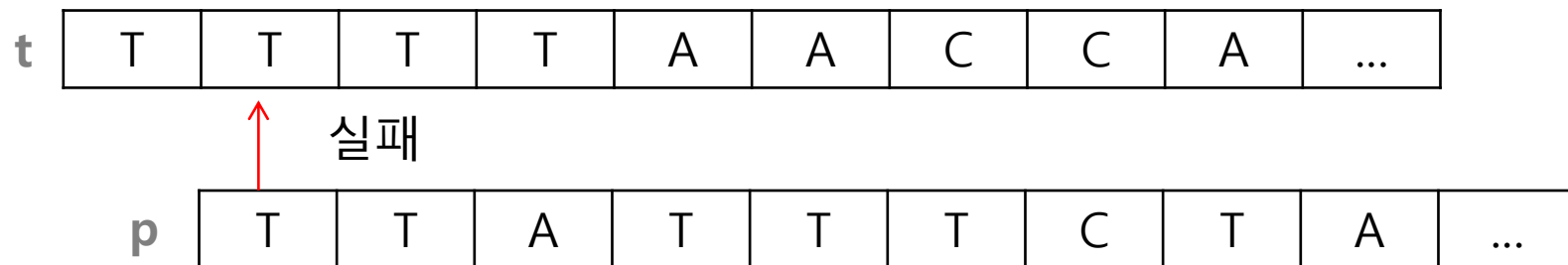
패턴 매칭

❑ 고지식한 알고리즘(Brute Force)

- 본문 문자열을 처음부터 끝까지 차례대로 순회하면서 패턴 내의 문자들을 일일이 비교하는 방식으로 동작



한 칸 이동, 비교



패턴 매칭

알고리즘 설명

t T T T T A A C C A ...
index i ↑

p T T A T T T C T A ...
index j ↑



t T T T T A A C C A ...
index i ↑

p T T A T T T C T A ...
index j ↑



패턴 매칭



✓ 고지식한 패턴 검색 알고리즘

```
// p[] : 찾을 패턴 - iss  
// t[] : 전체 텍스트 - This iss a book  
// M : 찾을 패턴의 길이  
// N : 전체 텍스트의 길이  
// i : t의 인덱스  
// j : p의 인덱스
```

```
BruteForce(char[] p, char[] t){  
    i ← 0, j ← 0  
    while(j < M and i < N) {  
        if(t[i] != p[j])  
            i ← i - j;  
        j ← -1;  
        i ← i + 1, i ← j + 1  
    }  
    if (j == M) return i - M;  
    else return -1;  
}
```


✓ 고지식한 패턴 검색 알고리즘의 시간 복잡도

- 최악의 경우 시간 복잡도는 텍스트의 모든 위치에서 패턴을 비교해야 하므로 $O(MN)$ 이 됨
- 예에서는 최악의 경우 약 $10,000 * 80 = 800,000$ 번의 비교가 일어난다.
- 비교횟수를 줄일 수 있는 방법은 없는가?

보이어 무어 알고리즘

- ✓ 오른쪽에서 왼쪽으로 비교
- ✓ 대부분의 상용 소프트웨어에서 채택하고 있는 알고리즘
- ✓ 보이어-무어 알고리즘은 패턴에 오른쪽 끝에 있는 문자가 불일치 하고 이 문자가 패턴 내에 존재하지 않는 경우, 이동 거리는 무려 패턴의 길이 만큼이 된다.

T[]

.	b	w	a	t	e	r	.	.
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

P[]

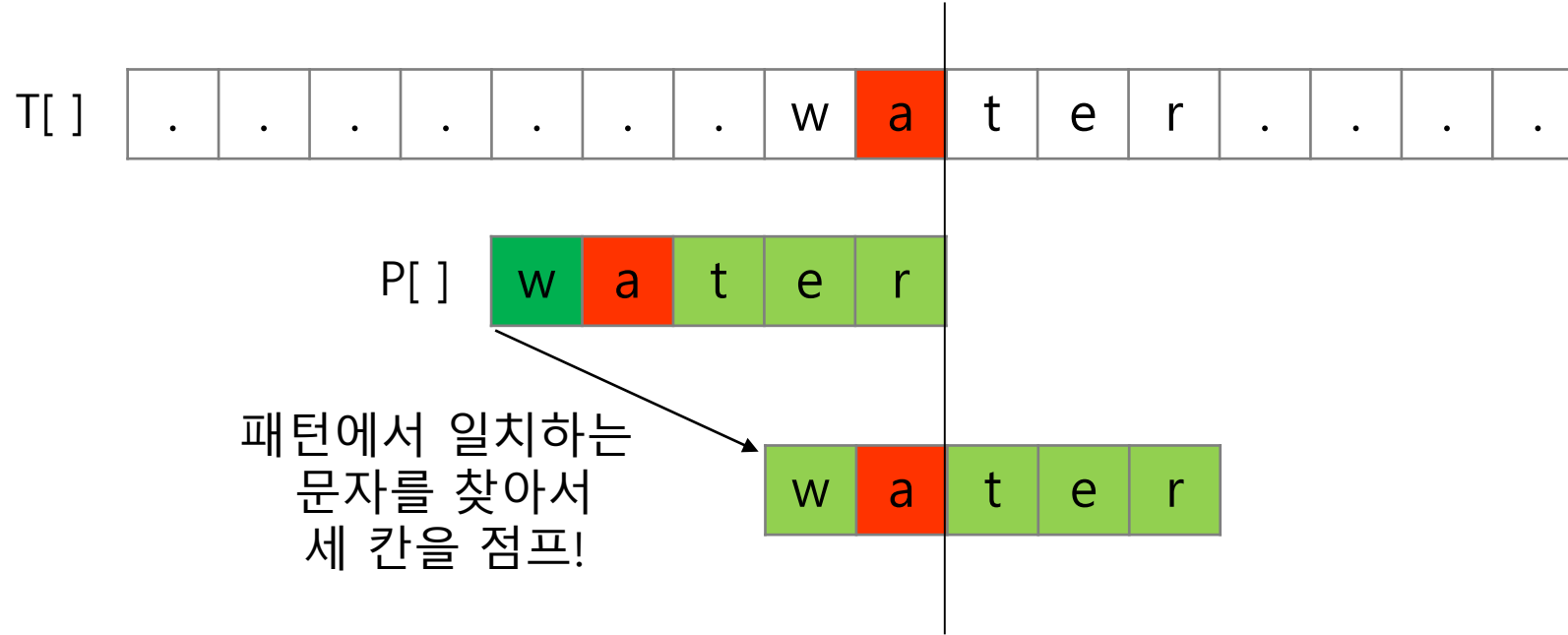
w	a	t	e	r
---	---	---	---	---

다섯칸 한꺼번에
점프!

w	a	t	e	r
---	---	---	---	---

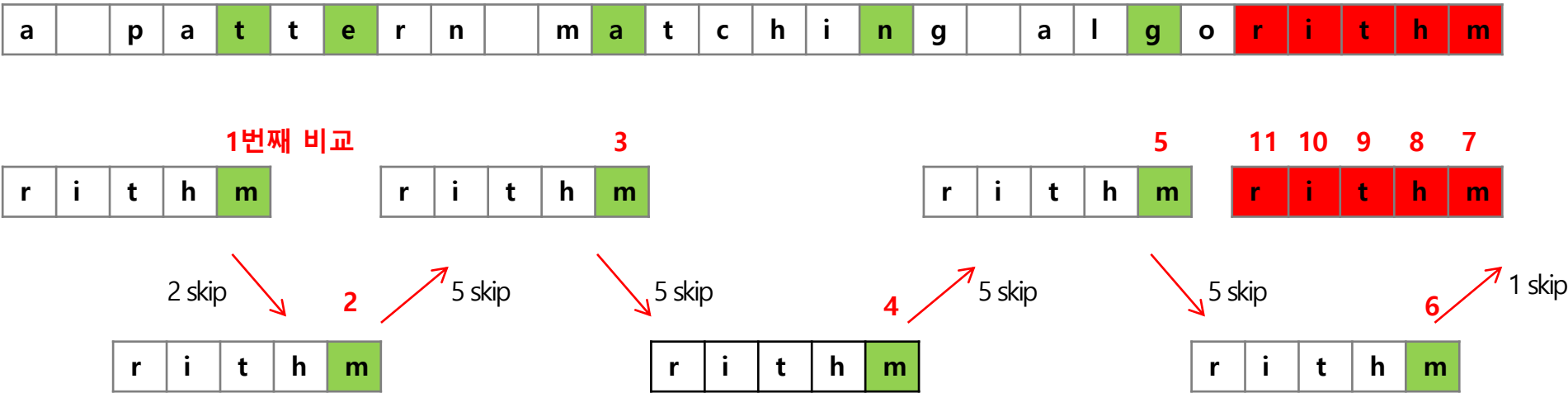
보이어 무어 알고리즘

- ✔ 오른쪽 끝에 있는 문자가 불일치 하고 이 문자가 패턴 내에 존재할 경우



보이어 무어 알고리즘

✓ 보이어-무어 알고리즘을 이용한 예



■ rithm 문자열의 skip 배열

m	h	t	i	r	다른 모든 문자
0	1	2	3	4	5

보이어 무어 알고리즘

✓ 문자열 매칭 알고리즘 비교

- 찾고자 하는 문자열 패턴의 길이 m , 총 문자열 길이 n
- 고지식한 패턴 검색 알고리즘 : 수행시간 $O(mn)$
- 카프-라빈 알고리즘 : 수행시간 $\Theta(n)$
- KMP 알고리즘 : 수행시간 $\Theta(n)$
- 보이어-무어 알고리즘
 - 앞의 두 매칭 알고리즘들의 공통점 텍스트 문자열의 문자를 적어도 한번씩 훑는다는 것이다. 따라서 최선의 경우에도 $\Omega(n)$
 - 보이어-무어 알고리즘은 텍스트 문자를 다 보지 않아도 된다
 - 발상의 전환 : 패턴의 오른쪽부터 비교한다
 - 최악의 경우 수행시간 : $O(mn)$
 - 입력에 따라 다르지만 일반적으로 $\Theta(n)$ 보다 시간이 덜 든다

연습문제 3

- ✓ 고지식한 방법을 이용하여 패턴을 찾아 봅시다.
- ✓ 임의의 본문 문자열과 찾을 패턴 문자열을 만듭니다.
- ✓ 결과 값으로 찾은 위치 값을 결과로 출력합니다.

문자열 암호화

✓ 시저 암호(Caesar cipher)

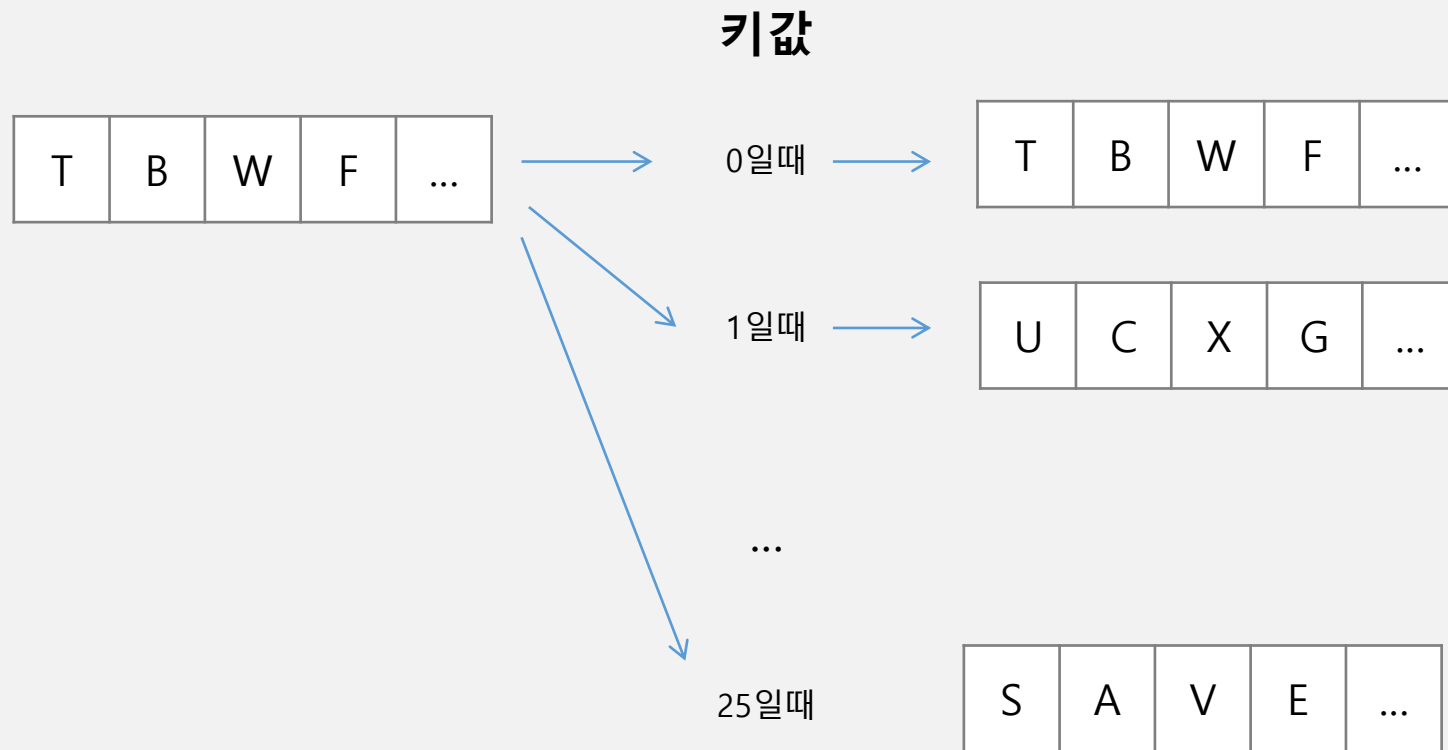
- 줄리어스 시저가 사용했다고 하는 암호이다.
- 시저는 기원전 100년경에 로마에서 활약했던 장군이었다.
- 시저 암호에서는 평문에서 사용되고 있는 알파벳을 일정한 문자 수만큼 [평행이동] 시킴으로써 암호화를 행한다.

✓ 1 만큼 평행했다는 카이사르 암호화의 예

평 문	S	A	V	E		P	R	I	V	A	T	E		R	Y	A	N
암호문	T	B	W	F	A	Q	S	J	W	B	U	F	A	S	Z	B	O

- ✓ 1만큼 평행했을 때 1을 키값이라 한다.
- ✓ 수신자 이외의 사람(키가 1이라는 사실을 모르는 사람)이 암호문 TBWFAQSJWBUFASZBO를 보고 다른 정보 없이도 SAVE PRIVATE RYAN 라는 메시지를 맞출 수는 없을까?
- ✓ 다시 말해, 시저 암호를 해독할 수 없을까?

✓ 시저 암호문에 대한 전사공격



✔ 문자 변환표를 이용한 암호화(단일 치환 암호)

- 단순한 카이사르 암호화보다 훨씬 강력한 암호화 기법

▪ 문자 변환표의 예

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Q	H	C	B	E	J	K	A	R	W	S	T	U	V	D		I	O	P	X	Z	F	G	L	M	N	Y

▪ 위 변환표를 사용한 암호화의 예

평 문	S	A	V	E		P	R	I	V	A	T	E		R	Y	A	N
암호문	X	H	G	J	Q	I	P	W	G	H	Z	J	Q	P	N	H	D

✓ 단일 치환 암호의 복호화

- 복호화 하기 위해서는 모든 키의 조합(key space)가 필요하다.

✓ 단일 치환 암호의 키의 총수는

- $26 \times 25 \times 24 \times 23 \times \dots \times 1 = 26! = 403291461126605635584000000$

- ✓ 1초에 10억 개의 키를 적용하는 속도로 조사한다고 해도, 모든 키를 조사하는데 120억년 이상의 시간이 걸린다. 방법이 없을까?
있다! 관심 있으면 찾아보길...

문자열 암호화

✓ bit열의 암호화

- 배타적 논리합(exclusive-or) 연산 사용

x	XOR	y
0	0	0
0	1	1
1	0	1
1	1	0

암호화

평 문	1	0	0	1	1	0	0	0	1	1	1	0
키	1	1	0	0	0	1	1	1	0	0	1	1
암호문	0	1	0	1	1	1	1	1	1	1	0	1

해 독

암호문	0	1	0	1	1	1	1	1	1	1	0	1
키	1	1	0	0	0	1	1	1	0	0	1	1
평 문	1	0	0	1	1	0	0	0	1	1	1	0

✓ bit열의 암호화

■ C코드

C

```
void Bbit_print(char a)
{
    int i;

    for(i=7; i>=0; i--)
        putchar((a & (1 << i)) ? '1' : '0');
    putchar(' ');
}
main(void)
{
    char a=0x86;
    char key=0xAA;

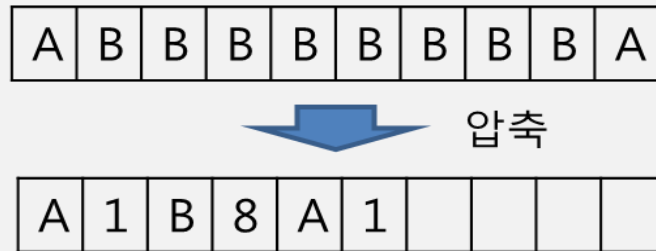
    printf("a      ==> ");
    Bbit_print(a);      putchar('\n');

    printf("a^=key ==> ");
    a^=key;
    Bbit_print(a);      putchar('\n');

    printf("a^=key ==> ");
    a^=key;
    Bbit_print(a);      putchar('\n');
}
```

✓ 다음과 같은 문자열이 있다. 저장소의 크기를 줄이며 정확한 정보를 저장하는 방법은?

- Run-length encoding 알고리즘
- 같은 값이 몇 번 반복되는가를 나타냄으로써 압축



- 이 방법은 이미지 파일포맷 중 BMP 파일포맷의 압축 방법이다.
- 좀 더 효율적이고 일반적인 압축방법은 없는가?
 - 있다. 많이 사용하는 알고리즘으로 허프만 코딩 알고리즘이 있다.

다음 방송에서 만나요!

삼성 청년 SW 아카데미