2024 자료구조

[과제 1 배열]

과목명 : 자료구조

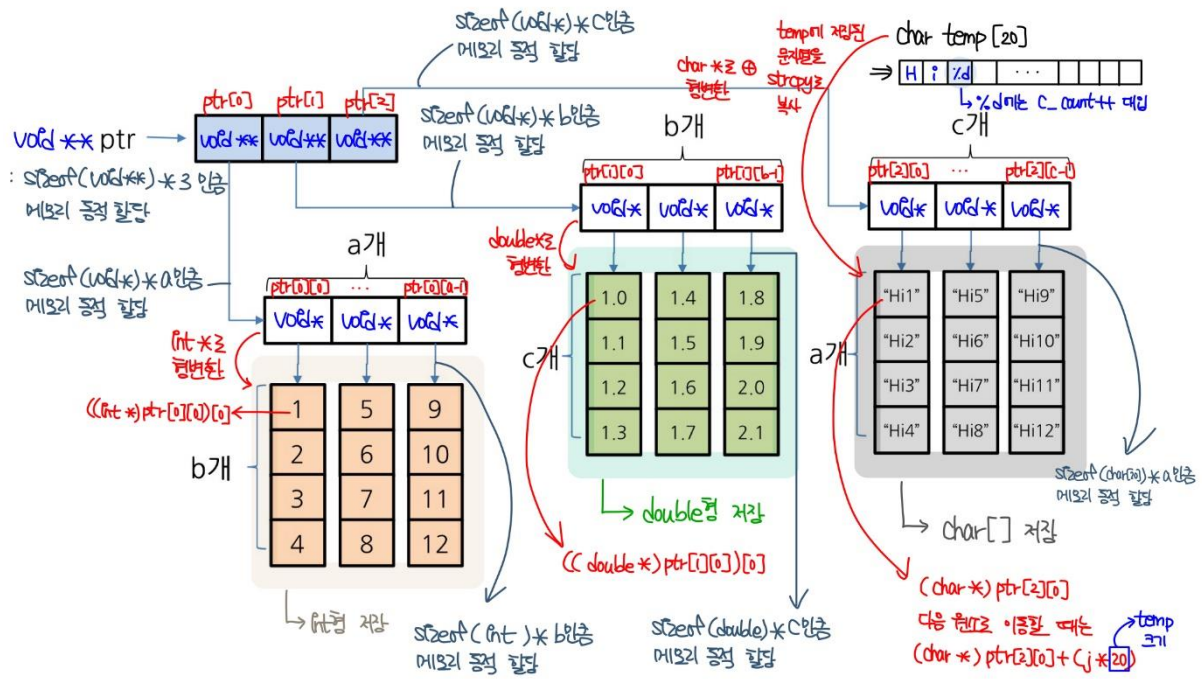
학과 : 소프트웨어학과

학번 : 20231371

이름 : 김혜린

Ⅰ 자료구조 만들기

1. 문제 내용과 해결 방안



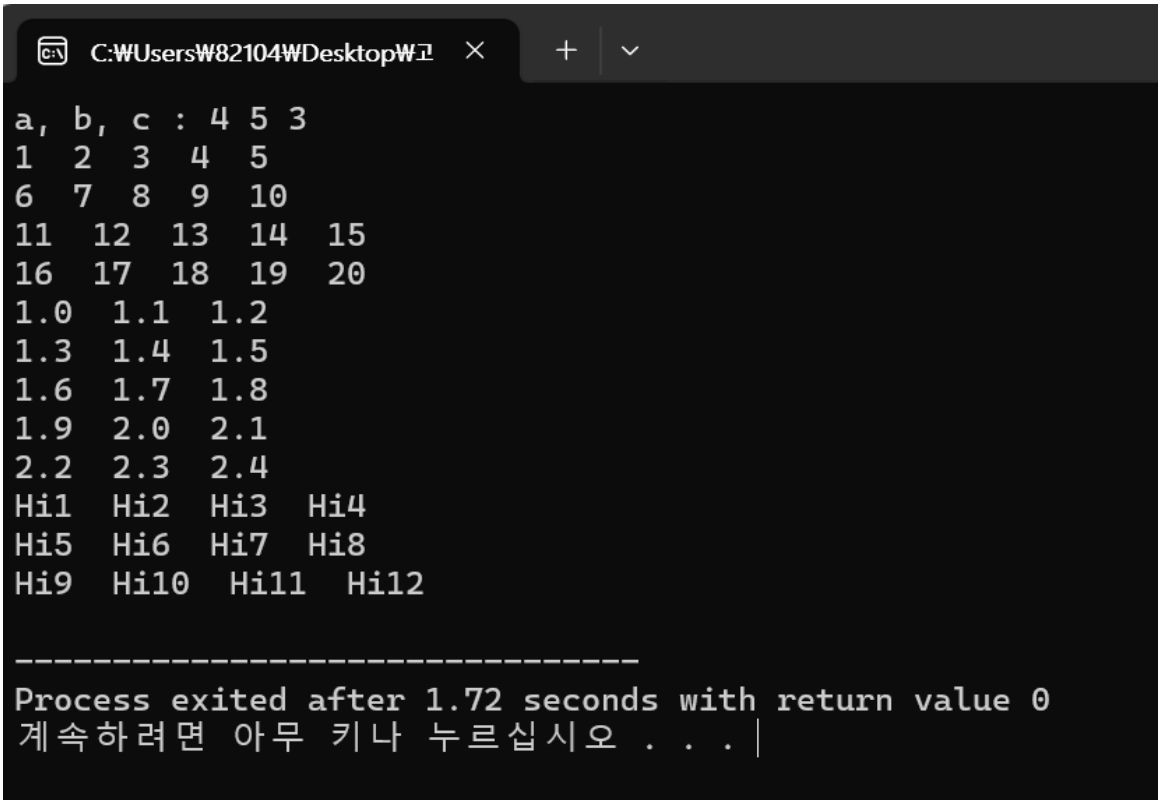
2. 실행결과

// a, b, c에 각각 2, 3, 2 넣었을 때

```
C:₩Users₩82104₩Desktop₩고
a, b, c : 2 3 2
1 2 3
4 5 6
1.0 1.1
1.2 1.3
1.4 1.5
Hi1 Hi2
Hi3 Hi4

--------------------------------
Process exited after 1.719 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

// a, b, c에 각각 4, 5, 3 넣었을 때



Ⅱ -1 파이썬의 List 내부 구조 파악하기

1. 생성과 사용

리스트명 = [요소1, 요소2, 요소3, ...] // 요솟값들을 쉼표로 구분하여 나열하고 대괄호로 감쌈

- 리스트 안의 각 요소는 어떤 자료형이든 가능 (비어 있는 리스트는 a =list()로 생성 가능)

2. 인덱싱과 슬라이싱

① 인덱싱 : 리스트의 요솟값 각각에 접근 가능 (ex. a = [1, 2, 3])

리스트명[참조할 요소의 자리] // 첫번째 원소 a[0], 두번째 원소 a[1], ...

- a[-1] = 마지막 원소
- 이중 리스트 (ex. a = [1, 2, 3, ['a', 'b', 'c']] //삼중도 같은 원리

→ a[-1][0] = 'a'

② 슬라이싱 : 리스트 쪼개기 (ex. a = [1, 2, 3, 4, 5])

리스트명[시작위치 : 끝위치]　　//시작위치 요소는 포함, 끝위치 원소는 포함 안 함

·◌시작위치 없음 = 처음 원소부터　(ex. b = a[: 2] >>> b = [1, 2])

·◌끝위치 없음 = 마지막 원소까지　(ex. c = a[2 :] >>> c = [3, 4, 5])

·◌이중 리스트 (ex. a = [1, 2, 3, ['a', 'b', 'c'], 4, 5]일 때, a[3][: 2] >>> ['a', 'b']

3. 연산

더하기 (+)	리스트1 + 리스트2 : 2개의 리스트 합침
반복하기 (*)	리스트명 * n : 리스트 내의 원소 n번 반복
리스트 길이 구하기	len (리스트명)

4. 수정과 삭제

① 수정 : 인덱싱 후 원하는 값 대입

ex) >>> a = [1, 2, 3]

>>> a[2] = 4

>>> a

[1, 2, 4]

② 삭제 (del 함수 사용)

del 리스트명 [없애려는 요소 순서]　　//del a[x] : 리스트 a의 x번째 요솟값 삭제

·◌슬라이싱 사용하여 리스트의 여러 요소 한꺼번에 삭제 가능

ex) >>> a = [1, 2, 3, 4, 5]

>>> del a [2 :]

>>> a

[1, 2]

5. 관련 함수 : 리스트명 뒤에 '.'를 붙여 사용

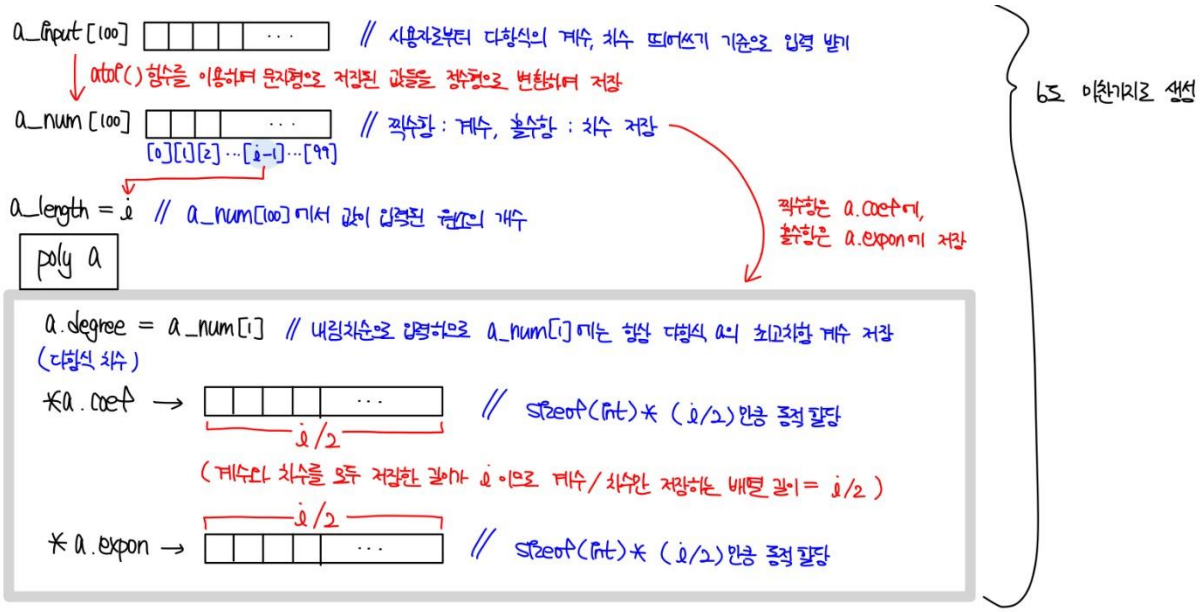
append (요소 추가)	리스트 . append(x) : 리스트의 맨 마지막에 x 추가
sort (정렬)	리스트 . sort() : 숫자 오름차순, 문자 알파벳 순
reverse (역순)	리스트 . reverse() : 순서대로 정렬 후 역순이 아니라 현재 리스트 그대로 거꾸로 뒤집음
index (인덱스 반환)	리스트 . index (x) : 리스트에 x가 있으면 x의 인덱스 값 리턴 (없으면 오류 발생)
insert (요소 삽입)	리스트 . insert (a, b) : a번째 위치에 b 삽입
remove (요소 제거)	리스트 . remove(x) : 리스트에서 첫번째로 나오는 x만 삭제
pop (요소 끄집어내기)	리스트 . pop() : 리스트의 맨 마지막 요소 리턴 후 삭제
count (요소 x의 개수 세기)	리스트 . count(x) : 리스트 안의 x의 개수 리턴
extend (확장)	리스트1 . extend (리스트2) : 리스트1에 리스트2 더함

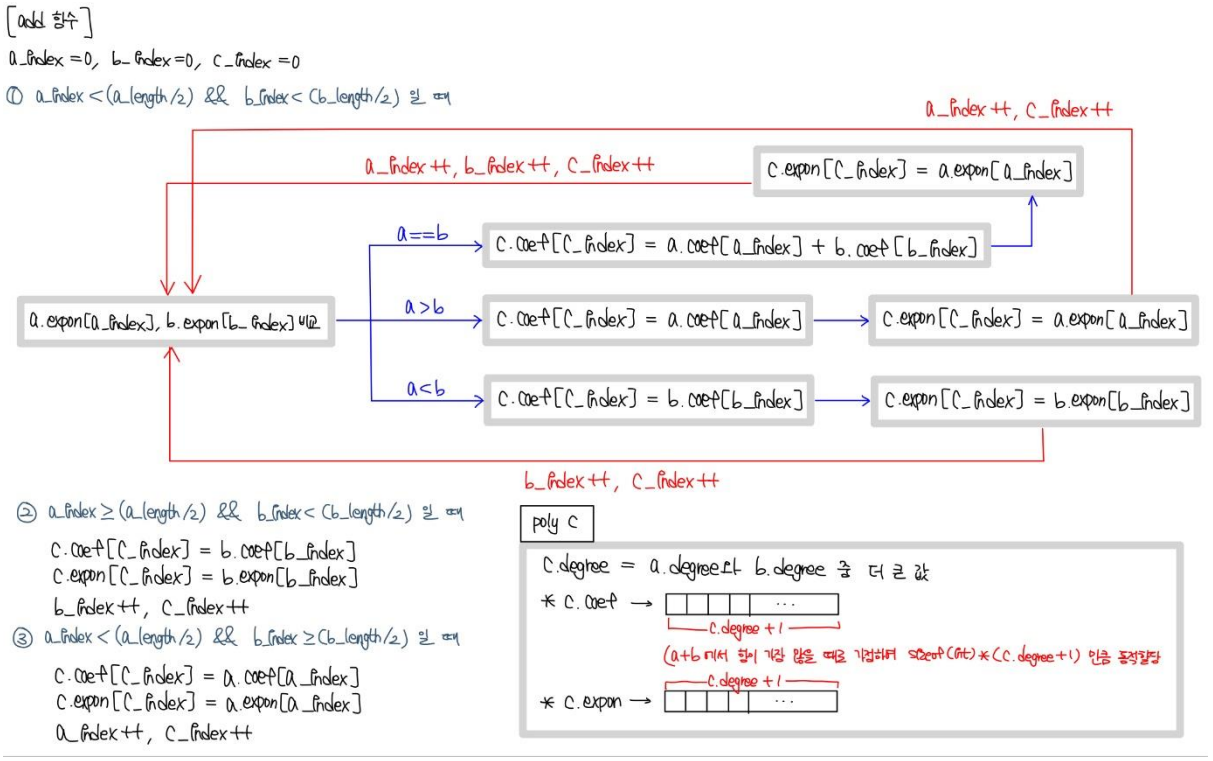
Ⅱ-2 파이썬 List의 장점

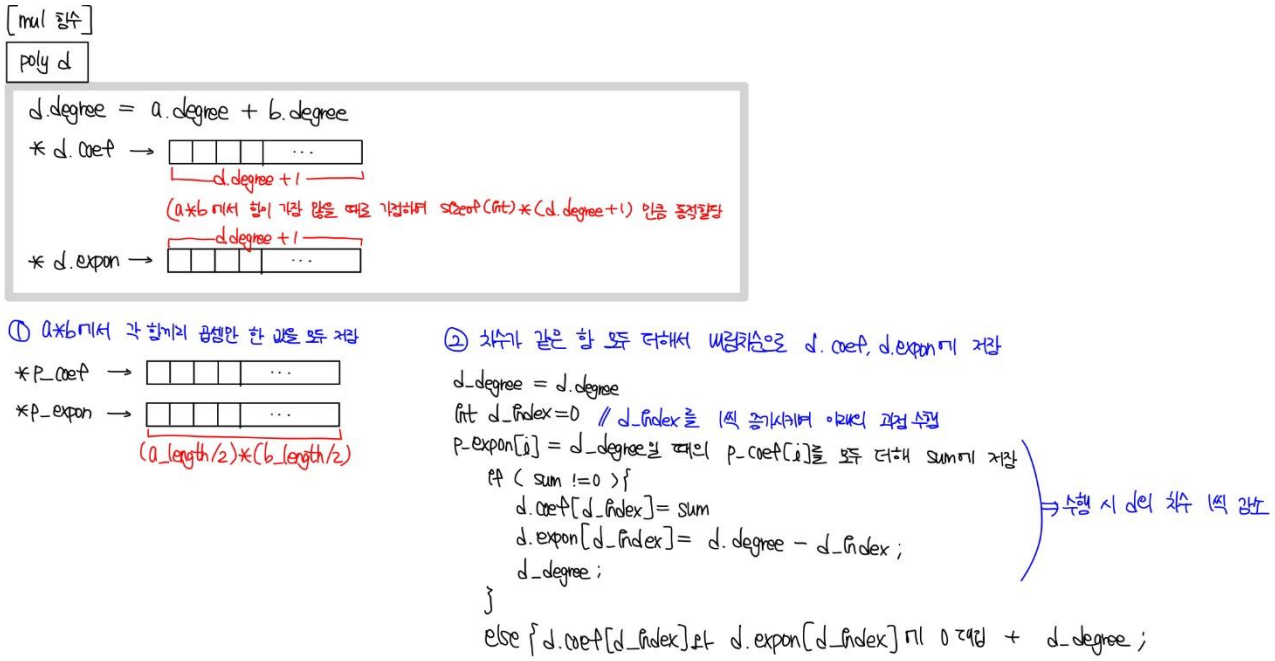
크기조절 가능	·○List는 필요에 따라 크기를 동적으로 조절 가능 ·○새로운 요소가 추가 시 메모리를 할당, 삭제 시 메모리 해제 ∴ 효율적인 메모리 관리
데이터 접근 및 조작에 편리	·○인덱싱과 슬라이싱으로 데이터에 접근하고 조작하는 게 편리함 ·○요소의 추가, 삭제, 수정 가능
다양한 연산	더하기(+), 반복하기(*) 등의 연산과 여러 관련 함수가 있음
모든 자료형 가능	리스트의 요소는 모든 자료형이 가능함

Ⅲ 다항식의 저장 방식 개선하기

1. 문제 내용과 해결 방안

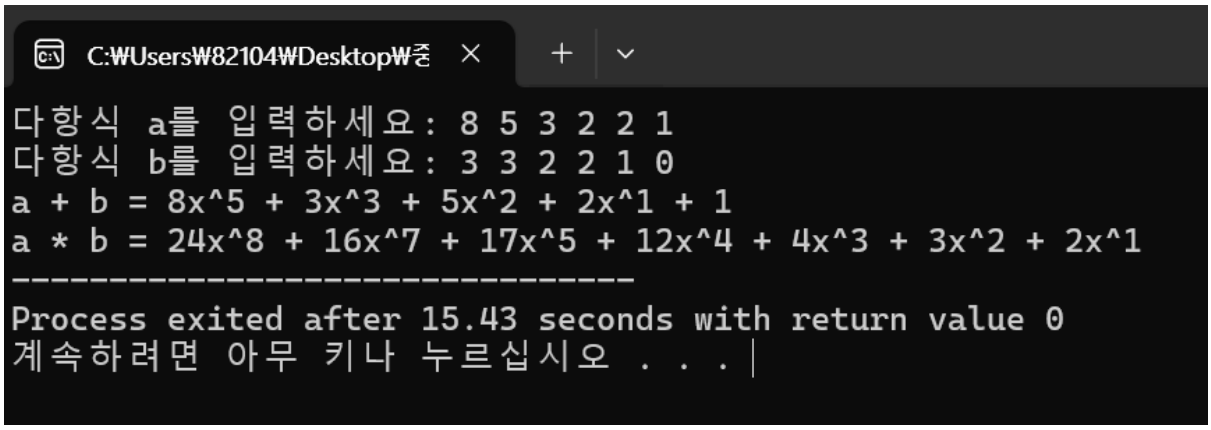






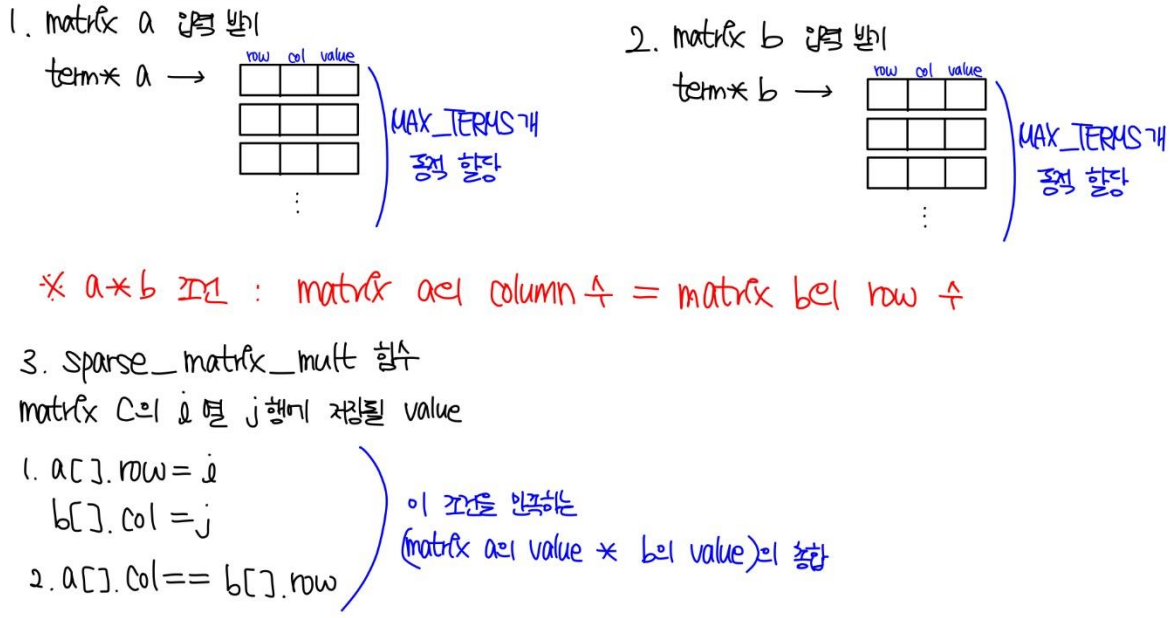
2. 실행결과

// a = 8 5 3 2 2 1, b = 3 3 2 2 1 0 입력했을 때



IV 희소 행렬 저장 방식의 곱셈

1. 문제 내용과 해결 방안



2. 실행결과

// matrix a : 9 X 10, matrix b : 10 X 13일 때

```
C:₩Users₩82104₩Desktop₩중
저장 멈춤 : value 뒤 '.'
matrix a
가로, 세로의 크기 : 9 10
0이 아닌 원소의 row column value
= 0 0 1
= 4 7 1
= 8 2 1.
matrix b
가로, 세로의 크기 : 10 13
0이 아닌 원소의 row column value
= 0 11 1
= 4 7 1
= 8 2 1.
matrix c
: (0, 11, 1)

--------------------------------
Process exited after 232.1 seconds with return value 3221226356
계속하려면 아무 키나 누르십시오 . . .
```

V 미로 표현하기

```
FILE *fp;                       // 파일 포인터
char maze[MAX][MAX];            // 파일에서 읽어온 문자를 저장할 2차원 배열
int com_maze[MAX][MAX];         // 문자를 정수로 변환하여 저장할 2차원 배열
int r_path, c_path;             // 각각 가로 길 수, 세로 길 수
int rows, cols;                 // 미로의 가로크기, 세로크기
```

1. 미로의 크기 파악하기

maze1.txt의 첫 부분의 정수 2개(미로의 규격)를 fscanf를 이용하여 읽어온 뒤 r_path, c_path에 저장한다. 가로의 크기 rows에 (r_path*2)+1을, 세로의 크기 cols에는 (c_path*2)+1을 대입한다. 현재 maze1.txt파일의 가로 크기는 19, 세로 크기는 25가 된다.

2. 문자를 1,0으로 바꾸어 19개씩 묶어서 int형 배열에 저장

미로를 저장할 배열 int a를 선언한다. //int a[23]

|와 +와 -는 주변 상황을 통해 구별할 수 있기 때문에 같은 1로 변환한다. ' '은 지나갈 수 있는 길이므로 0으로 변환한다. 이렇게 i번째 줄을 변환한 값들을 a[i]에 19개씩 묶어서 저장한다. int형은 최대 –(2^31) ~ (2^31)-1까지 저장할 수 있으므로 1,0으로 구성된 19자리 수(미로 가로 크기)를 저장할 수 있다. 이 변수 하나당 미로 가로 한 줄을 의미한다.

배열의 크기를 23으로 선언하는 이유는 미로의 가장 외벽은 무조건 벽이므로 1111111111111111111이 저장될 것이기 때문에 따로 저장하지 않는다.

예를 들어 미로의 맨 윗 줄(가장 외벽을 제외한)이 '| | | |'로 구성되어 있다면 int a[0] = 1000000010000000101으로 저장한다. 이렇게 하면 크기가 23인 int형 배열로 미로를 저장할 수 있다. 이때의 저장공간은 92바이트가 된다.