제16회 임베디드 소프트웨어 경진대회
The World Embedded Software Contest 2018

FUNERS

주니어 임베디드SW 챌린저

-3차 기술 지원 교육-

2018. 08. 18

INDEX

FUNERS

색상 정보 저장하기

색상 정보 저장하기

FUNERS

- 2차 배열을 사용하여 저장
  - (x, y)의 형태로 위치의 특정이 가능
- 안쪽과 바깥쪽의 두가지 정보를 저장해야 한다.
  - 2개의 2차 배열을 만들어서 각각 저장
  - 튜플을 이용해 배열 한 칸을 (바깥 색상, 안쪽 색상)으로 저장
  - 두 자리 수를 이용하여 10의 자리는 바깥쪽 1의 자리는 안쪽 색상으로 표현
  - 색상 정보를 가지는 클래스를 만들어 배열의 각 칸에 저장

색상 정보 저장하기

FUNERS

- 2개의 배열

R	Y	G	Y	G
B	G	Y	G	B
R	G	Y	R	G
B	Y	G	B	R
B	G	G	B	R

바깥 색상

Y	R	B	Y	B
B	Y	Y	R	B
R	B	B	R	B
Y	Y	B	B	R
B	R	R	B	Y

안쪽 색상

색상 정보 저장하기

FUNERS

- 튜플 배열

(바깥 색상, 안쪽 색상)

(R,B)	(G,B)	(Y,Y)	(Y,B)	(G,B)
(R,R)	(G,R)	(B,Y)	(B,G)	(G,R)
(Y,R)	(Y,Y)	(Y,B)	(R,B)	(R,G)
(Y,B)	(B,R)	(G,Y)	(G,R)	(R,Y)
(B,B)	(G,B)	(B,B)	(G,B)	(R,R)

색상 정보 저장하기

FUNERS

- 두 자리 수 숫자 배열

1 – 빨강
2 – 파랑
3 – 노랑
4 – 초록

11	23	21	13	32
14	33	22	23	42
32	23	43	32	41
21	42	33	12	12
22	31	41	14	31

색상 정보 저장하기

FUNERS

- 두 자리 수 숫자 배열

클래스 Store
안쪽색상
바깥색상

store	store	store	store	store
store	store	store	store	store
store	store	store	store	store
store	store	store	store	store
store	store	store	store	store

색상 정보 저장하기

FUNERS

- **클래스(class) – 설계도, 청사진, 형틀**
  - 객체지향 언어에서의 핵심 요소 중 하나
  - 객체란 세계를 구성하는 요소들의 정의
  - 프로그램의 런(Run)타임에서 객체(object)를 만들기 위한 설계도의 역할
  - 변수, 메소드, 클래스 등을 포함한다.
- **좋은 점?**
  - 필요한 기능을 가진 최적화된 객체를 만들고 사용 할 수 있다.
  - 논리적으로 직관적인 프로그램을 짤 수 있다.

색상 정보 저장하기

FUNERS

- 클래스(class) 사용하기
  - 클래스의 정의 – class 클래스이름:
  - 클래스의 생성자 – def __init__(self, 파라미터들):
  - 클래스의 메소드 – def 메소드이름(self, 파라미터들):

색상 정보 저장하기

FUNERS

- 클래스의 정의와 사용

```
class StoreColor: #클래스 정의
    def __init__(self, inside, outside):
        self.inside = inside
        self.outside = outside

    def getColor(self):
        return self.outside*10+self.inside

color = StoreColor(1,2)

print((color.inside, color.outside))
print(color.getColor())
```

색상 정보 저장하기

FUNERS

- 클래스 배열의 저장과 표시

```
from random import randint

class StoreColor: #클래스 정의
    def __init__(self, row, column, inside, outside):
        self.row = row
        self.column = column
        self.inside = inside
        self.outside = outside

    def getColor(self):
        return self.outside*10+self.inside
```

```
def printColorArray(colorArray): #클래스배열의 정보 표시하기
    for row in colorArray:
        string = ""
        for storeColor in row:
            string += "[" + str((storeColor.row, storeColor.column))
            string += ", " + str(storeColor.getColor()) + "]"

        print(string)


colorArray = []

for row in range(0, 5): #무작위 클래스 배열 생성
    colorArray.append([])
    for column in range(0, 5):
        insideColor = randint(1,3)
        outsideColor = randint(1,4)
        colorArray[row].append(StoreColor(row, column, insideColor, outsideColor))

print(colorArray)
printColorArray(colorArray)
```

원하는 정보 찾기

원하는 정보 찾기

FUNERS

- 보통 for를 이용하여 배열을 모든 칸을 확인한다
  - 2차 배열이기 때문에 for문을 2개 중첩해서 사용한다.
- for 안쪽에 if를 넣어 조건에 맞는 정보만 표시한다.
  - 여러 조건이 필요하면 여러 개의 if 와 elif를 활용
- 찾은 정보는 새로운 배열을 만들어 저장한다.

원하는 정보 찾기

FUNERS

- 1의 자리가 2인 칸을 모두 표시하고 저장

```
arr = [[11,13,42,41,31,32],
       [41,33,31,32,22,12],
       [44,12,11,31,23,31],
       [24,32,41,12,32,33,13]]
arr2 = []

for row in range(0, len(arr)):
    for column in range(0, len(arr[row])):
        if arr[row][column]%10 == 1:
            print((row, column)," ",arr[row][c
olumn])
            arr2.append([row, column])

print(arr2)
```

최단 경로 찾기

최단 경로 찾기

FUNERS

- **최단 경로의 정의**
  - 시작은 좌측 최상단, 끝은 우측 최하단
  - 4개의 추천 상점
  - 총 6개의 점을 모두 지나는 최단 경로

최단 경로 찾기

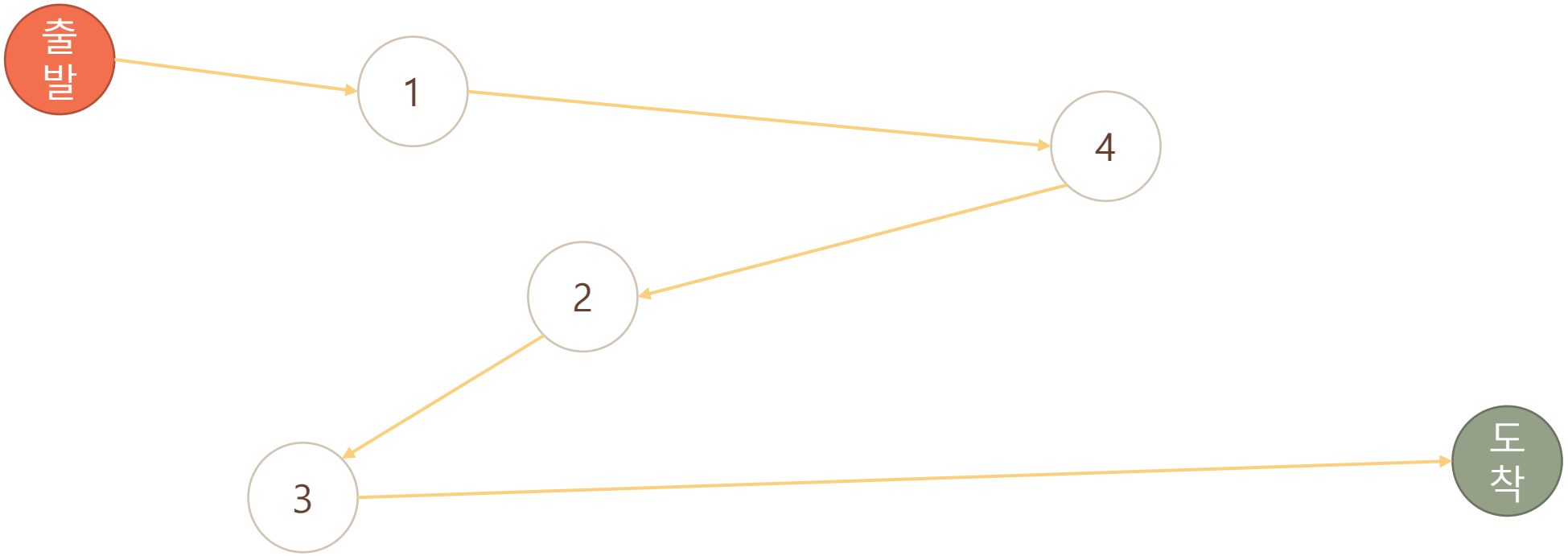
FUNERS

- 시작과 끝이 정해져 있는 최단 경로

최단 경로 찾기

FUNERS

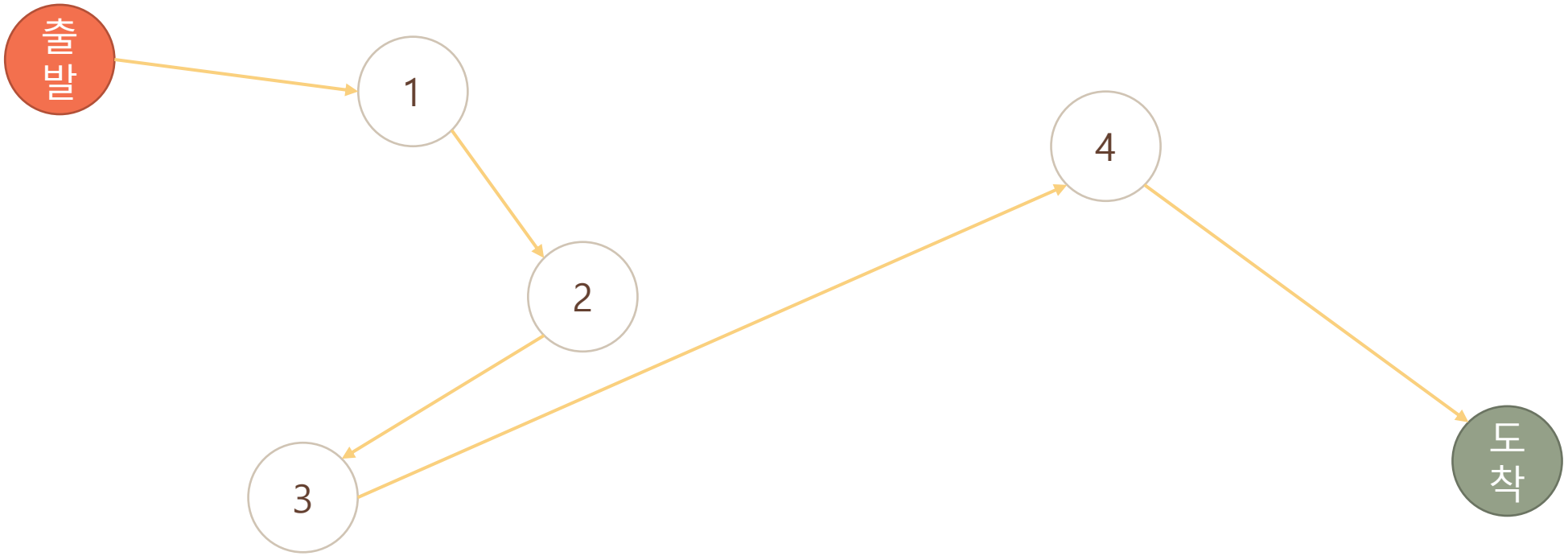
- 제일 위쪽부터 따라가기



최단 경로 찾기

FUNERS

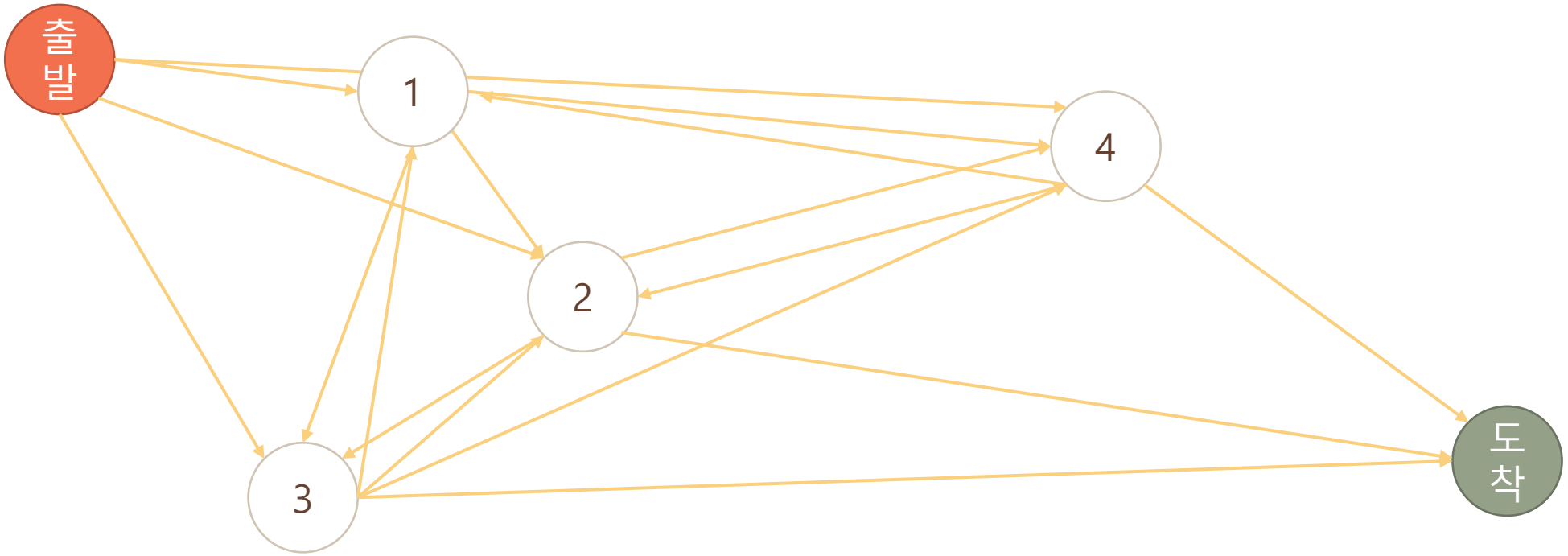
- 제일 가까운 점 따라가기



최단 경로 찾기

FUNERS

- 한번씩 다 해보기



최단 경로 찾기

FUNERS

- **진짜 최단 경로는?**
  - 정확하게 최단 경로를 찾으려면 모든 경우를 다 계산해야 한다.
  - 경우의 수는 지나가야 하는 경로의 수 N의 팩토리얼
    - N*(N-1)*(N-2)*…*3*2*1
    - N이 25일 때 25!은?
  - 다행이도 이번 미션에서는 4개 밖에 안된다.

최단 경로 찾기

FUNERS

- **최단 경로를 찾는 과정**

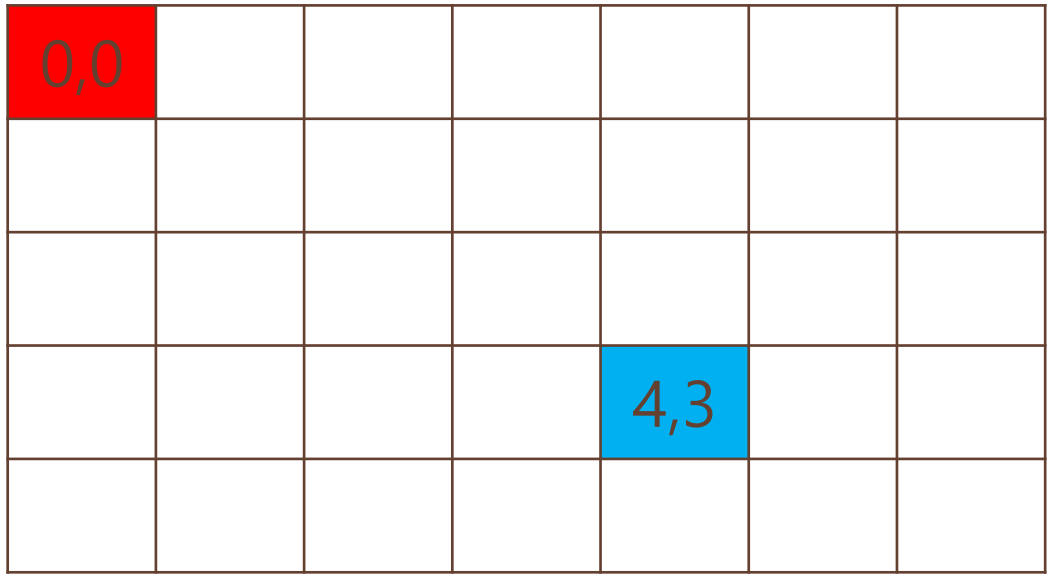
1. 가지고 있는 상점의 정보들 중에 입력 받은 추천 색상과 안쪽 색상이 같은 상점을 모두 찾아낸다.
2. 찾아낸 상점들 중 4개를 선택한다. 이 때 상점의 바깥 색상이 중복 되지 않도록 한다.
3. 선택한 4개의 상점의 순서를 바꿔가며 4! 개의 경로를 계산한다.
4. 2번과 3번을 반복하며 모든 상점 방문 순서를 계산한다.
5. 가장 짧은 경로를 표시한다.

최단 경로 주행

최단 경로 찾기

FUNERS

- 현재 위치에서 다음 위치로 이동하기
  - 현재위치가 (0,0)일 때 (4,3)으로 이동 하려면?

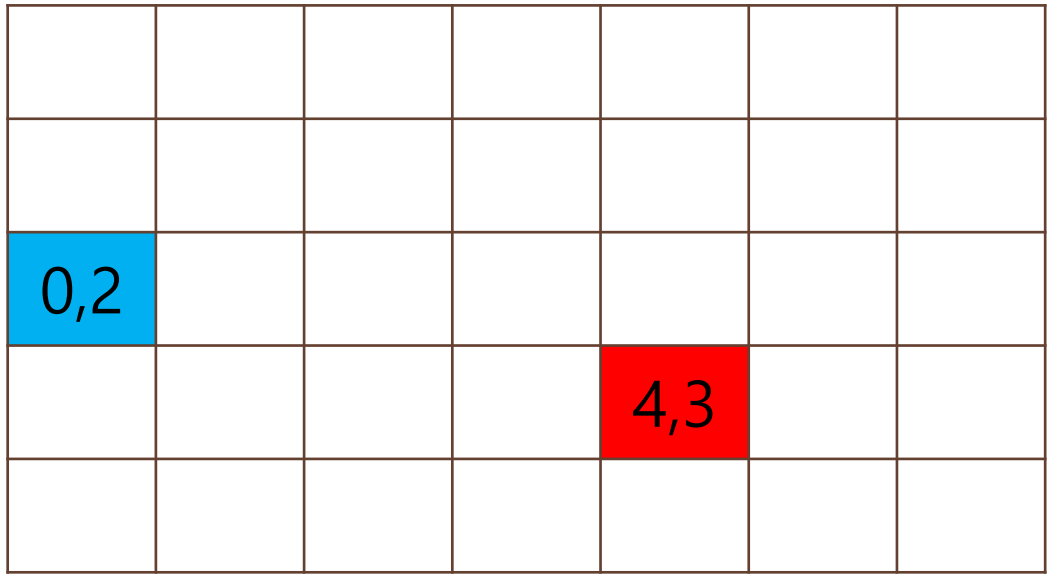


- 같은 위치의 숫자의 차를 이용
- 4 – 0 = 4 : 오른쪽으로 4칸
- 3 – 0 = 3 : 아래로 3칸

최단 경로 찾기

FUNERS

- 현재 위치에서 다음 위치로 이동하기
  - 현재위치가 (4,3)일 때 (0,2)으로 이동 하려면?



- 같은 위치의 숫자의 차를 이용
- 0 – 4 = -4 : 왼쪽으로 4칸
- 2 – 3 = -1 : 위로 1칸

최단 경로 주행

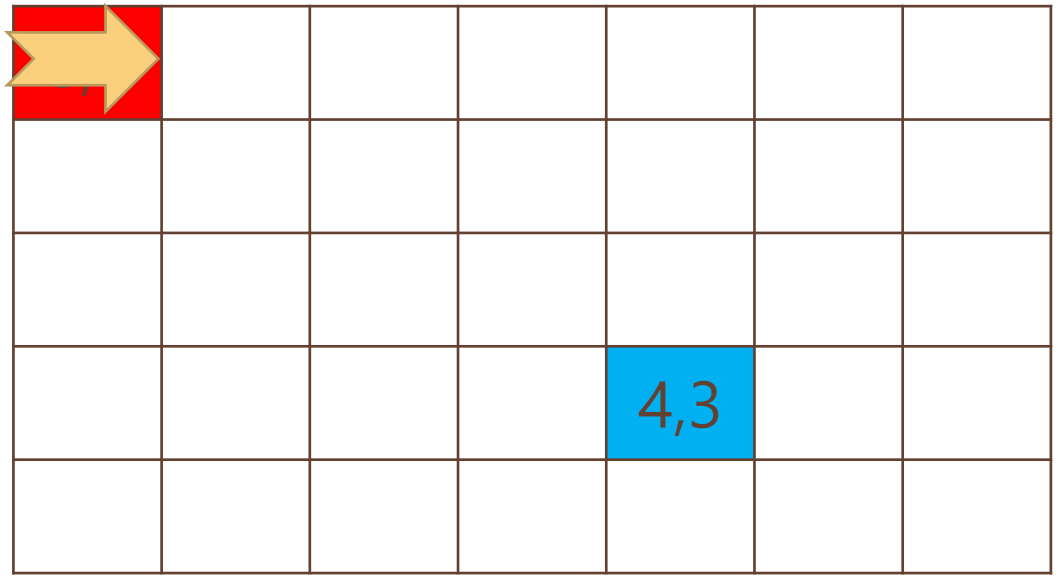
FUNERS

- **로봇의 이동은 언제나 한 칸 단위로 이동하도록 한다.**
  - 한 칸 이동을 함수로 만들어 반복문으로 정해진 횟수 만큼 실행 시키기 위함
- **이전 위치에서의 출발 모습과 다음 위치에서의 도착 모습이 같아야 한다.**
  - 마찬가지로 함수의 반복으로 이동 하기 위함
- **로봇의 이동 후에는 항상 일정한 방향을 바라보도록 한다.**
  - 로봇의 방향에 따라 왼쪽, 오른쪽, 위, 아래의 위치가 바뀌고 그에 따라 우회전, 좌회전을 바꾸어 주어야 하는데, 이것을 막기위한 방책
  - 또는 회전 할 때마다 로봇이 바라보는 방향을 저장 하여 우회전, 좌회전을 구분 할 수도 있다.

최단 경로 찾기

FUNERS

- 현재 위치에서 다음 위치로 이동하기
  - 현재위치가 (0,0)일 때 (4,3)으로 이동 하려면?

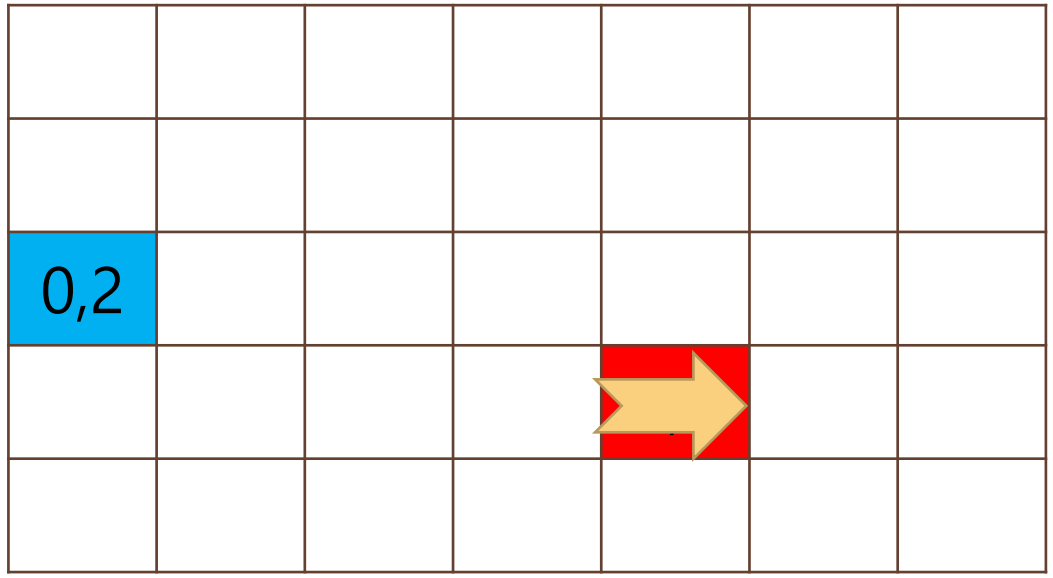


1. 전진
2. 전진
3. 전진
4. 전진
5. 우회전
6. 전진
7. 전진
8. 전진
9. 좌회전

최단 경로 찾기

FUNERS

- 현재 위치에서 다음 위치로 이동하기
  - 현재위치가 (4,3)일 때 (0,2)으로 이동 하려면?



1. 좌회전
2. 전진
3. 좌회전
4. 전진
5. 전진
6. 전진
7. 전진
8. 우회전
9. 우회전

하드웨어 조립의 기초

하드웨어 조립의 기초

FUNERS

- 빔 두개를 평행으로 연결하기1
- 빔 두개를 평행으로 연결하기2
- 빔 두개를 평행으로 연결하기3

하드웨어 조립의 기초

FUNERS

- 빔 두개를 수직으로 연결하기1
- 빔 두개를 수직으로 연결하기2
- 빔 두개를 수직으로 연결하기3

하드웨어 조립의 기초

FUNERS

- 기어를 모터에 연결하기
- 기어를 감속비로 사용하기
- 기어를 가속비로 사용하기

베이스 로봇 제작 팁

베이스 로봇 제작 팁

FUNERS

- 좌우 대칭이 기본
- 무게 중심은 구동 바퀴에 가깝고 되도록 낮은 위치
- 구동 바퀴가 로봇의 진행 방향(앞쪽)에 있는 것이 바람직하다
- 보조 바퀴는 최대한 마찰이 적어지도록
- 조립은 브릭과 모터에 부품을 덧붙이는 식으로
  - 부품으로 모양을 만들고 모터를 붙인다 => X
  - 모터를 놓고 부품을 붙여 나간다.

마무리

FUNERS

- 참고 사이트
  - EV3 Python 학습 사이트 : www.ev3python.com
  - API 참조 : http://python-ev3dev.readthedocs.io/en/latest/spec.html
  - Visual studio code 설치 관련: https://youtu.be/cqtRqsI6xMc
  - EV3와 PC의 연결 : https://youtu.be/TNXqizQTZhs
- 교육, 기술 지원 문의

  퓨너스 (www.funers.com) T.070-8670-8911
- 대회용 경기장 구매

  퓨너스 쇼핑몰(shopfuners.com)