Path finder robot report

202110647 junhyung-kim

202310632 Jeon Jong-hyun

-the contents-

-abstract-

One day, I read an article about the Pathfinder robot. this was developed to guide the way in a place so large as an airport or a train, but this article explains that this robot does not efficiently object to this task. In this process, I feel the doubting the limitations of the Pathfinder robot.

So, I decided on this topic for cross-checking in real life to build the robot and compared the pros and cons of the Pathfinder robot development directions. This report is for research and study about the shortest problem path-find algorithm.

In this Project I will use 'NXT Mindstorm', because I need a line tracer robot, to search the root.

I have already this material.

-What is the Dijkstra algorithm?-

Dijkstra algorithm

Simply put, the Dijkstra algorithm is a way to find the shortest path from one starting point to all other vertices. It is an algorithm (shortest path problem) that finds the shortest distance from one vertex to all vertices in the graph, respectively. It is an algorithm devised by Etzher Dijkstra. The practical use of this algorithm is applied to most problems of finding the fastest path to an answer at the lowest possible cost. Therefore, no further detailed explanation of how many practical examples are available is given. For example, if the real world is regarded as a graph in which cities are nodes and roads are edges, this algorithm can find the fastest way to connect two cities, such as navigation. It can also be used to find the minimum transportation cost point in maze-search algorithms, OSPF-type protocols in routing, and industrial location theory. A slightly difficult example is to make a program for solving a Rubik's cube Dijkstra algorithm. The Dijkstra algorithm is as follows. (P[A][B] is assumed to be the distance between A and B)

Create an array d[v] to store the shortest distance from the starting point, populate 0 for the starting node and a huge value INF for the other nodes except the starting point. (It means a value that can be considered infinite, not truly infinite.) Usually, the INF is set to fit the theoretical maximum value of the shortest-distance storage array. In practice, it is usually set to the maximum value for the element type of 'd.'

The starting node number is stored in variable A representing the current node.

For any node B that can go from A, compare the values of d[A] + P[A][B] and d[B]. Compared to INF, the electron is unconditionally small.

If the value of d[A] + P[A][B] is smaller, i.e., if it is a shorter path, update the value of d[B] to this value.

Do this for all neighboring nodes B of A.

Change A's status to "Visit Completed". Then A is no longer used.

Among all the nodes in the "unvisited" state, one node with the shortest distance from the starting point is selected and the node is stored in A.

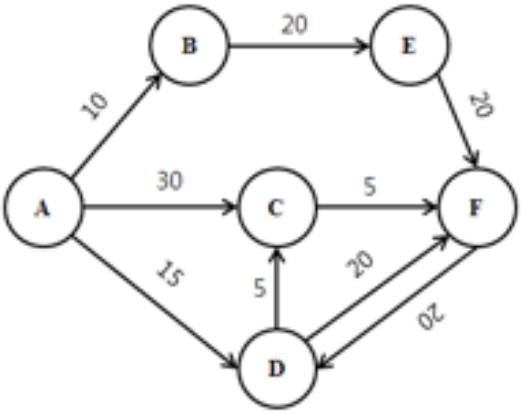
Repeat the processes 3 to 7 until the arriving node is in the "visited complete" state, or until it is no longer possible to select a node in the unvisited state.

```
function Dijkstra(Graph, Source):

    dist[source] ← 0                          //
소스와 소스 사이의 거리 초기화 --출발지와 출발지의
거리는 당연히 0--
    prev[source] ← undefined                  //
출발지 이전의 최적 경로 추적은 없으므로 Undefined

    create vertex set Q                       //
노드들의 집합 Q(방문되지 않은 노드들의 집합) 생성
시작

    for each vertex v in Graph:               //
Graph 안에 있는 모든 노드들의 초기화
        if v ≠ source:                        // V
노드가 출발지가 아닐 경우(출발지를 제외한 모든
노드!)
            dist[v] ← INFINITY                //
소스와 V노드 사이에 알려지지 않은 거리 --얼마나
```

```
먼지 모르니까-- = 무한값 (모든 노드들을 초기화하는
값)
           prev[v] ← UNDEFINED           //
V노드의  최적경로 추적 초기화
        add v to Q                       //
Graph에 존재하고 방금 전 초기화된 V 노드를
Q(방문되지 않은 노드들의 집합)에 추가

//요약하자면, Graph에 존재하는 모든 노드들을
초기화한 뒤, Q에 추가함.

    while Q is not empty:                // Q
집합이 공집합 아닐 경우, 루프 안으로!
        u ← vertex in Q with min dist[u]   //
첫번째 반복에서는 dist[source]=0이 선택됨. 즉,
u=source에서 시작
        remove u from Q                  // U
노드를 방문한 것이므로 Q집합에서 제거

        for each neighbor v of u:        //
U의 이웃노드들과의 거리 측정
           alt ← dist[u] + length(u, v)   //
출발지 노드 부터 계산된 U노드까지의 거리 + V에서
U의 이웃노드까지의 거리
                                         // =
source to U + V to U = source to U
           if alt < dist[v]:             //
방금 V노드까지 계산한 거리(alt)가 이전에
V노드까지 계산된 거리(dist[v])보다 빠른 또는
가까운 경우
               dist[v] ← alt             //
V에 기록된 소스부터 V까지의 최단거리를 방금
V노드까지 계산한 거리로 바꿈
               prev[v] ← u               //
제일 가까운 노드는 지금 방문하고 있는 노드(U)로
바꿈

    return dist[], prev[]                //
계산된 거리값과 목적지를 리턴
```

i

Assume that a network exists as follows. First, router A's goal is to calculate the shortest distance to F, and the Dijkstra algorithm is used as a means. In this case, the meaning of each data is as follows.



S = {}

d[A] = 0
d[B] = infinity
d[C] = infinity
d[D] = infinity
d[E] = infinity
d[F] = infinity

Q = {A, B, C, D, E, F}

1. Without having to check the distance between the starting point and the starting point, of course, it has a value of 0. Because the starting point is set to A, d[A]=0. (Node A has not been visited yet.)

2. All nodes except the origin have not yet been identified, so d[other nodes]=.

3. Q is a set of unvisited nodes, so at initialization it becomes a set of all nodes.

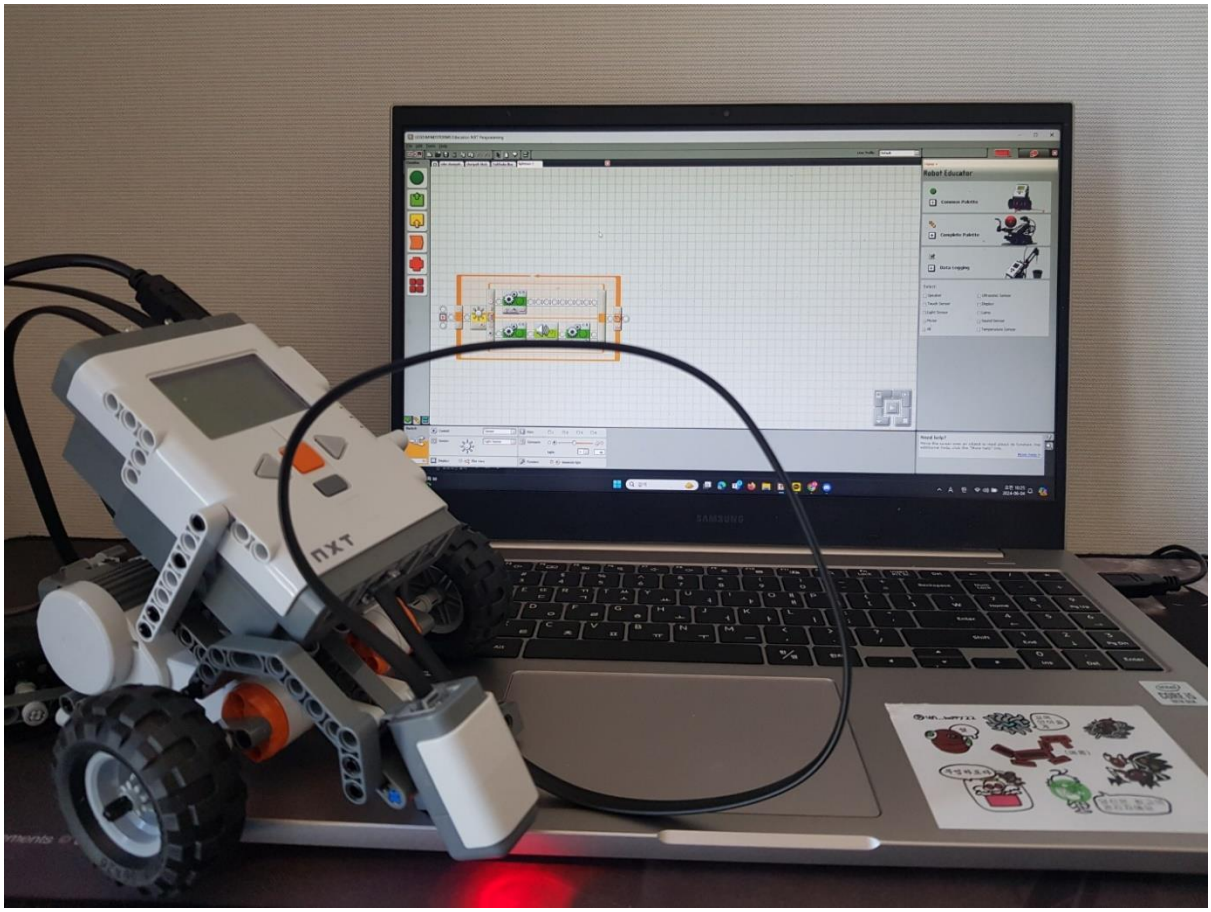
4. S is in an empty state because no nodes have been visited yet.

-What is the 'NXT'-

'The Lego Mind storm NXT model is a kit made by 'MIT' and 'Lego' for research on algorithms and robotics kits [ii]

These kits consist of the motor blocks, a Lego frame block as the robot's body, sensors(each of the light sensor, radar sensor, touch sensor, sound sensor) a G-block(main board of the NXT model),

However, NXT 2.0 is a very old model, so I had a problem with the find 'robots' version, so I decided to use 'NXT-G' which is the block programming language.



-how-to programming for Dijkstra algorithm-

-First try: Make the graph and specify the shortest way-

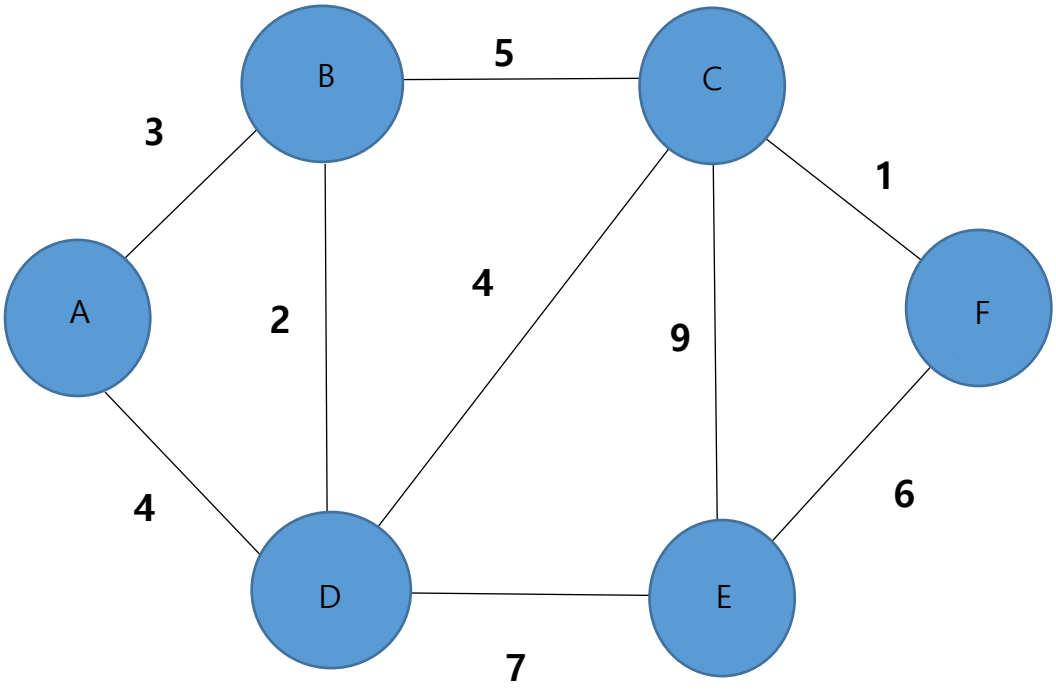
For example, make the graph for the node of the A, B, C, D, E, F, and specify some nodes for the shortest way to arrival point.

Make all of the nodes give the value (weight) to each node and, make the expression for the shortest path, And operator does find the minimum value to the sum of the edges.

For example, give the A node to 3, give the B node to 7, give the C node to 4 and the D node to 5, and then specify the start node to 'A' and the arrival node to 'D' and then make the expression 'A + B + D' and 'A + C + D' for the compare two sums of the edges.

Compare the minimum size and choose the shortest way.

This is the first Dijkstra algorithm map for this project



1. A-> B-> C-> F (3 + 5+ 1)
2. A -> B-> D-> C ->F (3 + 2 + 4+ 1)
3. A->B->C->E->F (3 + 5 + 9 + 6)
4. A->B->C->D->E->F (3 + 5 + 4 + 7 + 6)
5. A->B->D->C->E->F (3 + 2 + 7 + 9 + 6)
6. A-> B -> D-> E -> C -> F (3 + 2 + 7 + 9 + 1)
7. A-> D -> B -> C -> E -> F (4 + 2 + 5 + 9 + 1)
8. A -> D -> E ->C -> F (4 + 7 + 9 + 1)
9. A-> D-> C -> E -> F (4 + 4 + 9 + 6)

-Explain the code block- [iii]

-variable block-

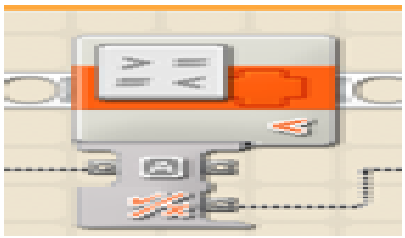
The variable block consists of Numbers, text, and Logic, in this code, I used this logic block to declare the node weights as the Input parameter.

-Move block-

This is a block for moving the motor working with user input data.

This block can have the condition, could be logical, or interface to the sensor blocks

-Compare-



This block compares the value of the logic blocks or variable blocks and operator block, this can connect with the other logic block or variable.

-operator block-

This is done operator as the can calculate the expression. In this code, I'm using this for the sorting node weight.

-Loop function block-

This block do task as like loop statements (while loop, for loop)

This is repeat your code as for the when you trying to one more than actions.

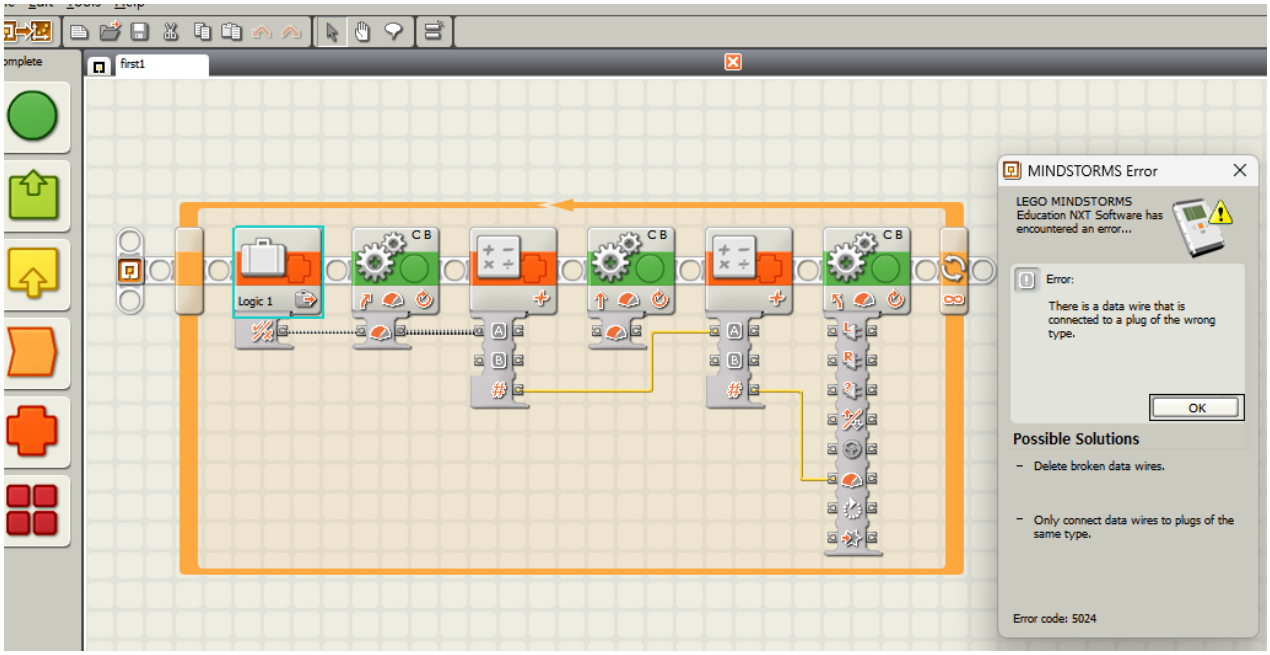
-First algorithm: sorting the data-

Firstly, I try the logic connection to when the operator blocks the task to an expression

-Switch function block-

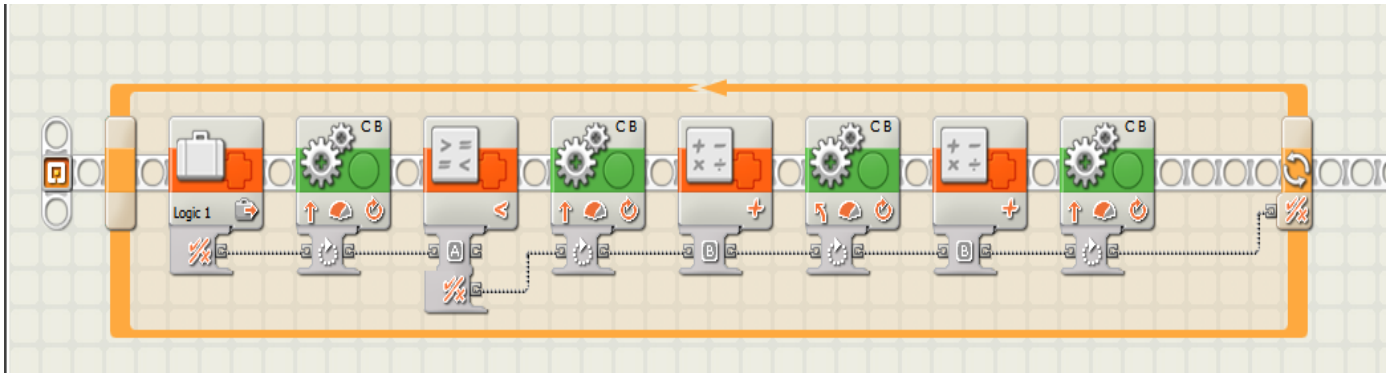
This is block change the task of the user code.

As like a Switch statements.



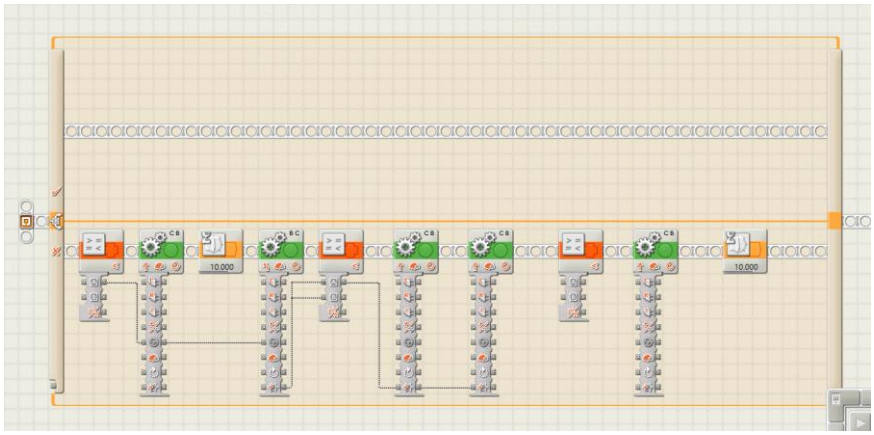
However, this has an error, so I found the other code structure.

The program alert to my input data wire connected with the wrong connection, so I re-connected the code logic



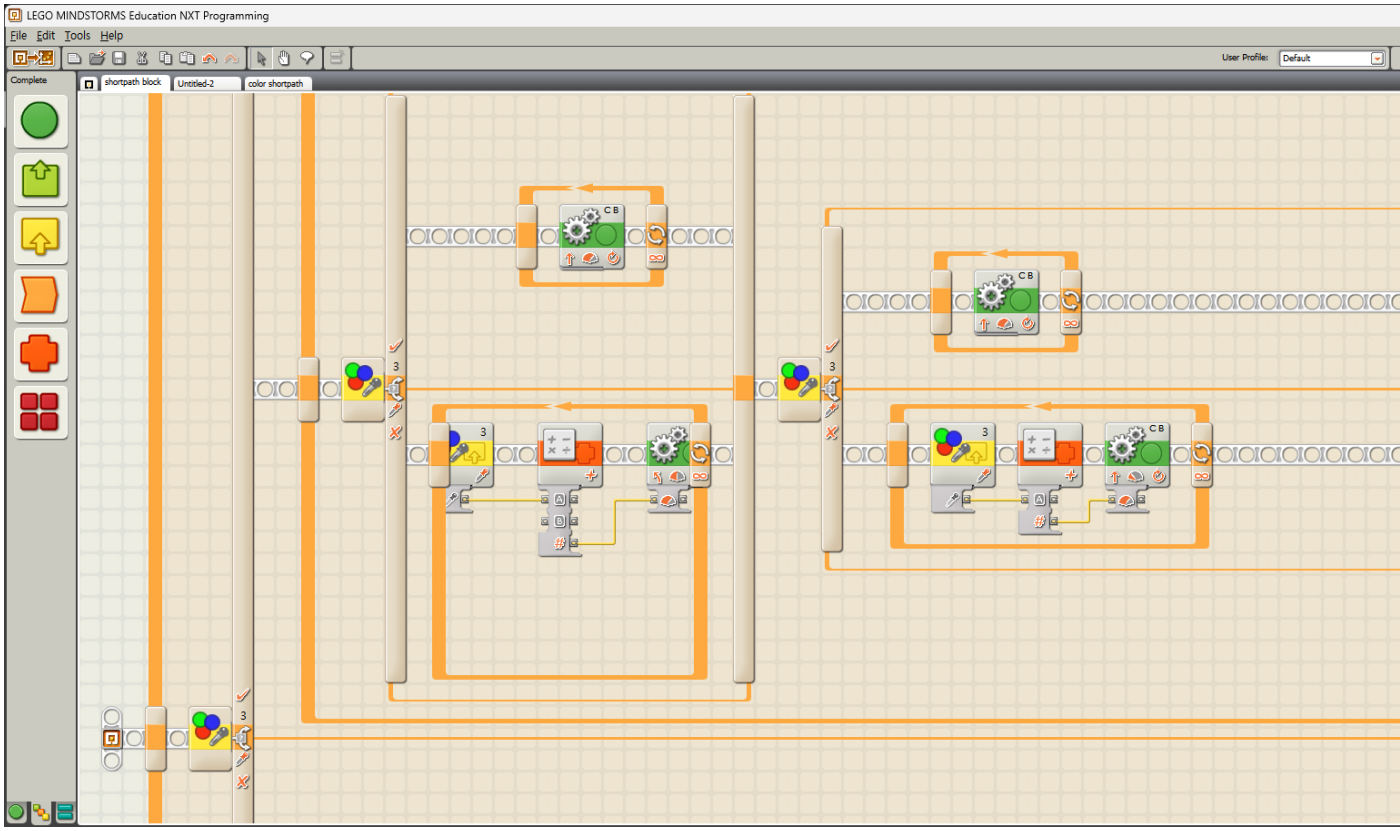
Secondly, I'm trying to sort each of the node's weights (3 + 5 + 1) using the operator blocks to make the expressions, however, this can't store the data, and the prompt alert that this is a program has a logical issue and the source of the data is not founded in code.

Thirdly, I tried to set the timer as the program as do the task in the time



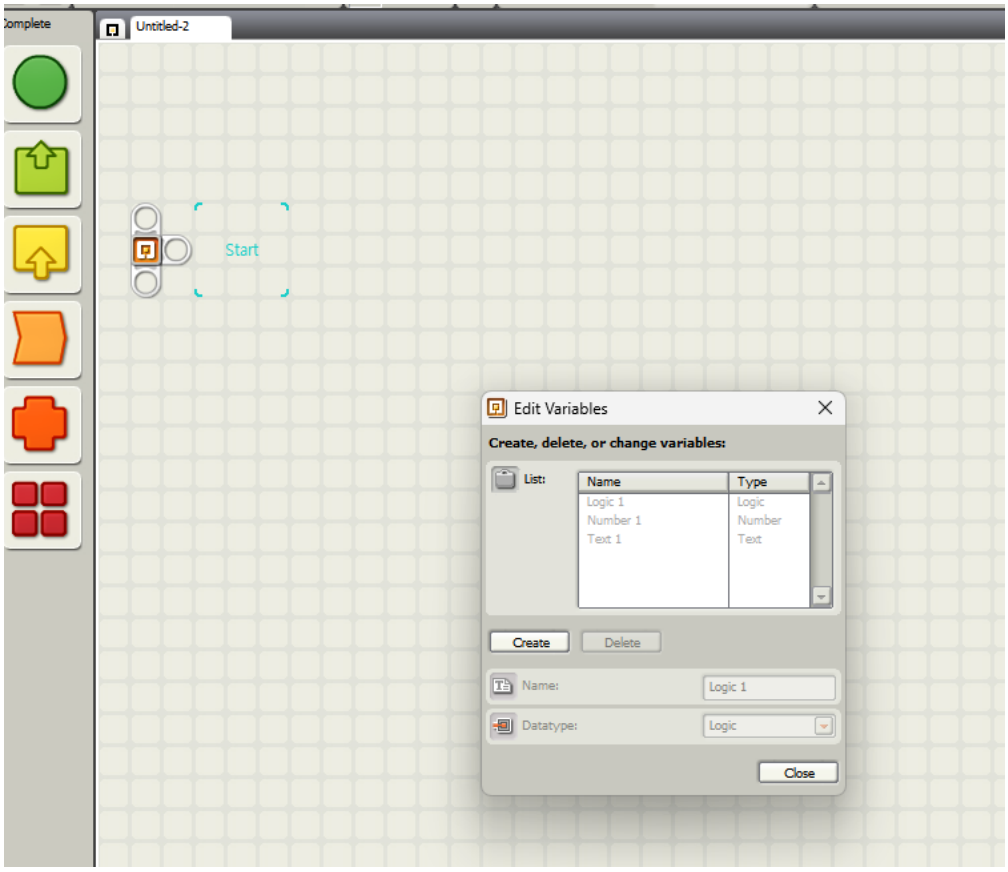
But it also, the logic of the data source is not founded.

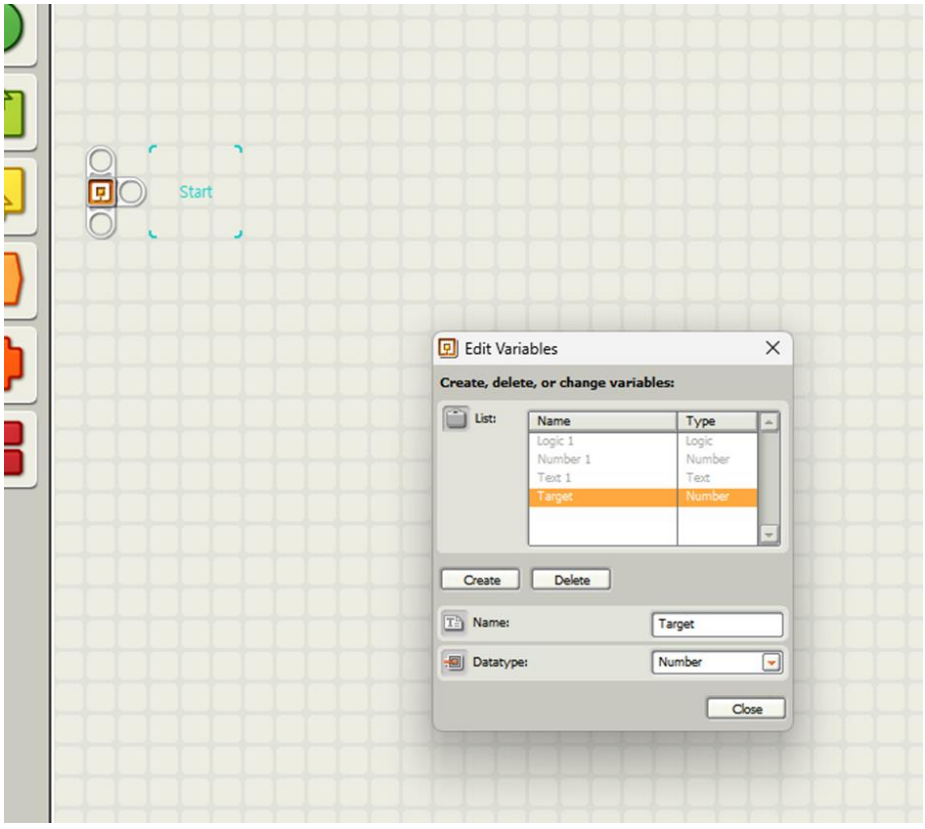
-Solve the problem try: Using Sensor blocks for data source-

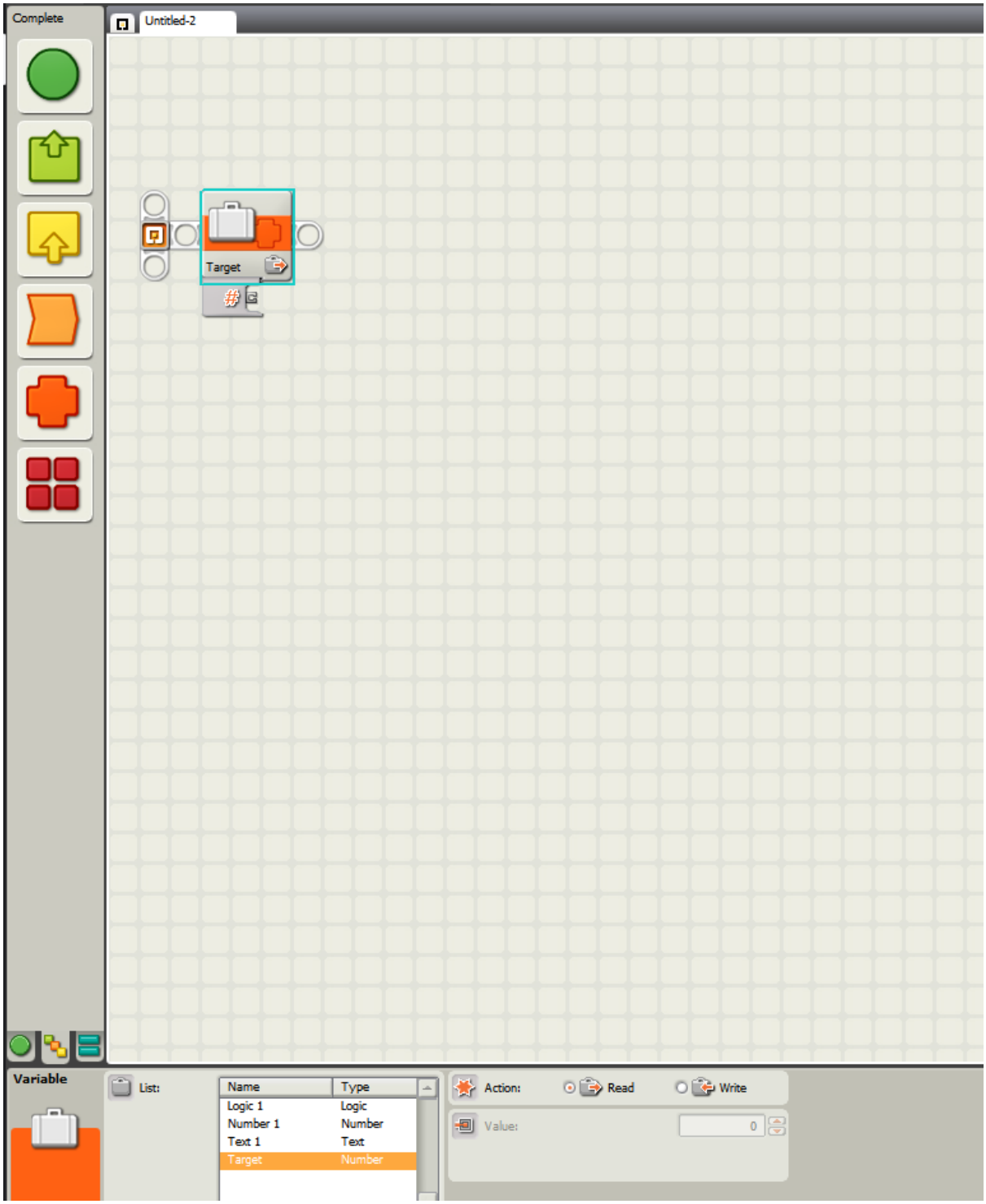


This is the first try using sensor blocks, so in this process, I was expected the make the node different each color as the node for example, 'a to red color, and node 'b' to blue color, and the task to light sensor detected the color and categorized (compare) to each value as switch function. However, my robot sensor can't detect the color.

So I choose the make line detecting robot for the follow-to-edges.



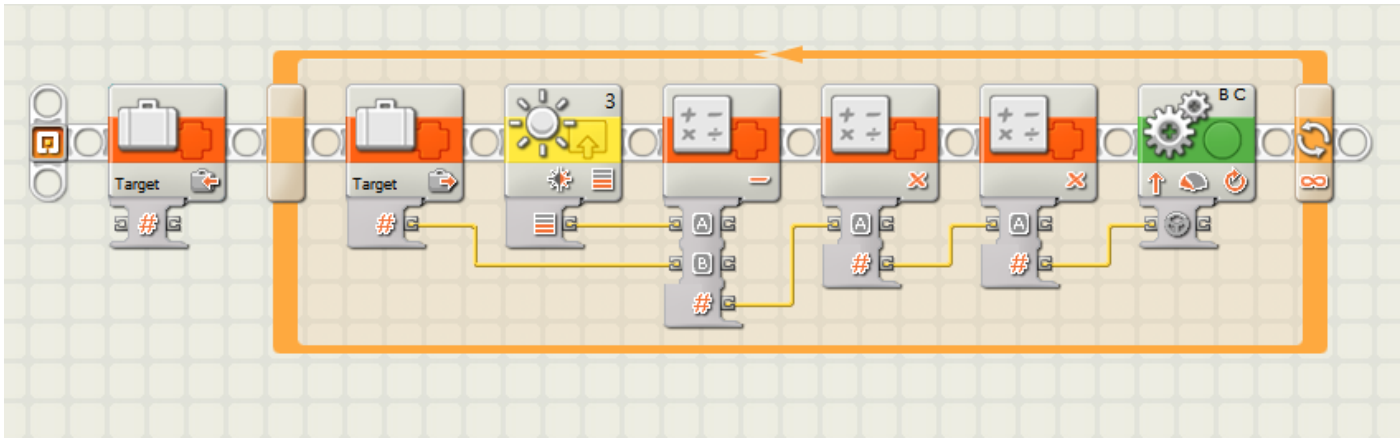


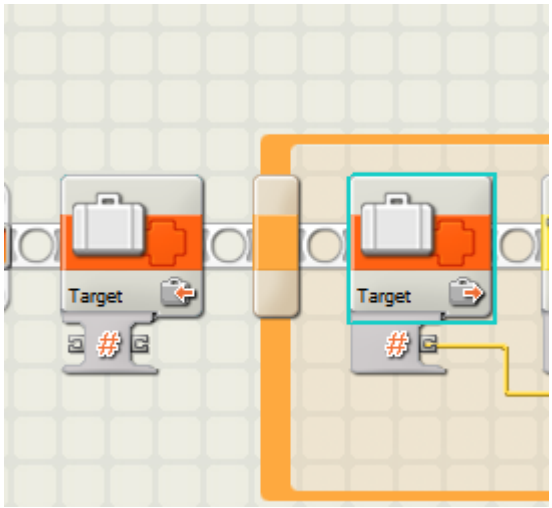


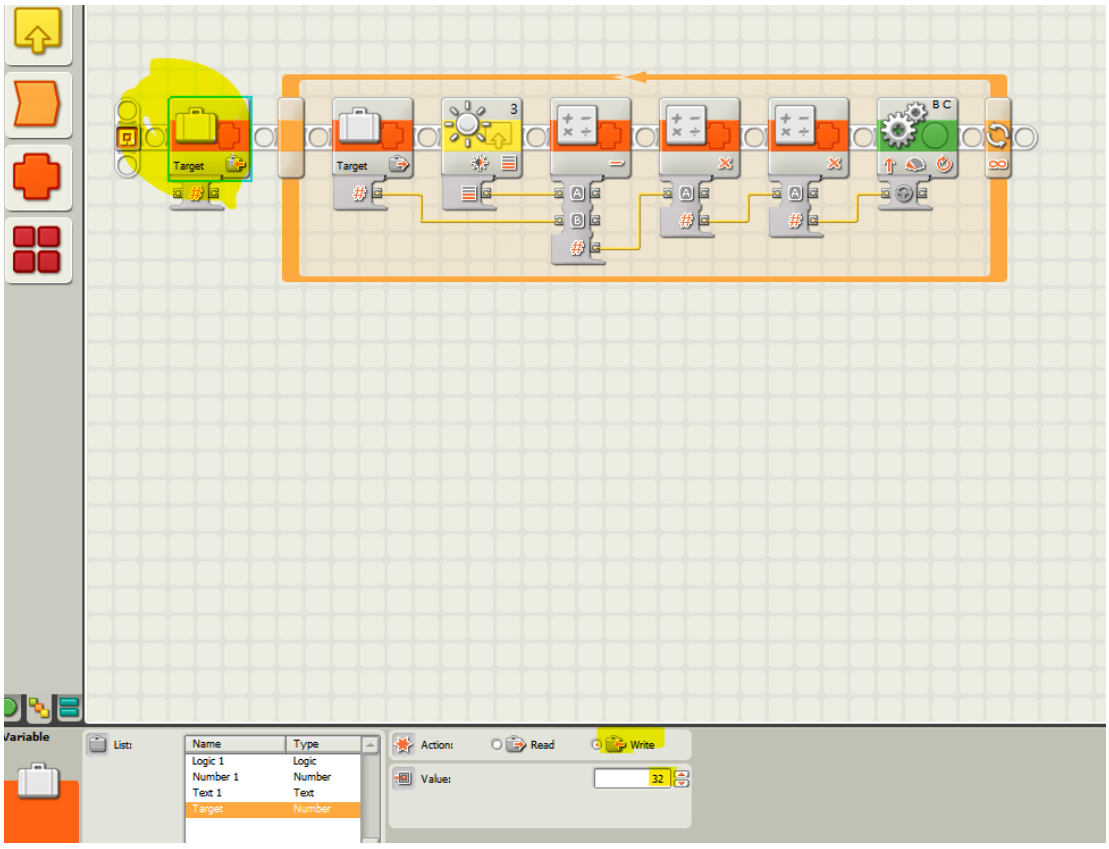
Firstly, set the variable block name as the 'Target' in this code, this is target to the 'line'

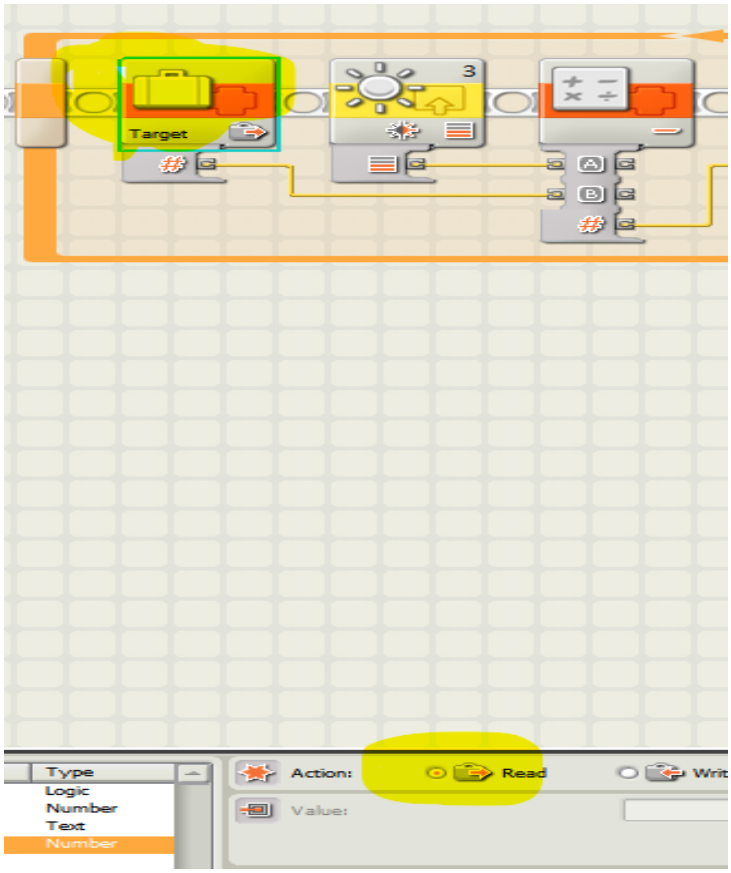
When I use sensor blocks. After writing entire the code, this target variable action will be 'Write' for the light sensor can be categorized as the line between the Floor.

After the 'Target'-Read variable directly connects with the light sensor block, this 'Target'


Target
Target
3
B C


Target
Target


Target
Target
3
B C
Variable
List:
Name
Type
Logic 1
Logic
Number 1
Number
Text 1
Text
Target
Number
Action:
Read
Write
Value:
32



-Pros and cons of NXT-G VS ROBOTC-

Pros of the NXT-G

This is friendly for the developer and user to understand, and the biggest advantage of the Block programming language is that, it has a simple and easy structure for understanding algorithms and engineering for mature or other major students this is a code example like 'scratch'

Cons of the NXT-G

This is its one-driver program. So, this technical logic has lower skills than 'RobotC' programming. NXT-G skills depend on the user experience and major skills, however, it still uses many of the code blocks and after the Discontinued of the 'Mindstorm Sereies' this version no longer gets technical support.

Pros of the 'RobotC'

ROBOTC is a cross-robotics-platform programming language for popular educational robotics systems, this programming language is used to study more variety of algorithms,[iv] and most of the

'STEAM' students using 'RobotC' for their project. The reason is 'RobotC' model is more technical than 'Nxt-G' and creative.

Cons of the 'RobotC'

I as mentioned, the 'RobotC' is need more technical skills, and need the major skills, so 'RobotC' is hard to understanding algorithm structure for A person without a major.

-Conclusion-

-Summary of the project-

Actually, my development Direction is focused on the path finder robot based on sorting the data and comparing the lower ones, after the Operator(CPU) finds the short path

The reason why I chose the design for two reasons, the first reason is the make a unique project, and to avoid the embezzlement issue, many students and engineers already use a sensor' block for tasks then the robot can find the node of direction. So, I chose the sorting data as the 'Logic blocks' but this needs more 'Sensor blocks' The second reason is that Building the robot is easy to understand for users. If using the 'RobotC' the code is more 'Succint' and 'Technical' however, 'RobotC' needs the setup works and follow-up work, this is means, the user or developer requires more tasks.

'RobotC' is the cross-robotics-platform programming language. This is using a code line, not a block. So, this needs more skills than 'Nxt-G' as block diagram languages.

-Future plan-

After the project, I continued to research this project, In the current situation, before comparing the two algorithms targeted in advance, the top priority is the growth of NXT-G's algorithm writing skills, and resetting the target fee to actually implement the model.

Reference

[i] A study on the extension of Dijkstra algorithm, kuwangwoon university post.doc junhsun_sin, page[8],[9],[10],[11],[14], 200000189303_20240604143239.pdf (dcollection.net)

[ii] Whst is the nxt?, What is the Nxt-G ? , Lego Mindstorm NXT, james folyd kelly, Apress, page[xviii],[1],[8],https://books.google.co.kr/books?hl=ko&lr=&id=IAU4QHrP1WsC&oi=fnd&pg=PR1&dq=info:jO1h

A5qyEsJ:scholar.google.com/&ots=UoldJL9YkC&sig=oST6KlhkljmPRfBCZggVRHN3mu0#v=onepage&q&f=false

[iii] Explain the code block, first algorithm,

https://www.nxtprograms.com/help/MyBlocks/tutorial.html

[iv] Advanced NXT The Da Vinci Inventions Book, Author: Matthias Paul Scholz · 2007, page[1],[26],[27],[43]