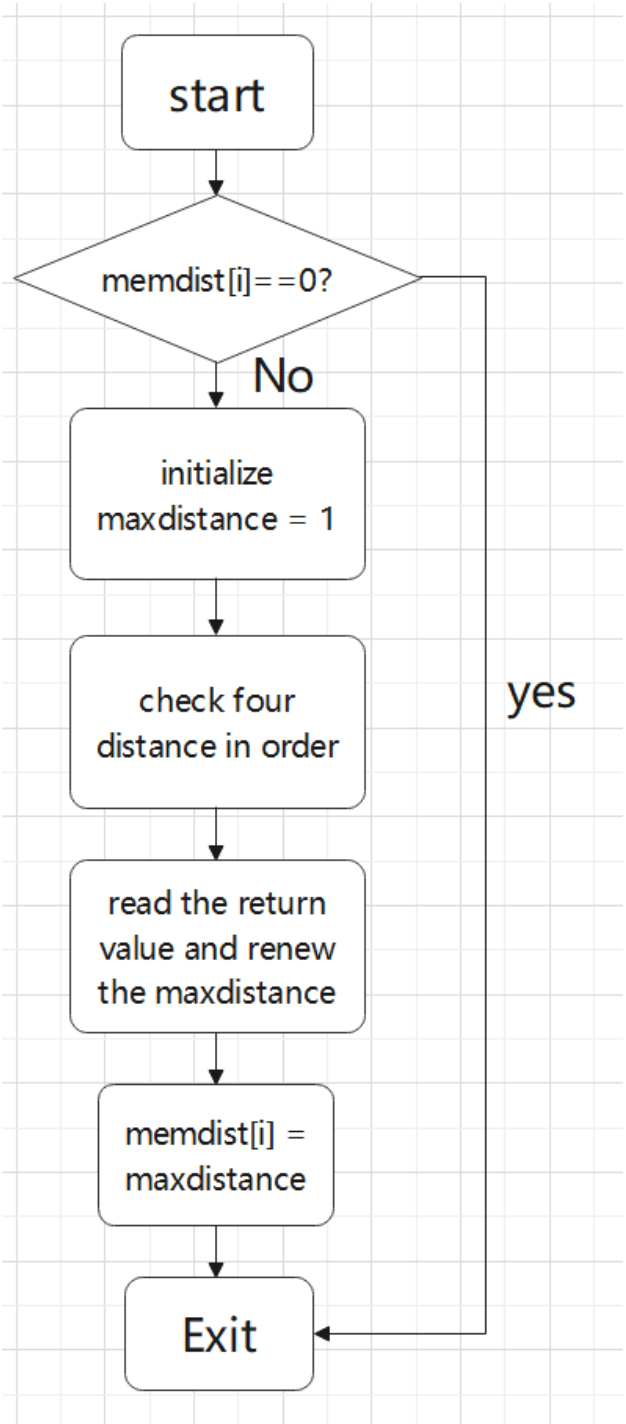
PartA. Algorithm

I use memorization search and depth first search.

①Initialize an array to record the max distance from this point

②start loop, check each point's max distance

③for each point, go through this procedure



Read the return value and renew the max distance

PartB. Essential parts of my code

1.

```
;check if memdist[i] == 0
LEA         R0,MemDist    ;check the memdist, if it is not 0, then return it
ADD          R0,R0,R1
LDR          R0,R0,#0
BRnz        CHECK_NORTH
STR          R0,R5,#0
BRnzp       EXIT_CD


;store result in memdist array
LEA         R0,MemDist    ;store the result
ADD          R0,R0,R1
LDR          R2,R5,#0
STR          R2,R0,#0
STR          R2,R5,#3
```

These two parts are essential to memorization, which can improve efficiency of my program.

2.

```
ADD          R6,R6,#-3     ;push arguments onto the stack
STR          R1,R6,#2
STR          R4,R6,#1
STR          R3,R6,#0
JSR         CCL_Dist

LDR          R0,R6,#0       ;R0 <- return value (result of DFS)
ADD          R0,R0,#1
ADD          R6,R6,#4

LDR          R3,R5,#4
LDR          R4,R5,#5
LDR          R1,R5,#6
```

This part set the argument and call the subroutine, then read the return value from the stack.

PartC. The questions that TA asked you, and answers.

1. Describe the procedures of your program:

(See partA) I use memorization search and depth first search. For each point, I go the same direction until it can't go further. I use memdist array to record the max distance to this point. So next time I walk here, I can simply read the result from the array without doing dfs again.