ANSWERS

F5.5 Linear search

Algorithms · GCSE · OCR J277 2.1.3, AQA 8525 3.1.3, Edexcel 1CP2 1.2.6 · about 15 min

What this lesson is about

Checking every item: finding a marker in the robot's sightings.

- The algorithm
- Counting the work
- Searching what the robot sees
- When linear search is the right choice

Questions

5 marks in all

1. Linear search looks for 22 in [12, 5, 31, 8, 22, 3]. How many items does it check? [1 mark]

> **Answer:** 5. It checks 12, 5, 31, 8 and then finds 22.

2. Linear search looks for 99 in a list of 8 items that does not contain it. How many items does it check? [1 mark]

> **Answer:** 8. Every item must be checked to be sure it is not there.

3. Does linear search need the list to be sorted? [1 mark]

- ☐ **A** No, it works on any list
- ☐ **B** Yes, always
- ☐ **C** Only for numbers
- ☐ **D** Only for long lists

> **Answer:** A. It checks every item in turn, so the order does not matter.

4. What is the main disadvantage of linear search? [1 mark]

- ☐ **A** It is slow on large lists
- ☐ **B** It needs sorted data
- ☐ **C** It cannot find the first item
- ☐ **D** It uses a lot of memory

> **Answer:** A. In the worst case it checks every item, so a million items can mean a million checks.

5. What does this program print? [1 mark]

```
def find(items, target):
    for i in range(len(items)):
        if items[i] == target:
            return i
    return -1

print(find([4, 9, 2], 2), find([4, 9, 2], 7))
```

Answer:

```
2 -1
```

2 is at index 2; 7 is not found, so -1.

The task: find the marker

The task asks `Which marker?` and answers `31`. Look in all eight directions and record the marker straight ahead in each, then use **linear search** on your list to find where the wanted marker is. Print `found <id> at <degrees> degrees after <n> checks`, turn to face it, and beep.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

set_cv("apriltag")
target = int(input("Which marker? "))
```

The hint students can ask for: Record the id straight ahead (cx near 160) at each of 8 turns. Then loop through the list counting checks until you find the target; its index times 45 is the direction.

A solution

```
from bugbot import *
connect()
set_cv("apriltag")
target = int(input("Which marker? "))

def ahead_id():
    for tag in marker_tags():
        if abs(tag[1] - 160) < 30:
            return tag[0]
    return None

sightings = []
for i in range(8):
    sightings.append(ahead_id())
    turn_right(30, angle=45)

checks = 0
where = -1
for i in range(len(sightings)):
    checks = checks + 1
    if sightings[i] == target:
        where = i
        break
print("found", target, "at", where * 45, "degrees after", checks, "checks")
turn_right(30, angle=where * 45)
tone(880, 0.3)
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.