

Filtering: Dynamic Conditions

A **dynamic filtering condition** changes while an algorithm processes the data. Instead of comparing each item with a fixed value, we compare it with a variable whose value may change during the iteration—for example, keeping track of the highest mark seen so far.

1. Static vs. dynamic filtering

A **static condition** compares a data field with a fixed value:

```
X.gender == "F"
```

The condition stays the same for every student.

A **dynamic condition** compares a data field with a changing variable:

```
X.marks > max
```

Here, max may change when a higher mark is found. As max changes, the condition changes too.

2. Finding the maximum

Suppose we want the highest value of a field, such as students' Maths marks.

Use a variable called max to hold the largest mark found so far.

Steps

1. Set max to a value lower than any possible mark. For marks from 0 to 100, use 0.
2. Examine each student's card.
3. Compare the student's mark with max.
4. If the mark is greater, replace max with that mark.
5. When all cards have been examined, max contains the highest mark.

Pseudocode

```
max = 0
```

```
For each student X:
```

```
    If X.mathsMarks > max:
```

```
        max = X.mathsMarks
```

Example

Marks are:

65, 82, 78, 91

The value of max changes as follows:

Mark being checked	Current max	Update?	New max
--------------------	-------------	---------	---------

65	0	Yes	65
82	65	Yes	82
78	82	No	82
91	82	Yes	91

The maximum is **91**.

3. Finding which items have the maximum

Finding the maximum number does not automatically tell us which students earned it.

We can store the IDs of the students who have the highest value in a list called `maxIds`.

- If a new value is **greater** than `max`, update `max` and replace the list with that student's ID.
- If a value is **equal** to `max`, add that student's ID to the list.
- If a value is **less** than `max`, do nothing.

Pseudocode

```
max = 0
```

```
maxIds = []
```

```
For each student X:
```

```
  If X.marks > max:
```

```
    max = X.marks
```

```
    maxIds = [X.uniqueId]
```

```
  Else if X.marks == max:
```

```
    Add X.uniqueId to maxIds
```

This handles ties correctly. For example, if two students both have 91, the list contains both of their IDs.

Dynamic Filtering Condition

A dynamic filtering condition changes while an algorithm processes the data. We compare each item with a variable whose value may change during the iteration.

The condition changes as we go!

1. Static vs Dynamic Filtering

Static Condition

- Compares a data field with a fixed value.
- The condition stays the same for every item.

`X.gender == "F"`

We check the same condition for every student.

Dynamic Condition

- Compares a data field with a changing variable.
- The variable's value may change during the iteration.

`X.marks > max`

As `max` changes, the condition also changes for the next item.

2. Finding the Maximum

Steps

- 1 Set `max` to a value lower than any possible mark. (For marks 0-100, use 0)
- 2 Examine each student's card.
- 3 Compare the student's mark with `max`.
- 4 If the mark is greater, replace `max` with that mark.
- 5 When all cards have been examined, `max` contains the highest mark.

Pseudocode

```
max = 0
For each student X:
  If X.mathsMarks > max:
    max = X.mathsMarks
```

Maths Marks = 78

We check each student one by one.

3. Example: Finding the Maximum

Marks are:
65, 82, 78, 91

Check 65
max becomes 65

Check 82
max becomes 82

Check 78
max stays 82

Check 91
max becomes 91

Step-by-step values of max			
Mark being checked	Current max	Update?	New max
65	0	Yes	65
82	65	Yes	82
78	82	No	82
91	82	Yes	91

The maximum is **91**.

4. Finding which items have the Maximum

Finding the maximum number does not automatically tell us which students earned it. We store the IDs of the students who have the highest value in a list called `maxIds`.

Logic

- If a new value is greater than `max`, update `max` and replace the list with that student's ID.
- If a value is equal to `max`, add that student's ID to the list.
- If a value is less than `max`, do nothing.

Pseudocode

```
max = 0
maxIds = []
For each student X:
  If X.marks > max:
    max = X.marks
    maxIds = [X.uniqueId]
  Else if X.marks == max:
    Add X.uniqueId to maxIds
```

Example (with ties)

Student ID	Marks	max	maxIds
S1	65	65	[S1]
S2	82	82	[S2]
S3	78	82	[S2]
S4	91	91	[S4]
S5	91	91	[S4, S5]

Two students (S4 and S5) have the maximum mark 91, so both IDs are stored.

4. Finding the minimum

To find a minimum, use a variable called min.

1. Set min to a value at least as large as any possible value.
2. Examine each item.
3. If its value is less than min, update min.
4. At the end, min contains the smallest value.

For subject marks with a range of 0-100, start with 100.

Pseudocode

```
min = 100
```

```
For each student X:
```

```
    If X.mathsMarks < min:
```

```
        min = X.mathsMarks
```

To collect all students with the minimum, use a list in the same way as for the maximum: replace the list when a smaller value is found, and add to it when an equal value is found.

5. Example: Longest sentence

To find the sentence containing the most words, keep two variables:

- count: number of words in the current sentence.
- max: largest sentence length found so far.

For each word:

1. Increase count.
2. If the word ends with a full stop, the sentence is complete.
3. Compare count with max.
4. If count is larger, update max.
5. Reset count to 0 for the next sentence.

Pseudocode

```
count = 0
```

```
max = 0
```

```
For each word X, in paragraph order:
```

```
    count = count + 1
```

```
    If X.word ends with ".":
```

```
        If count > max:
```

```
            max = count
```

```
        count = 0
```

The dynamic comparison is `count > max`: the value of max may change after each sentence.

Dynamic Filtering - Finding the Minimum & Longest Sentence

In dynamic filtering, the condition changes as we process each item.

4. Finding the Minimum

Idea
To find the smallest value, use a variable min and keep updating it while examining each item.

Steps

- Set min to a value at least as large as any possible value. (For marks 0–100, use 100).
- Examine each item.
- If its value is less than min, update min.
- At the end, min contains the smallest value.

Pseudocode

```
min = 100
For each student X:
  If X.mathsMarks < min:
    min = X.mathsMarks
```

Example
Marks are: 65, 82, 78, 91, 52

Mark being checked	Current min	Update?	New min
65	100	Yes	65
82	65	No	65
78	65	No	65
91	65	No	65
52	65	Yes	52

The minimum mark is 52.

Flowchart (Finding the Minimum)

```

graph TD
    Start([Start]) --> Init[min = 100  
i = 0]
    Init --> Decision1{Are there more students?}
    Decision1 -- No --> Output[Output min  
(smallest value)]
    Decision1 -- Yes --> Read[Read student X]
    Read --> Decision2{X.maths < min?}
    Decision2 -- No --> Decision1
    Decision2 -- Yes --> Update[min = X.maths]
    Update --> Decision1
    
```

Finding all students with the minimum
To get the IDs of all students who have the minimum value, use a list called minIds.

Pseudocode

```
min = 100
minIds = []
For each student X:
  If X.maths < min:
    min = X.maths
    minIds = [X.uniqueId]
  Else if X.maths == min:
    Add X.uniqueId to minIds
```

Example (with ties)

Student ID	Maths Mark	min	minIds	Action
S1	65	100	[S1]	New min (replace)
S2	52	65	[S2]	New min (replace)
S3	52	52	[S2, S3]	Equal to min (add)
S4	78	52	[S2, S3]	No change
S5	52	52	[S2, S3, S5]	Equal to min (add)

Minimum mark = 52, Students = S2, S3, S5

5. Example: Longest Sentence

Problem
To find the sentence containing the most words, keep two variables:

- count**: number of words in the current sentence.
- max**: largest sentence length found so far.

Steps

- Examine each word in order.
- Increase count for each word.
- If the word ends with a full stop (.), the sentence is complete.
- Compare count with max.
- If count is larger, update max.
- Reset count to 0 for the next sentence.

Pseudocode

```
count = 0
max = 0
For each word X, in paragraph order:
  count = count + 1
  If X.word ends with ".":
    If count > max:
      max = count
    count = 0
```

The dynamic comparison is count > max.
The value of max may change after each sentence.

Example
Paragraph: "I love programming. It is very interesting. Keep practising."

Word	count	Ends with .?	Action	max
I	1	No	–	0
love	2	No	–	0
programming.	3	Yes	Update max = 3	3
It	1	No	–	3
is	2	No	–	3
very	3	No	–	3
interesting.	4	Yes	Update max = 4	4
Keep	1	No	–	4
practising.	2	Yes	No change	4

Sentence lengths:
1st = 3 words
2nd = 4 words
3rd = 2 words

The longest sentence has 4 words:
"It is very interesting."

6. Combining filter types

A problem may use several kinds of conditions together.

Static condition

Checks a fixed property of the current item.

`X.partOfSpeech == "Verb"`

Dynamic condition

Compares a field with a variable that can change.

`X.letterCount > max`

State condition

Tracks where we are in a sequence or what has happened earlier. For example, a sentenceNumber variable can increase each time a word ends with a full stop.

Example: Longest noun in the third sentence

We need to:

- Track which sentence we are reading: **state**.
- Select nouns: **static filter**.
- Compare word lengths with the current longest length: **dynamic filter**.

The approach is to keep a sentence counter and a maximum word length, process nouns in the third sentence, and stop after that sentence has been completed.

Combining Filter Types: Static + Dynamic + State

A real problem may use more than one type of condition together.

1. Static Condition (Fixed value)

Checks a fixed property of the current item. The condition stays the same for every item.

`X.partOfSpeech == "Verb"`

Example: Select only verbs from the words in a sentence.

Verb

- run
- eat
- play

2. Dynamic Condition (Changing value)

Compares a field with a variable that can change during the iteration.

`X.letterCount > max`

Example: Keep track of the longest word by comparing each word's length with max.

max may change

3. State Condition (Remember information)

Tracks where we are in a sequence or what has happened earlier.

`sentenceNumber = sentenceNumber + 1`

Example: Increase sentenceNumber each time a word ends with a full stop.

- 1.
- 2.
- 3.

4. Example Problem

Longest noun in the third sentence

We need to:

1. Track which sentence we are reading. (State condition)
2. Select nouns from the words. (Static condition)
3. Compare word lengths with the current longest length. (Dynamic condition)
4. Stop after the third sentence has been completed.

Find the longest noun in the third sentence!

5. Algorithm Steps

1. Set sentenceNumber = 1, max = 0 and longestNoun = "".
2. Read each word in order.
3. If the word ends with a full stop, the sentence is complete. Increase sentenceNumber.
4. If sentenceNumber == 3 and the word is a noun (static filter), compare its length with max (dynamic filter). If it is greater, update max and longestNoun.
5. Stop after the third sentence has been completed.

6. Pseudocode

```

sentenceNumber = 1
max = 0
longestNoun = ""

for each word X, in order:
    if X.word ends with ".":
        sentenceNumber = sentenceNumber + 1
        if sentenceNumber > 3:
            break # stop after third sentence

    if sentenceNumber == 3 and
       X.partOfSpeech == "Noun": # static
        if X.letterCount > max: # dynamic
            max = X.letterCount
            longestNoun = X.word
        
```

7. Flowchart

```

graph TD
    Start([Start]) --> Init[sentenceNumber = 1  
max = 0  
longestNoun = ""]
    Init --> Read[Read next word X]
    Read --> EndDot{X ends with "."?}
    EndDot -- No --> IsNoun{Is sentenceNumber == 3  
and X is a Noun?}
    EndDot -- Yes --> IncSentence[sentenceNumber =  
sentenceNumber + 1]
    IsNoun -- No --> Read
    IsNoun -- Yes --> LetterCount{X.letterCount > max?}
    LetterCount -- No --> Read
    LetterCount -- Yes --> Update[max = X.letterCount  
longestNoun = X.word]
    Update --> Read
    IncSentence --> Sentence3{sentenceNumber > 3?}
    Sentence3 -- No --> Read
    Sentence3 -- Yes --> End([End])
    
```

8. Example

Paragraph: The cat sat on the mat. Dogs run quickly in the park. A big elephant walks near the river.

Sentence	Words (Nouns)	Length	Longest so far
1	cat, mat	3, 3	-
2	Dogs, park	4, 4	-
3	elephant, river	8, 5	elephant (8)

Output: The longest noun in the third sentence is "elephant" (length 8).

9. Remember

- Static: compare with a fixed value.
- Dynamic: compare with a variable that may change.
- State: remember information about what has happened earlier.
- For a maximum, update when the new value is greater.
- For a minimum, update when the new value is smaller.
- To keep every tied maximum or minimum, store matching items in a list.

Remember

- **Static:** compare with a fixed value.
- **Dynamic:** compare with a variable that may change.
- **State:** remember information about what has happened earlier in the iteration.
- For a maximum, update when the new value is **greater**.
- For a minimum, update when the new value is **smaller**.
- To keep every tied maximum or minimum, store matching items in a list.