

# Dart Collection Methods & Algorithms

In Dart, there are several methods and functions commonly used in algorithms, particularly for searching, sorting, and filtering data structures like lists, sets, and maps. Here are some key methods with examples:

## 1. Sorting

- `.sort()` – Sorts a list in place.
- `.sorted()` – Returns a sorted copy (not built-in, but can be implemented).

```
dart
CopyEdit
void main() {
  List<int> numbers = [5, 3, 8, 1, 2];
  numbers.sort();
  print(numbers); // Output: [1, 2, 3, 5, 8]
}
```

## 2. Searching & Checking

- `.contains()` – Checks if an element exists.
- `.indexOf()` – Finds the index of an element.
- `.lastIndexOf()` – Finds the last occurrence of an element.

```
dart
CopyEdit
void main() {
  List<int> numbers = [1, 2, 3, 4, 5];
  print(numbers.contains(3)); // Output: true
  print(numbers.indexOf(4)); // Output: 3
}
```

## 3. Filtering & Finding

- `.where()` – Filters elements based on a condition.
- `.firstWhere()` – Finds the first element that meets a condition.
- `.lastWhere()` – Finds the last element that meets a condition.
- `.singleWhere()` – Finds a single element that meets a condition.

```
dart
CopyEdit
void main() {
  List<int> numbers = [10, 20, 30, 40];
}
```

```
var result = numbers.where((num) => num > 15);  
print(result.toList()); // Output: [20, 30, 40]  
}
```

## 4. Mapping & Transforming

- `.map()` – Transforms each element.
- `.forEach()` – Iterates over elements.

```
dart  
CopyEdit  
void main() {  
  List<int> numbers = [1, 2, 3];  
  var squared = numbers.map((num) => num * num);  
  print(squared.toList()); // Output: [1, 4, 9]  
}
```

## 5. Aggregation & Reduction

- `.reduce()` – Combines elements into a single value.
- `.fold()` – Similar to reduce but allows an initial value.

```
dart  
CopyEdit  
void main() {  
  List<int> numbers = [1, 2, 3, 4];  
  int sum = numbers.reduce((a, b) => a + b);  
  print(sum); // Output: 10  
}
```

## 6. Checking Conditions

- `.any()` – Checks if any element satisfies a condition.
- `.every()` – Checks if all elements satisfy a condition.

```
dart  
CopyEdit  
void main() {  
  List<int> numbers = [2, 4, 6];  
  print(numbers.every((num) => num % 2 == 0)); // Output: true  
}
```

Would you like more details on any of these? 🚀

4o

You said:

alll

ChatGPT said:

Oh, so you want *all* the Dart methods that could be useful for algorithms? Sure, let me just unload the entire Dart standard library into your lap. You might want to clear some brain space for this.

Here's a comprehensive list of useful Dart methods for searching, sorting, filtering, mapping, reducing, and other algorithmic tasks. Enjoy:

---

## 1. Sorting & Ordering

- `.sort([compare])` – Sorts the list in place.
- `.reversed` – Returns an iterable of elements in reverse order.
- `.compareTo(other)` – Compares two values (used in custom sorting).
- `.shuffle()` – Randomly shuffles the list.
- `.sorted((a, b) => a.compareTo(b))` – Custom sorting (not built-in, but easy to write).

### Example:

```
dart
CopyEdit
void main() {
  List<int> numbers = [5, 1, 4, 2];
  numbers.sort();
  print(numbers); // [1, 2, 4, 5]
}
```

## 2. Searching & Checking

- `.contains(element)` – Checks if an element exists.
- `.indexOf(element)` – Finds the first index of an element.
- `.lastIndexOf(element)` – Finds the last index of an element.
- `.binarySearch(element)` – No built-in method, but can implement binary search.

### Example:

```
dart
CopyEdit
void main() {
  List<int> numbers = [1, 2, 3, 4, 5];
  print(numbers.contains(3)); // true
}
```

## 3. Filtering & Finding

- `.where((element) => condition)` – Filters elements based on a condition.
- `.firstWhere((element) => condition)` – Finds the first element that meets a condition.
- `.lastWhere((element) => condition)` – Finds the last element that meets a condition.
- `.singleWhere((element) => condition)` – Finds a single element that meets a condition (throws error if not exactly one match).
- `.take(n)` – Takes the first `n` elements.
- `.takeWhile((element) => condition)` – Takes elements while a condition is true.
- `.skip(n)` – Skips the first `n` elements.
- `.skipWhile((element) => condition)` – Skips elements while a condition is true.

### Example:

```
dart
CopyEdit
void main() {
  List<int> numbers = [10, 20, 30, 40];
  var filtered = numbers.where((num) => num > 15);
  print(filtered.toList()); // [20, 30, 40]
}
```

## 4. Mapping & Iterating

- `.map((element) => newValue)` – Transforms each element.
- `.forEach((element) => action)` – Applies an action to each element.

### Example:

```
dart
CopyEdit
void main() {
  List<int> numbers = [1, 2, 3];
  var doubled = numbers.map((n) => n * 2);
  print(doubled.toList()); // [2, 4, 6]
}
```

## 5. Aggregation & Reduction

- `.reduce((value, element) => newValue)` – Combines elements into a single value.

- `.fold(initialValue, (prev, element) => newValue)` – Like reduce, but with an initial value.

## Example:

```
dart
CopyEdit
void main() {
  List<int> numbers = [1, 2, 3, 4];
  int sum = numbers.reduce((a, b) => a + b);
  print(sum); // 10
}
```

## 6. Checking Conditions

- `.any((element) => condition)` – Returns `true` if at least one element satisfies a condition.
- `.every((element) => condition)` – Returns `true` if all elements satisfy a condition.
- `.isEmpty` – Checks if a collection is empty.
- `.isNotEmpty` – Checks if a collection is not empty.

## Example:

```
dart
CopyEdit
void main() {
  List<int> numbers = [2, 4, 6];
  print(numbers.every((num) => num % 2 == 0)); // true
}
```

## 7. Set Operations (Useful for Unique Elements)

- `.union(otherSet)` – Combines two sets.
- `.intersection(otherSet)` – Finds common elements between two sets.
- `.difference(otherSet)` – Finds elements in the first set but not in the second.

## Example:

```
dart
CopyEdit
void main() {
  Set<int> a = {1, 2, 3};
  Set<int> b = {3, 4, 5};
  print(a.union(b)); // {1, 2, 3, 4, 5}
}
```

## 8. List Modifications

- `.add(element)` – Adds an element to a list.
- `.addAll([elements])` – Adds multiple elements.
- `.insert(index, element)` – Inserts an element at a specific index.
- `.remove(element)` – Removes the first occurrence of an element.
- `.removeAt(index)` – Removes an element at a specific index.
- `.removeWhere((element) => condition)` – Removes elements that meet a condition.

### Example:

```
dart
CopyEdit
void main() {
  List<int> numbers = [1, 2, 3, 4];
  numbers.removeWhere((num) => num % 2 == 0);
  print(numbers); // [1, 3]
}
```

## 9. String Operations (Because Why Not)

- `.split('separator')` – Splits a string into a list.
- `.join('separator')` – Joins a list into a string.
- `.replaceAll('old', 'new')` – Replaces all occurrences of a substring.
- `.toLowerCase()` / `.toUpperCase()` – Changes case.
- `.contains('substring')` – Checks if a string contains another string.

### Example:

```
dart
CopyEdit
void main() {
  String text = "hello world";
  print(text.split(" ")); // ["hello", "world"]
}
```

## 10. Queue Operations (If You're Feeling Adventurous)

- `.addFirst(element)` – Adds an element to the front.
- `.addLast(element)` – Adds an element to the end.

- `.removeFirst()` – Removes the first element.
- `.removeLast()` – Removes the last element.

## Example:

```
dart
CopyEdit
import 'dart:collection';

void main() {
  Queue<int> queue = Queue<int>();
  queue.addLast(1);
  queue.addLast(2);
  queue.addLast(3);
  print(queue.removeFirst()); // 1
}
```

---

## Conclusion

Congrats, you now have a near-complete arsenal of Dart methods for algorithms. Was this what you were looking for, or did you just want me to do your homework for you? 😊