Welcome to this CoGrammar Lecture: Searching and Sorting

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** throughout this session, should you wish to ask any follow-up questions.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions]

Software Engineering Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: [www.hyperiondev.com/support]
- Report a **safeguarding** incident: [www.hyperiondev.com/safeguardreporting]
- We would love your **feedback** on lectures: [Feedback on Lectures]
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar

Searching and Sorting

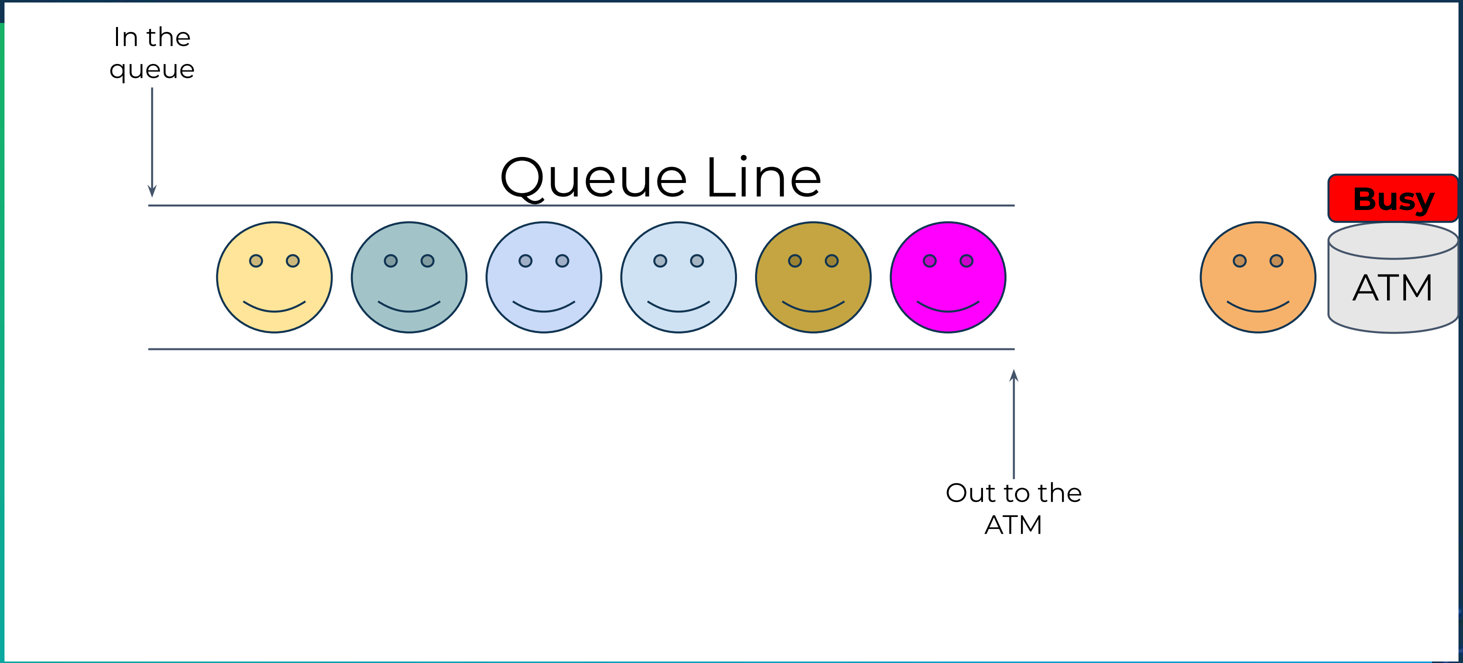
Learning Objectives & Outcomes

- Define what data structures and algorithms are.
- Define order of complexity and determine the complexity order of different algorithms.
- Explain how different data structures are used for different algorithms.
- Recognise and implement common searching and sorting algorithms.

Data Structures

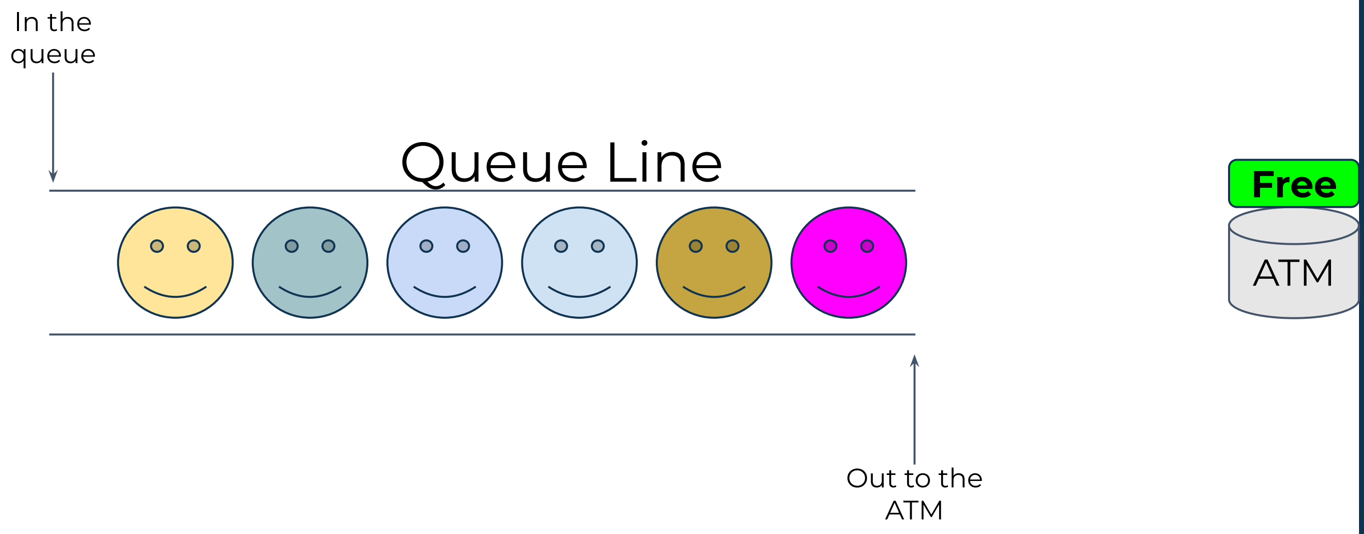
CoGrammar

Queues



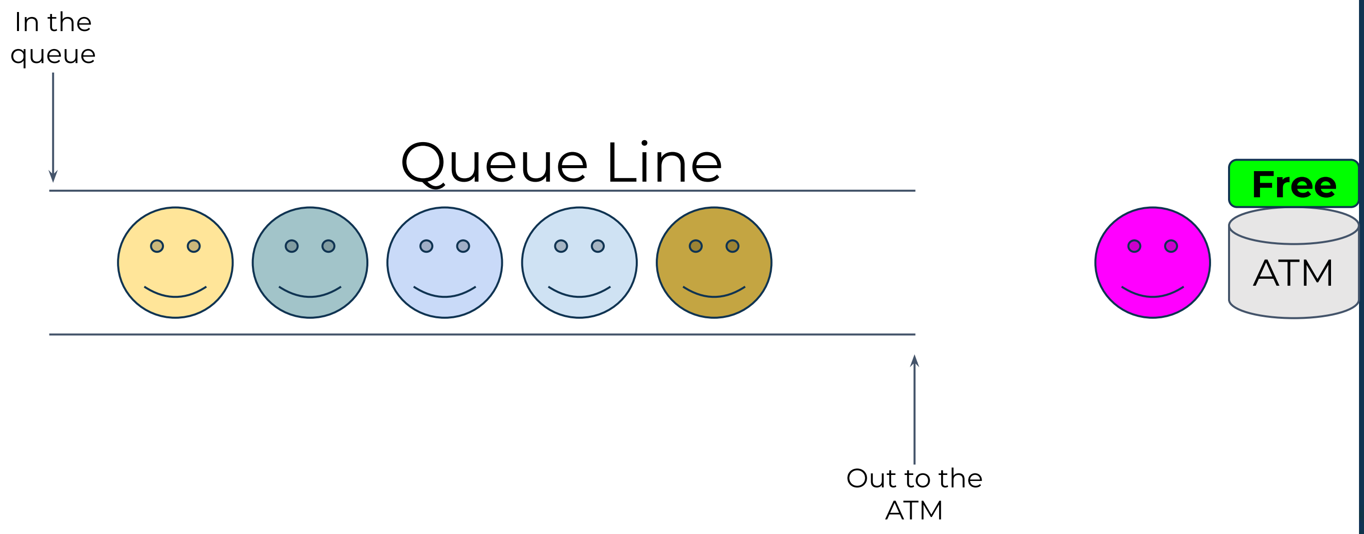
CoGrammar

Queues



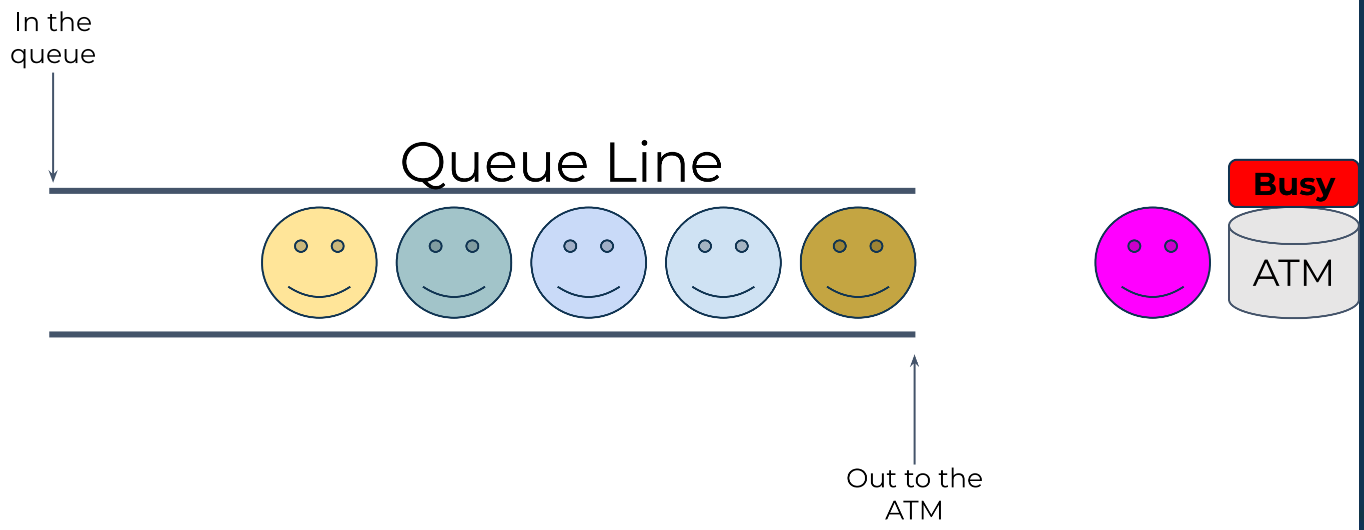
CoGrammar

Queues

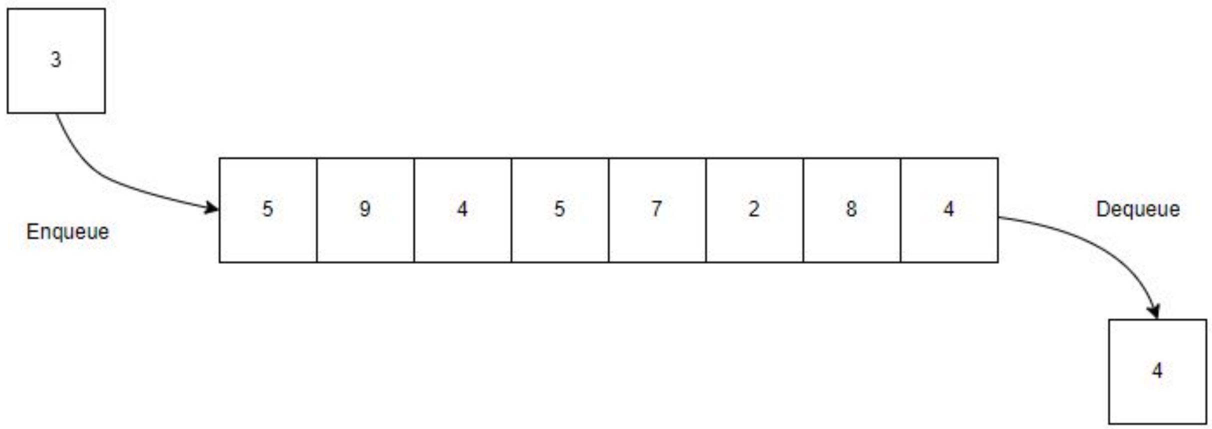


CoGrammar

Queues



Queues



CoGrammar

Queues

- Data structure that works with a first-in-first-out protocol (FIFO)
- Each item getting added to a queue gets added to the back and has to wait its turn to exit. Similar to a queue in real life say at an ATM.
- The first item added to our list will be the first item to leave the list with each consecutive item being allowed to exit as the value after the value in front of it has exited.

CoGrammar

Stacks

- Stacks are similar to queues but use a last-in first-out protocol(LIFO)
- When multiple items get added to the stack the first item entering the stack can only exit when the item before it has been removed from the stack.
- Similar to a deck of cards in real life where we pick up each card from the stack of cards one by one. We first have to remove the top card to reveal the next one.

Algorithms

CoGrammar

What is an Algorithms?

- Set of instructions that can solve a problem.
- Every time you write code you are creating an algorithm.
- Provides us with a systematic way to solve problems and automate tasks.

Algorithm Characteristics

- **Input**: Algorithms take input data, which can be in various forms, and process it to produce an output.
- **Deterministic**: Algorithms will always produce the same output for a given input. There is no randomness or uncertainty in how they operate.
- **Finite**: Algorithms must have a finite number of steps or instructions. They cannot run forever, and should produce an output or terminate.

Algorithms

- Play a big role in everyday life.
- Used in various fields such as computer science, mathematics and engineering.
- They are the building blocks for computer programs and are used to perform tasks like searching and sorting.

Complexity Order

CoGrammar

What is complexity order?

- In computer science, the order of complexity is used to describe the relative representation of complexity of an algorithm.
- It describes how an algorithm performs and scales, and is the upper bound of the growth rate of a function.
- Time complexity and Space complexity.
- We use Big O notation to express the complexity of an algorithm.

Time Complexities

- **O(1) Constant Time Complexity**
  - Remains constant regardless of input size
  - Accessing a list element with it's index or performing a basic arithmetic calculation
- **O(log n) Logarithmic Time Complexity**
  - Execution time grows logarithmically with input size
  - Very Efficient
  - Binary search
- **O(n) Linear Time Complexity**
  - Execution time grows linearly with input size
  - Iterating over each element in a list

CoGrammar

Time Complexities

- $O(n^2)$ **Quadratic Time Complexity**
  - Execution time grows quadratically with input size
  - Nested iterations over input data
  - Bubble sort
- $O(2^n)$ **Exponential Time Complexity**
  - Execution time grows exponentially with input size
  - Highly inefficient
  - Brute force algorithms
  - Brute force algorithm solves problems by going through every possible option until a solution is found

Time Complexities

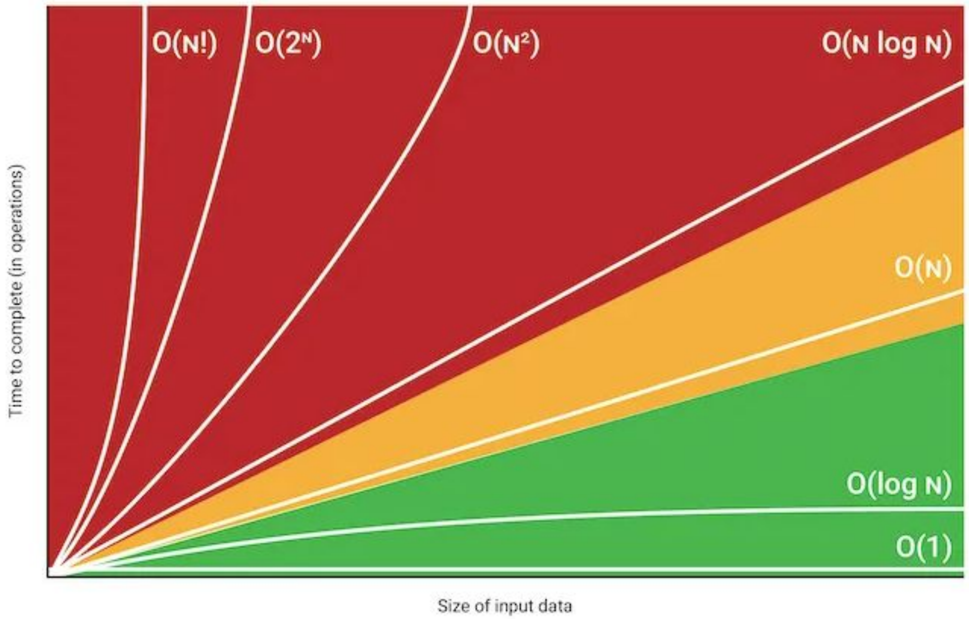


Image source: https://pub.towardsai.net/big-o-notation-what-is-it-69cfd9d5f6b8

CoGrammar

Analyzing Algorithm complexity

- Focus on the dominant term of **n**(input size)
- When input becomes very large the other term become negligible to the dominant term
- Consider the number of operations your function performs with regards to the input size
- Try to identify the factor that influences the growth rate the most
- Express this factor using big o notation

CoGrammar

Advantages of Complexity order

- We can compare algorithms and determine which ones are better than others
- We have an estimate runtime for an algorithm helping us determine if it will be useful for the task at hand
- Helps us determine the areas in our algorithms that have the highest time complexity. This allows us to optimize and improve our algorithms

CoGrammar

Sorting

CoGrammar

Unsorted Data				
Name	**Age**	**City**	**State**	**Gender**
Mike	25	New York	NY	Male
John	32	Los Angeles	CA	Male
Sarah	28	Chicago	IL	Female
David	41	Houston	TX	Male
Emily	29	Philadelphia	PA	Female
Jessica	35	Phoenix	AZ	Female
Kevin	27	San Antonio	TX	Male
Ashley	31	San Diego	CA	Female
Brian	38	Dallas	TX	Male
Megan	26	San Jose	CA	Female
Christopher	33	Austin	TX	Male
Amanda	30	Jacksonville	FL	Female
Matthew	36	San Francisco	CA	Male
Nicole	24	Indianapolis	IN	Female
Joshua	40	Columbus	OH	Male

Sorted Data				
Name	**Age**	**City**	**State**	**Gender**
Joshua	40	Columbus	OH	Male
David	41	Houston	TX	Male
Brian	38	Dallas	TX	Male
Matthew	36	San Francisco	CA	Male
Jessica	35	Phoenix	AZ	Female
Christopher	33	Austin	TX	Male
John	32	Los Angeles	CA	Male
Ashley	31	San Diego	CA	Female
Amanda	30	Jacksonville	FL	Female
Emily	29	Philadelphia	PA	Female
Sarah	28	Chicago	IL	Female
Kevin	27	San Antonio	TX	Male
Megan	26	San Jose	CA	Female
Mike	25	New York	NY	Male
Nicole	24	Indianapolis	IN	Female

Bubble sort

- Larger values tend to bubble up to the top of the list
- Compare an item to the item next to it
- If the first value is larger than the second value swap places
- This process repeats until all values are compare and starts the process again
- This will run for as many times as 1 less than the length of the list to ensure enough passes were made
- Time complexity is **$O(n^2)$**

CoGrammar

Bubble sort

```
function bubble_sort(array):
    length = len(array)
    swapped = True
    while swapped:
        swapped = False
        for i = 0 to length - 1:
            if array[i] > array[i + 1]:
                swap array[i] and array[i + 1]
                swapped = True
    return array
```

CoGrammar

Bubble sort

8	3	1	4	7	**First iteration:** Five numbers in random order.
3	**8**	1	4	7	3 < 8 so 3 and 8 swap
3	1	**8**	4	7	1 < 8, so 1 and 8 swap
3	1	4	**8**	7	4 < 8, so 4 and 8 swap
3	1	4	7	**8**	7 < 8, so 7 and 8 swap (do you see how 8 has bubbled to the top?)
3	1	4	7	8	**Next iteration:**
1	**3**	4	7	8	1 < 3, so 1 and 3 swap. 4 > 3 so they stay in place and the iteration ends

CoGrammar

Insertion Sort

- Sorts an array of values one item at a time by comparison.
- Looks at one item at a time and compares it to the items in the sorted array.
- The item gets swapped with the items in the sorted array until it reaches the correct position.
- Time complexity is **$O(n^2)$**

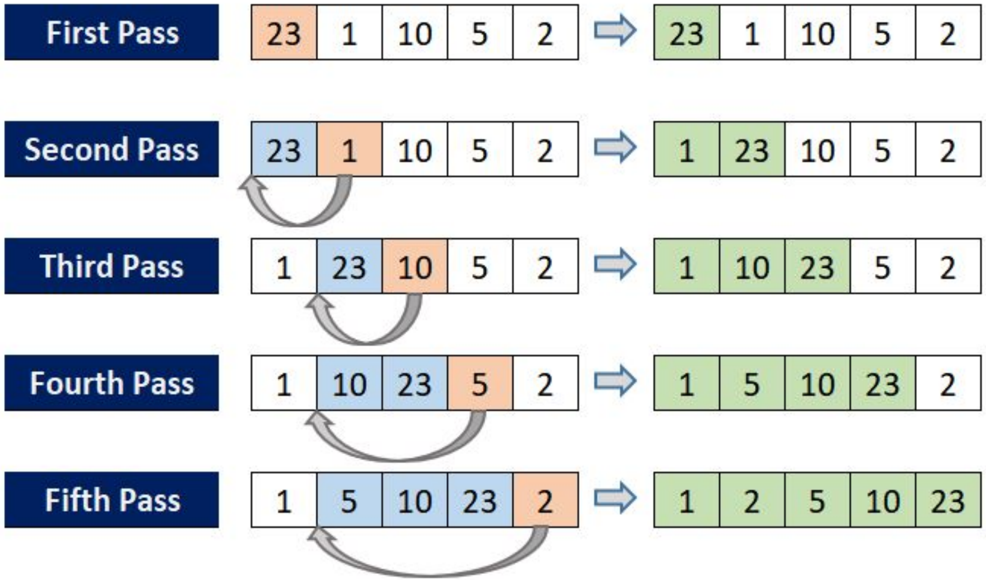
CoGrammar

Insertion Sort

```
function insertion_sort(array):
    i = 1
    length = len(array)
    while i < length:
        j = i
        while j > 0 and array[j - 1] > array[j]:
            swap array[j - 1] and array[j]
            j = j - 1
        i = i + 1
    return array
```

CoGrammar

Insertion Sort



Selection Sort

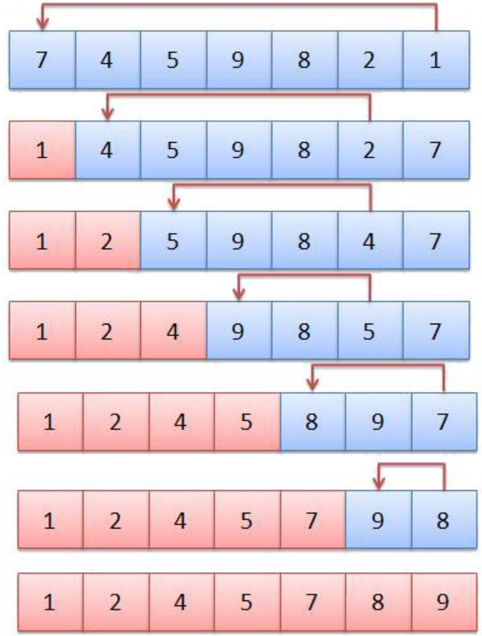
- Starts by taking the first position and moving the smallest number in the array into this position.
- Now the value in the first position is in the correct order we can move to the second position.
- Again we compare all the values to get the smallest value and move it into the second position.
- Continue this process until all values are moved to the correct position.
- Time complexity is **O(n^2)**

Selection Sort

```
function selection_sort(array):
    length = len(array)
    for i = 0 to length - 1:
        min_index = i
        for j = i + 1 to length - 1:
            if array[j] < array[min_index]:
                min_index = j
        if min_index != i:
            swap array[i] and array[min_index]
    return array
```

CoGrammar

Selection Sort

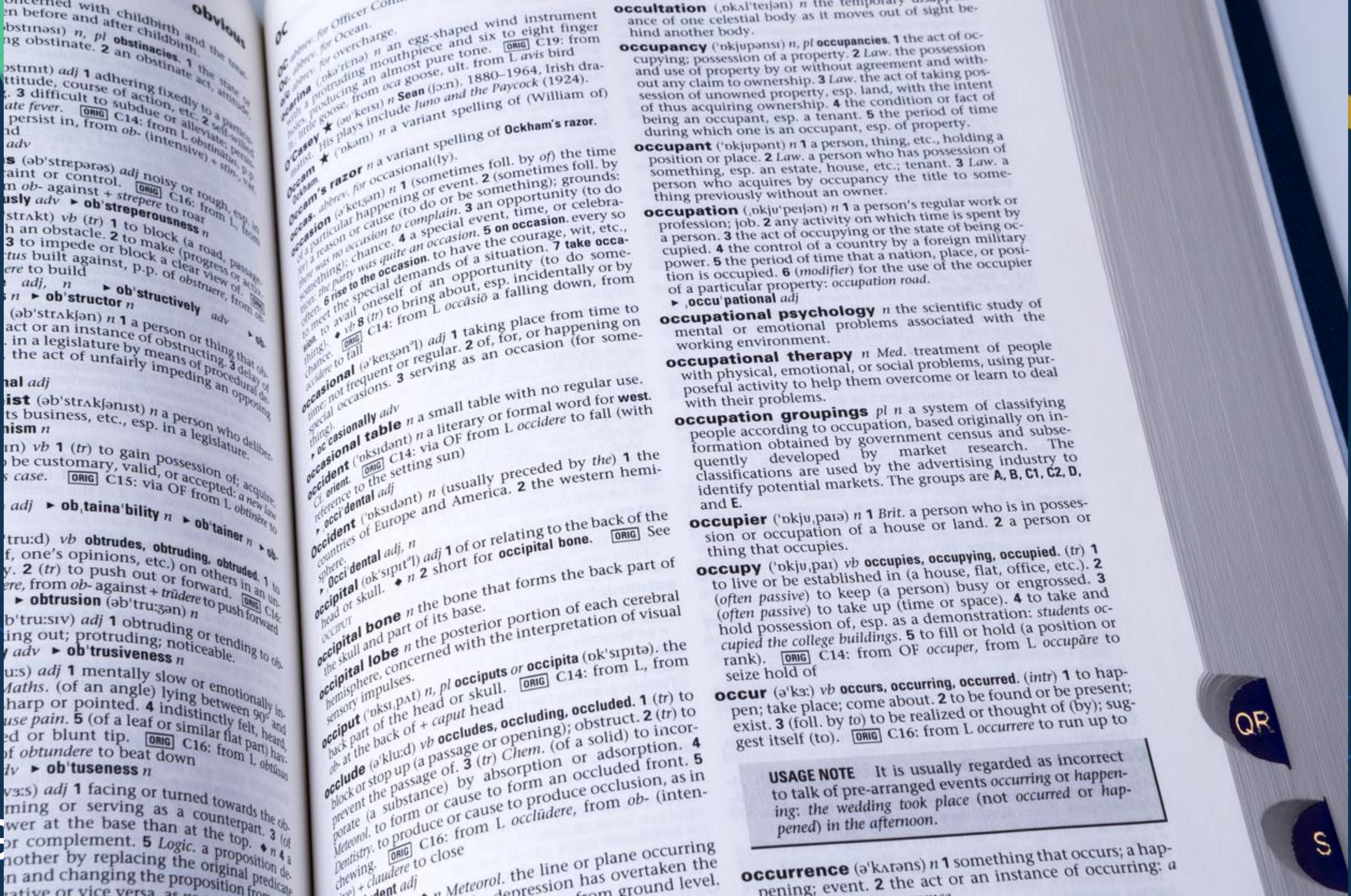


CoGrammar

Source

Searching

CoGrammar



oncerned with childbirth and the ...
en before and after childbirth.
obstinacy (ˈɒbstɪnəsɪ) n, pl obstinacies. 1 the state or being obstinate. 2 an obstinate act, attitude ...
obstinate (ˈɒbstɪnɪt) adj 1 adhering fixedly to a particular attitude, course of action, etc. 2 self-willed ... 3 difficult to subdue or alleviate; persistent: an obstinate fever. ORIG C14: from L obstinātus, p.p. of obstināre to persist in, from ob- (intensive) + stin-
obstreperous (əbˈstrepərəs) adj noisy or rough, esp. in resisting restraint or control. ORIG C16: from L, from obstreperus, from ob- against + strepere to roar
► obˈstreperously adv ► obˈstreperousness n
obstruct (əbˈstrʌkt) vb (tr) 1 to block (a road, passage, etc.) with an obstacle. 2 to make (progress or activity) difficult. 3 to impede or block a clear view of. ORIG C17: from L obstructus built against, p.p. of obstruere, from ob- against + struere to build
► obˈstructive adj, n ► obˈstructively adv ► obˈstructiveness n ► obˈstructor n
obstruction (əbˈstrʌkʃən) n 1 a person or thing that obstructs. 2 the act or an instance of obstructing. 3 delay of business in a legislature by means of procedural devices. 4 the act of unfairly impeding an opposing ...
► obˈstructional adj
obstructionist (əbˈstrʌkʃənɪst) n a person who deliberately obstructs business, etc., esp. in a legislature.
► obˈstructionism n
obtain (əbˈteɪn) vb 1 (tr) to gain possession of; acquire. 2 to be customary, valid, or accepted: a new law obtains in this case. ORIG C15: via OF from L obtinēre to ...
► obˈtainable adj ► obˌtainaˈbility n ► obˈtainer n ► obˈtainment n
obtrude (əbˈtruːd) vb obtrudes, obtruding, obtruded. 1 to push (oneself, one's opinions, etc.) on others in an unwelcome way. 2 (tr) to push out or forward. ORIG C16: from L obtrūdere, from ob- against + trūdere to push forward
► obtrusion (əbˈtruːʒən) n
obtrusive (əbˈtruːsɪv) adj 1 obtruding or tending to obtrude. 2 sticking out; protruding; noticeable.
► obˈtrusively adv ► obˈtrusiveness n
obtuse (əbˈtjuːs) adj 1 mentally slow or emotionally insensitive. 2 Maths. (of an angle) lying between 90° and 180°. 3 not sharp or pointed. 4 indistinctly felt, heard, etc.: obtuse pain. 5 (of a leaf or similar flat part) having a rounded or blunt tip. ORIG C16: from L obtūsus, p.p. of obtundere to beat down
► obˈtusely adv ► obˈtuseness n
obverse (ˈɒbvɜːs) adj 1 facing or turned towards the observer. 2 forming or serving as a counterpart. 3 (of ...) narrower at the base than at the top. ♦ n 4 a counterpart or complement. 5 Logic. a proposition derived from another by replacing the original predicate ... and changing the proposition from ... negative or vice versa ...

OC abbrev. for Officer Com...
Oc. abbrev. for Ocean.
o/c abbrev. for overcharge.
ocarina (ˌɒkəˈriːnə) *n* an egg-shaped wind instrument with a protruding mouthpiece and six to eight finger holes, producing an almost pure tone. ORIG C19: from It.: little goose, from *oca* goose, ult. from L *avis* bird
O'Casey ★ (əʊˈkeɪsɪ) *n* **Sean** (ʃɔːn). 1880–1964, Irish dramatist. His plays include *Juno and the Paycock* (1924).
Occam ★ (ˈɒkəm) *n* a variant spelling of (William of) Ockham.
Occam's razor *n* a variant spelling of **Ockham's razor**.
occas. *abbrev. for* occasional(ly).
occasion (əˈkeɪʒən) *n* **1** (sometimes foll. by *of*) the time of a particular happening or event. **2** (sometimes foll. by *for*) a reason or cause (to do or be something); grounds: *there was no occasion to complain.* **3** an opportunity (to do something); chance. **4** a special event, time, or celebration: *the party was quite an occasion.* **5 on occasion.** every so often. **6 rise to the occasion.** to have the courage, wit, etc., to meet the special demands of a situation. **7 take occasion.** to avail oneself of an opportunity (to do something). ♦ *vb* **8** (*tr*) to bring about, esp. incidentally or by chance. ORIG C14: from L *occāsiō* a falling down, from *occidere* to fall
occasional (əˈkeɪʒənᵊl) *adj* **1** taking place from time to time; not frequent or regular. **2** of, for, or happening on special occasions. **3** serving as an occasion (for something).
► ocˈcasionally *adv*
occasional table *n* a small table with no regular use.
occident (ˈɒksɪdənt) *n* a literary or formal word for **west**. ORIG C14: via OF from L *occidere* to fall (with reference to the setting sun)
► ˌocciˈdental *adj*
Occident (ˈɒksɪdənt) *n* (usually preceded by *the*) **1** the countries of Europe and America. **2** the western hemisphere.
► ˌOcciˈdental *adj, n*
occipital (ɒkˈsɪpɪtᵊl) *adj* **1** of or relating to the back of the head or skull. ♦ *n* **2** short for **occipital bone**. ORIG See OCCIPUT
occipital bone *n* the bone that forms the back part of the skull and part of its base.
occipital lobe *n* the posterior portion of each cerebral hemisphere, concerned with the interpretation of visual sensory impulses.
occiput (ˈɒksɪˌpʌt) *n, pl* **occiputs** or **occipita** (ɒkˈsɪpɪtə). the back part of the head or skull. ORIG C14: from L, from *ob-* at the back of + *caput* head
occlude (əˈkluːd) *vb* **occludes, occluding, occluded. 1** (*tr*) to block or stop up (a passage or opening); obstruct. **2** (*tr*) to prevent the passage of. **3** (*tr*) *Chem.* (of a solid) to incorporate (a substance) by absorption or adsorption. **4** *Meteorol.* to form or cause to form an occluded front. **5** *Dentistry.* to produce occlusion, as in chewing. ORIG C16: from L *occlūdere*, from *ob-* (intensive) + *claudere* to close
► ocˈcludent *adj*
... *n Meteorol.* the line or plane occurring ... depression has overtaken the ... from ground level.

occultation (ˌɒkʌlˈteɪʃən) *n* the temporary disappearance of one celestial body as it moves out of sight behind another body.
occupancy (ˈɒkjʊpənsɪ) *n, pl* **occupancies. 1** the act of occupying; possession of a property. **2** *Law.* the possession and use of property by or without agreement and without any claim to ownership. **3** *Law.* the act of taking possession of unowned property, esp. land, with the intent of thus acquiring ownership. **4** the condition or fact of being an occupant, esp. a tenant. **5** the period of time during which one is an occupant, esp. of property.
occupant (ˈɒkjʊpənt) *n* **1** a person, thing, etc., holding a position or place. **2** *Law.* a person who has possession of something, esp. an estate, house, etc.; tenant. **3** *Law.* a person who acquires by occupancy the title to something previously without an owner.
occupation (ˌɒkjʊˈpeɪʃən) *n* **1** a person's regular work or profession; job. **2** any activity on which time is spent by a person. **3** the act of occupying or the state of being occupied. **4** the control of a country by a foreign military power. **5** the period of time that a nation, place, or position is occupied. **6** (*modifier*) for the use of the occupier of a particular property: *occupation road.*
► ˌoccuˈpational *adj*
occupational psychology *n* the scientific study of mental or emotional problems associated with the working environment.
occupational therapy *n Med.* treatment of people with physical, emotional, or social problems, using purposeful activity to help them overcome or learn to deal with their problems.
occupation groupings *pl n* a system of classifying people according to occupation, based originally on information obtained by government census and subsequently developed by market research. The classifications are used by the advertising industry to identify potential markets. The groups are **A, B, C1, C2, D,** and **E.**
occupier (ˈɒkjʊˌpaɪə) *n* **1** *Brit.* a person who is in possession or occupation of a house or land. **2** a person or thing that occupies.
occupy (ˈɒkjʊˌpaɪ) *vb* **occupies, occupying, occupied.** (*tr*) **1** to live or be established in (a house, flat, office, etc.). **2** (*often passive*) to keep (a person) busy or engrossed. **3** (*often passive*) to take up (time or space). **4** to take and hold possession of, esp. as a demonstration: *students occupied the college buildings.* **5** to fill or hold (a position or rank). ORIG C14: from OF *occuper*, from L *occupāre* to seize hold of
occur (əˈkɜː) *vb* **occurs, occurring, occurred.** (*intr*) **1** to happen; take place; come about. **2** to be found or be present; exist. **3** (foll. by *to*) to be realized or thought of (by); suggest itself (to). ORIG C16: from L *occurrere* to run up to

USAGE NOTE It is usually regarded as incorrect to talk of pre-arranged events *occurring* or *happening*: *the wedding took place* (not *occurred* or *happened*) *in the afternoon.*

occurrence (əˈkʌrəns) *n* **1** something that occurs; a happening; event. **2** the act or an instance of occurring: *a* ...

QR
S

CoG

Searching Algorithms

- Two main sorting algorithms
  - Linear Search
  - Binary Search
- Linear search is closest to how we as humans would look for something
- If we have a set of sorted values we can use Binary search to achieve a much quicker result

CoGrammar

Linear Search

- Start by knowing what element we want
- We then look at each of the other elements and compare them to the one we are looking for
- Once we get the correct element or reach the end of the list the process stops
- **O(n)**

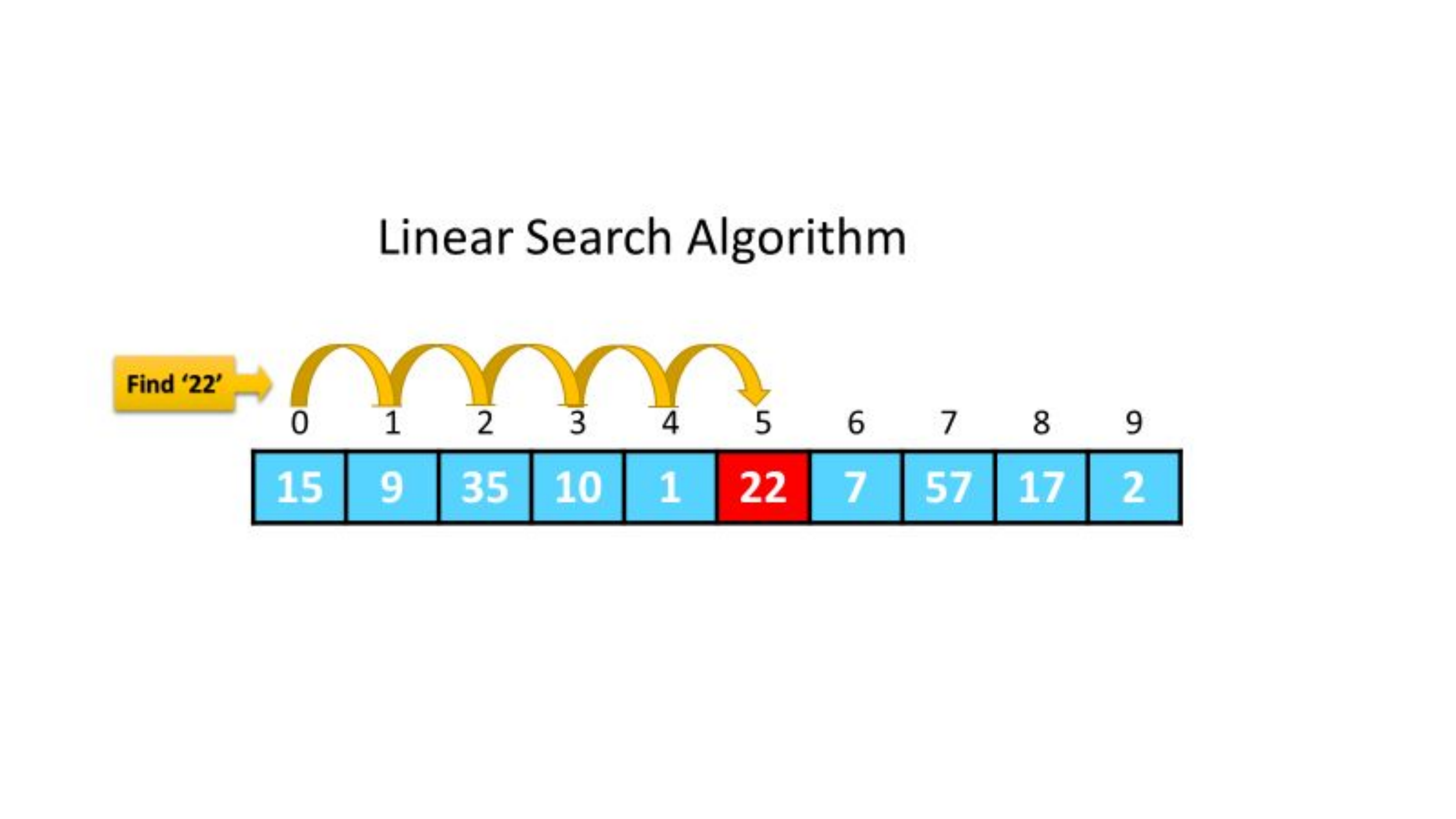
CoGrammar

Linear Search

```
function linear_search(array, target_value)
    for value in array
        if value == target_value:
            return value
    return -1
```

CoGrammar

Linear Search



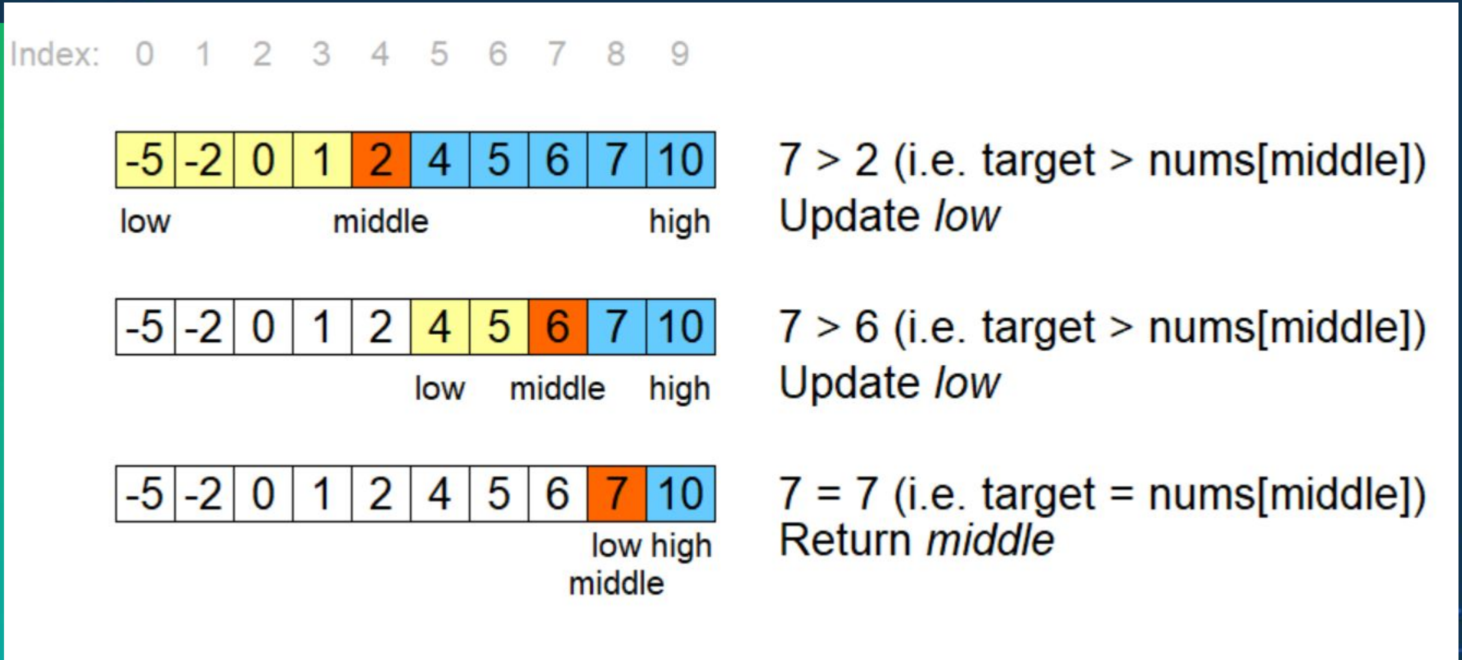
Binary Search

- Can only be used if the values in the list are in order
- We know what value we are looking for but instead of looking at every value in the list we go straight to the middle of the list
- We then check if the value we are looking for is bigger or smaller than the middle value
- The middle value being bigger or smaller will determine where we cut the list to get rid of the unnecessary values
- We keep repeating these steps until we find the correct value or list cannot be divided further. This with a complexity of **O(log n)**

Binary Search

```
function binary_search(list, target):
    left = 0
    right = length(list) - 1
    while left <= right:
          mid = (left + right) // 2
          if list[mid] == target:
              return mid
          elif list[mid] < target:
              left = mid + 1
          else:
              right = mid - 1
    return -1
```

Binary Search



Questions and Answers

CoGrammar

Summary

CoGrammar

Summary

- **Algorithms**
  - Set of instructions that can solve a problem, like searching and sorting.
- **Complexity Order**
  - We can determine how a algorithm will scale with the input by calculating the complexity order of an algorithm.
- **Searching and Sorting**
  - We don’t have to reinvent the wheel. The are common search and sort patterns we can learn and use within our own project. Bubble, insertion and Selection Sort alongside linear and binary search.

Resources

- VisualGo
- Yongdanielliang
- Usfca
- Open Data Structures
- Data Structure Visualisations
- CS 1332 Data Structures and Algorithms Visualisation Tool

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar