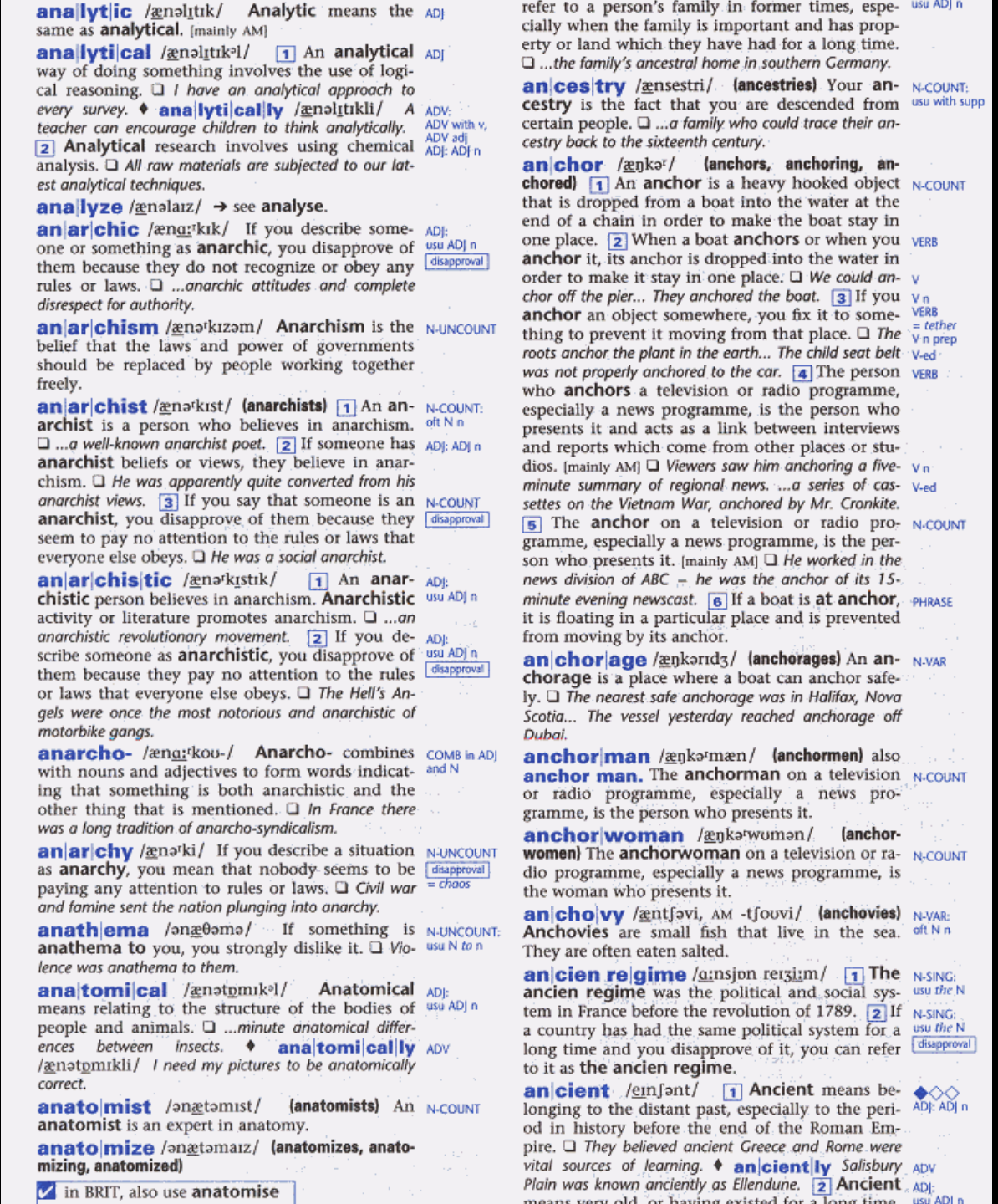


ana|lyt|ic /ænəlɪtɪk/ **Analytic** means the same as **analytical**. [mainly AM] ADJ

ana|lyt|ical /ænəlɪtɪkᵊl/ [1] An **analytical** way of doing something involves the use of logical reasoning. ❑ *I have an analytical approach to every survey.* ♦ **ana|lyt|ical|ly** /ænəlɪtɪkli/ *A teacher can encourage children to think analytically.* ADJ

ADV: ADV with v, ADV adj

[2] **Analytical** research involves using chemical analysis. ❑ *All raw materials are subjected to our latest analytical techniques.* ADJ: ADJ n

ana|lyze /ænəlaɪz/ → see **analyse**.

an|ar|chic /ænɑːrkɪk/ If you describe someone or something as **anarchic**, you disapprove of them because they do not recognize or obey any rules or laws. ❑ *...anarchic attitudes and complete disrespect for authority.* ADJ: usu ADJ n disapproval

an|ar|chism /ænərkɪzəm/ **Anarchism** is the belief that the laws and power of governments should be replaced by people working together freely. N-UNCOUNT

an|ar|chist /ænərkɪst/ **(anarchists)** [1] An **anarchist** is a person who believes in anarchism. ❑ *...a well-known anarchist poet.* [2] If someone has **anarchist** beliefs or views, they believe in anarchism. ❑ *He was apparently quite converted from his anarchist views.* [3] If you say that someone is an **anarchist**, you disapprove of them because they seem to pay no attention to the rules or laws that everyone else obeys. ❑ *He was a social anarchist.* N-COUNT: oft N n

ADJ: ADJ n

N-COUNT disapproval

an|ar|chis|tic /ænərkɪstɪk/ [1] An **anarchistic** person believes in anarchism. **Anarchistic** activity or literature promotes anarchism. ❑ *...an anarchistic revolutionary movement.* [2] If you describe someone as **anarchistic**, you disapprove of them because they pay no attention to the rules or laws that everyone else obeys. ❑ *The Hell's Angels were once the most notorious and anarchistic of motorbike gangs.* ADJ: usu ADJ n

ADJ: usu ADJ n disapproval

anarcho- /ænɑːrkoʊ-/ **Anarcho-** combines with nouns and adjectives to form words indicating that something is both anarchistic and the other thing that is mentioned. ❑ *In France there was a long tradition of anarcho-syndicalism.* COMB in ADJ and N

an|ar|chy /ænərki/ If you describe a situation as **anarchy**, you mean that nobody seems to be paying any attention to rules or laws. ❑ *Civil war and famine sent the nation plunging into anarchy.* N-UNCOUNT disapproval = chaos

anath|ema /ənæθəmə/ If something is **anathema to** you, you strongly dislike it. ❑ *Violence was anathema to them.* N-UNCOUNT: usu N to n

ana|tom|ical /ænətɒmɪkᵊl/ **Anatomical** means relating to the structure of the bodies of people and animals. ❑ *...minute anatomical differences between insects.* ♦ **ana|tom|ical|ly** /ænətɒmɪkli/ *I need my pictures to be anatomically correct.* ADJ: usu ADJ n

ADV

anato|mist /ənætəmɪst/ **(anatomists)** An **anatomist** is an expert in anatomy. N-COUNT

anato|mize /ənætəmaɪz/ **(anatomizes, anatomizing, anatomized)**

in BRIT, also use **anatomise**

refer to a person's family in former times, especially when the family is important and has property or land which they have had for a long time. ❑ *...the family's ancestral home in southern Germany.* usu ADJ n

an|ces|try /ænsestri/ **(ancestries)** Your **ancestry** is the fact that you are descended from certain people. ❑ *...a family who could trace their ancestry back to the sixteenth century.* N-COUNT: usu with supp

an|chor /æŋkəʳ/ **(anchors, anchoring, anchored)** [1] An **anchor** is a heavy hooked object that is dropped from a boat into the water at the end of a chain in order to make the boat stay in one place. [2] When a boat **anchors** or when you **anchor** it, its anchor is dropped into the water in order to make it stay in one place. ❑ *We could anchor off the pier... They anchored the boat.* [3] If you **anchor** an object somewhere, you fix it to something to prevent it moving from that place. ❑ *The roots anchor the plant in the earth... The child seat belt was not properly anchored to the car.* [4] The person who **anchors** a television or radio programme, especially a news programme, is the person who presents it and acts as a link between interviews and reports which come from other places or studios. [mainly AM] ❑ *Viewers saw him anchoring a five-minute summary of regional news. ...a series of cassettes on the Vietnam War, anchored by Mr. Cronkite.* [5] The **anchor** on a television or radio programme, especially a news programme, is the person who presents it. [mainly AM] ❑ *He worked in the news division of ABC – he was the anchor of its 15-minute evening newscast.* [6] If a boat is **at anchor**, it is floating in a particular place and is prevented from moving by its anchor. N-COUNT

VERB

V

V n

VERB = tether

V n prep

V-ed

VERB

V n

V-ed

N-COUNT

PHRASE

an|chor|age /æŋkərɪdʒ/ **(anchorages)** An **anchorage** is a place where a boat can anchor safely. ❑ *The nearest safe anchorage was in Halifax, Nova Scotia... The vessel yesterday reached anchorage off Dubai.* N-VAR

anchor|man /æŋkəʳmæn/ **(anchormen)** also **anchor man.** The **anchorman** on a television or radio programme, especially a news programme, is the person who presents it. N-COUNT

anchor|woman /æŋkəʳwʊmən/ **(anchorwomen)** The **anchorwoman** on a television or radio programme, especially a news programme, is the woman who presents it. N-COUNT

an|cho|vy /æntʃəvi, AM -tʃoʊvi/ **(anchovies)** **Anchovies** are small fish that live in the sea. They are often eaten salted. N-VAR: oft N n

an|cien re|gime /ɑːnsjɒn reɪʒiːm/ [1] **The ancien regime** was the political and social system in France before the revolution of 1789. [2] If a country has had the same political system for a long time and you disapprove of it, you can refer to it as **the ancien regime**. N-SING: usu *the* N

N-SING: usu *the* N disapproval

an|cient /eɪnʃənt/ [1] **Ancient** means belonging to the distant past, especially to the period in history before the end of the Roman Empire. ❑ *They believed ancient Greece and Rome were vital sources of learning.* ♦ **an|cient|ly** *Salisbury Plain was known anciently as Ellendune.* [2] **Ancient** ◆◇◇ ADJ: ADJ n

ADV

ADJ: usu ADJ n

Stable sorting

Lexicographic order

Radix sort

a sorting summary

and a glimpse of Timsort

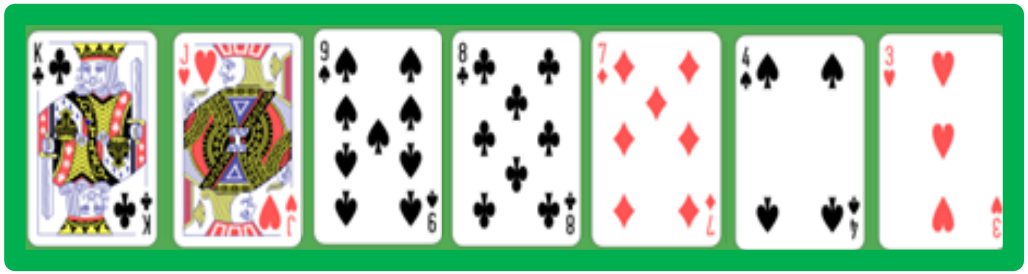
Party or Tech Group	Aontú	Fianna Fáil	Fine Gael	Green	Independent	Labour	Right to Change	Sinn Féin	Social Democrats	Solidarity PBP
TDs	1	37	35	12	19	6	1	37	6	5

Party or Tech Group	Sinn Féin	Fianna Fáil	Fine Gael	Independent	Green	Social Democrats	Labour	Solidarity PBP	Aontú	Right to Change
TDs	37	37	35	19	12	6	6	5	1	1

Party or Tech Group	Fianna Fáil	Sinn Féin	Fine Gael	Independent	Green	Labour	Social Democrats	Solidarity PBP	Aontú	Right to Change
TDs	37	37	35	19	12	6	6	5	1	1

Stable

A *stable* sorting algorithm is one that keeps the original order of any pair of items that have equivalent keys.



KC JH 9S 8C 7D 4S 3H

A stable sorting algorithm, now sorting by suit in order S<H<C<D will give

9S 4S JH 3H KC 8C 7D

Let x0, x1, x2, ..., xn be the items in the list in original order.
Let the key for item xi be xi.k, and now re-sort the list by the key.
Let p(xi) be the position of xi in the resorted list.

The sorting algorithm is *stable* if and only if
for any pair (i,j) where xi.k == xj.k and i<j, then p(xi) < p(xj)

Which of our previous algorithms (as written) are stable?

Bubblesort	yes
Selectionsort	no
Insertionsort	yes
Heapsort	no
Mergesort	yes
Quicksort	no
Countingsort	yes, version 2

Lexicographic order

For two strings of characters, add a sequence of 'a' characters to the end of the shorter one, so that they are the same length.

string s is *lexicographically before* t if and only if
s[0] < t[0], or
s[0] == t[0] and s[1] < t[1], or
s[0] = t[0], s[1] == t[1], and s[2] < t[2], or
...

string s is *lexicographically before* t
 if and only if
$\exists\ i \geq 0$ s.t. $\forall j < i$ s[j] == t[j] and s[i] < t[i]

Exercise: write an algorithm to sort a list of tuples in lexicographic order, but where we only compare one element of each tuple on each pass?

[(4,1), (5,7), (3,2), (7,2), (6,4), (9,3), (9,2), (7,1), (5,4)]

Answer?

Choose any stable sorting algorithm.

1. Sort by 2nd elt
2. Re-sort by 1st elt

Applying this technique using Counting Sort(v2) is called *Radix* sort.

[(4,1), (5,7), (3,2), (7,2), (6,4), (9,3), (9,2), (7,1), (5,4), (4,5)]

Sort the list using Counting Sort on the 2nd element

[(4,1), (7,1), (3,2), (7,2), (9,2), (9,3), (6,4), (5,4), (4,5), (5,7)]

Now sort the list again using Counting sort, but now on the 1st element

[(3,2), (4,1), (4,5), (5,4), (5,7), (6,4), (7,1), (7,2), (9,2), (9,3)]

Analysis: there are possibly k=9 digits, and the length of the list is n=10

O(n+k) for the first sort

O(n+k) for the second sort

We can also apply Radix sort to sorting integers

[41, 57, 32, 72, 64, 93, 92, 71, 54, 45]

Max integer in list has 2 digits, so any values with fewer than 2 digits get 0s added to the front.
Sort the list using Counting sort on the 2nd digit

[41, 71, 32, 72, 92, 93, 64, 54, 45, 57]

Now sort on 1st digit

[32, 41, 45, 54, 57, 64, 71, 92, 93]

Analysis: as before, k=9 and n = 10
O(n+k) for the first sort
O(n+k) for the second sort

A single counting sort would have had complexity O(n+k) where k was 100 (or k=61 if we know the range is 32—93)

Exercise: what algorithm would be best to sort 1000 integers with values from 0 to 99999?

Sorting: a summary

Bubblesort

Selectionsort

Insertionsort

Heapsort

Mergesort

Quicksort

Counting

Summary: Bubblesort

Comparison sort

Simple to write, easy to understand, stable, and in-place.

But worst case time complexity is $O(n^2)$, and the simple implementation always does the worst case.

In practice, it is slow (slower than other $O(n^2)$ worst case algorithms), and is almost never used.

Summary: Selectionsort

Comparison sort.

Reasonably simple to write, easy to understand, in-place.

Normal implementation is not stable.

Worst case time complexity is $O(n^2)$

In practice, it is slow and rarely used.

Summary: Insertionsort

Comparison sort.

Reasonably simple to write, easy to understand, in-place and stable.

Worst case time complexity is $O(n^2)$

It performs well on input that is already close to being sorted.

Can be easily adapted to sort an online stream of incoming data.

Usually better than the other $O(n^2)$ algorithms, is reasonably fast on smaller inputs, and is often used inside recursive algorithms when the input list gets below a certain size

Summary: Heapsort

Comparison sort.

Complex to write, requiring a heap data structure

Can be written as an in-place algorithm.

It is not stable.

Worst case time complexity is O(n log n)

Generally slower in practice than other O(n log n) algorithms

Summary: Mergesort

Comparison sort.

Reasonably easy to write for linked lists, but more complex for a bottom-up array sort.

Difficult to write as in-place

Most implementations are stable.

Worst case time complexity is O(n log n)

Summary: Quicksort

Comparison sort.

Basic idea is reasonably simple, but need to be careful when implementing

Can be written as in-place

Not stable.

Worst case time complexity is $O(n^2)$,
but average complexity is O(n log n), and is normally faster than the worst case O(n log n) algorithms.

For a long time, thought to be the best practical algorithm.

Summary: Counting sort

Not a comparison sort.

Simple to understand and to implement

Normally implemented to be stable.

Worst case time complexity is O(n+k), where k is the range of the values to be sorted.

Requires O(k) additional space.

Summary: Radix sort

Not a comparison sort.

Essentially multiple iterations of counting sort on different keys.
More varieties than discussed here.

Stable.

Worst case time complexity is not clear. Based on number of items in the key words, and number of items.

Not in-place

Summary: Quicksort

Comparison sort.

Basic idea is reasonably simple, but need to be careful when implementing

Can be written as in-place

Not stable.

Worst case time complexity is $O(n^2)$,
but average complexity is O(n log n), and is normally faster than the worst case O(n log n) algorithms.

For a long time, thought to be the best practical algorithm.

?

Timsort

Specifically designed for Python in 2002 , by Tim Peters.

It is a hybrid of mergesort and insertion sort, designed to perform well when the data is already partially sorted. It is stable.

Timsort:

- finds 'runs' – sequences of items already in sort order
- merges runs together
- runs of short length are combined with larger ones using insertion sort

but is too complex and too detailed to present in this module.

Its worst case complexity is O(*n log n*), its best case is O(*n*), and it is believed to be faster than Quicksort in most cases.
Now also the default sorting algorithm in Java, and in Android.

The original description to the Python community is here:
https://svn.python.org/projects/python/trunk/Objects/listsort.txt

Sorting: final words

It is important that you understand how the different sorting algorithms work, what their time complexities are, space complexities, and whether or not they are stable.

Sorting is a fundamental skill in computer science and software development, and forms the building blocks for many more complex operations.

But: when you are developing software and programming for real products in well-designed and well-supported languages, you should normally use their built-in sort routines

- they will have been optimised for the language design

Next lecture

Selecting items from lists