From YouTube series:

Python CP Lesson 2: String Manipulation
Page 1

Python CP Lesson 4: Sort & Search Algorithms
Page 2 and 3

Caesar Cipher Encrypter: use this to encrypt the string.

Caesar Cipher is an encryption technique that involves shifting letters in the alphabet left/right

The sender encodes the message by shifting the letters three places left

Exp: Fed -> Cba

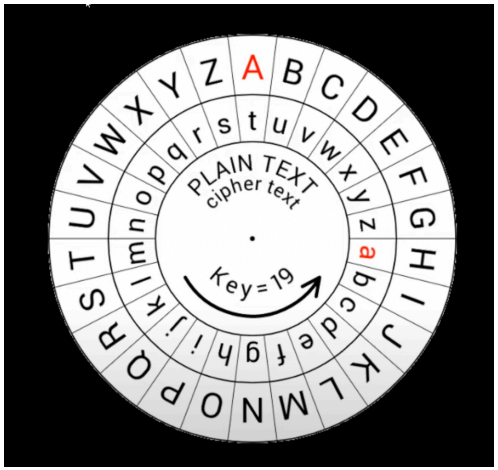
The receiver decodes the message by shifting the letters three places right

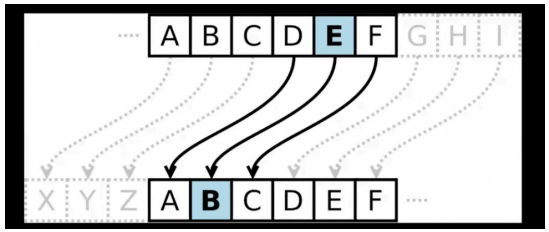
Exp: Cba -> Fed

A good example:

- Decode this secret message:
  jung qbrf gur sbk fnl !

• In order to solve this problem we can shift the message by 13 letters.
• This is known as ROT13.
• The message becomes:
What does the fox say.



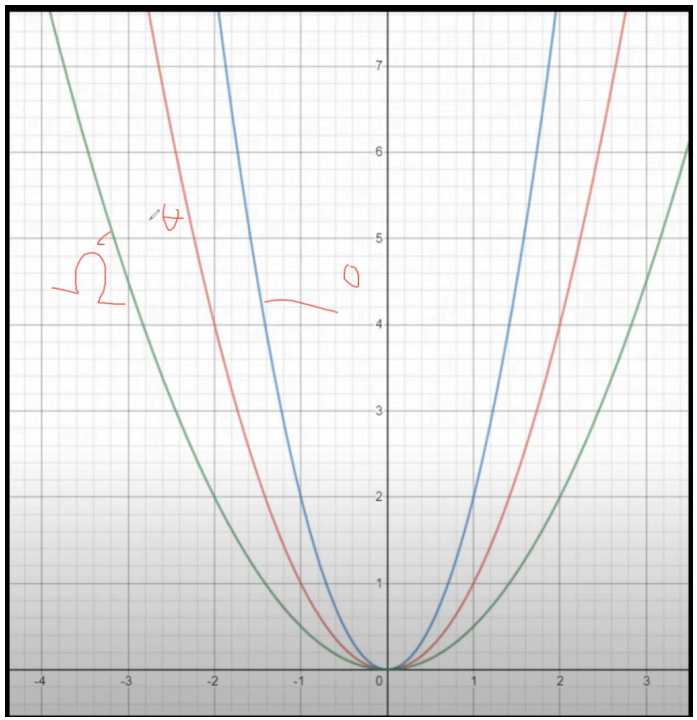


Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	@	@	96	60	140	`	`
65	41	101	A	A	97	61	141	a	a
66	42	102	B	B	98	62	142	b	b
67	43	103	C	C	99	63	143	c	c
68	44	104	D	D	100	64	144	d	d
69	45	105	E	E	101	65	145	e	e
70	46	106	F	F	102	66	146	f	f
71	47	107	G	G	103	67	147	g	g
72	48	110	H	H	104	68	150	h	h
73	49	111	I	I	105	69	151	i	i
74	4A	112	J	J	106	6A	152	j	j
75	4B	113	K	K	107	6B	153	k	k
76	4C	114	L	L	108	6C	154	l	l
77	4D	115	M	M	109	6D	155	m	m
78	4E	116	N	N	110	6E	156	n	n
79	4F	117	O	O	111	6F	157	o	o
80	50	120	P	P	112	70	160	p	p
81	51	121	Q	Q	113	71	161	q	q
82	52	122	R	R	114	72	162	r	r
83	53	123	S	S	115	73	163	s	s
84	54	124	T	T	116	74	164	t	t
85	55	125	U	U	117	75	165	u	u
86	56	126	V	V	118	76	166	v	v
87	57	127	W	W	119	77	167	w	w
88	58	130	X	X	120	78	170	x	x
89	59	131	Y	Y	121	79	171	y	y
90	5A	132	Z	Z	122	7A	172	z	z

From YouTube series:

Python CP Lesson 2: String Manipulation Page 1

Python CP Lesson 4: Sort & Search Algorithms Page 2 and 3



About OOP:

The concepts of oops in python focuses on creating reusable code. This concept is also known as DPY - don't repeat yourself

"If we know Big O, we also know 'it can't get worst than this'"

If we want to use math (calculus) to understand this, the formula is:

f(x) <= g(x) <= h(x)

K = lim f(x) <= lim g(x) <= lim h(x) = K

||

K

Basic polynomial asymptotic growth:

- the polynomial asymptotic rule is:

O(ax^n) = O(x^n)

O(P(x)) = O(a_n * x ^ n + a_n - 1 * x ^ n - 1 …) = O(x ^ n)

In short we take the highest degree term.

*quick note: we can use l'Hopital rule (calculus) and degree approximations.

Asymptotic growth of searching:

From YouTube series:

Python CP Lesson 2: String Manipulation
Page 1

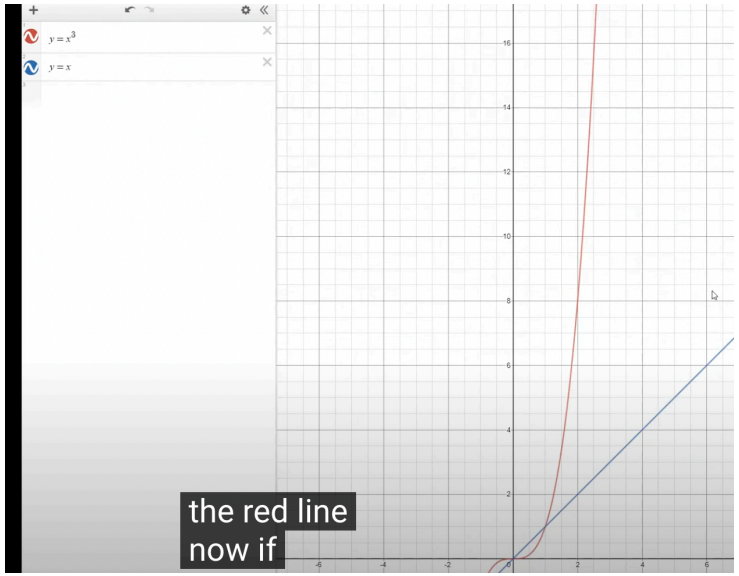
Python CP Lesson 4: Sort & Search Algorithms
Page 2 and 3

Looking at linear search, the worst case to find an element is O(N) because we have to search for every element.

The best case is O(1), since we find the answer immediately

The average case is O((1 + N) / 2) = O(N / 2 + 1 / 2) => O(N)

But if we look at the binary search, the best case is also O(1), but the worse case is actually log2(N)



Time complexity:

logN = K

We can induct on N and make it smaller until it hits the base case of 1, which we are done. Then we count the steps.

To measure an algorithm, we also induct on each step until we get our base case.

In computer science, **amortized analysis** is a method for analyzing a given algorithm's complexity, or how much of a resource, especially time or memory, it takes to execute. The motivation for amortized analysis is that looking at the worst-case run time can be too pessimistic.

"The amortised analyses it is not only based on the build analyses"

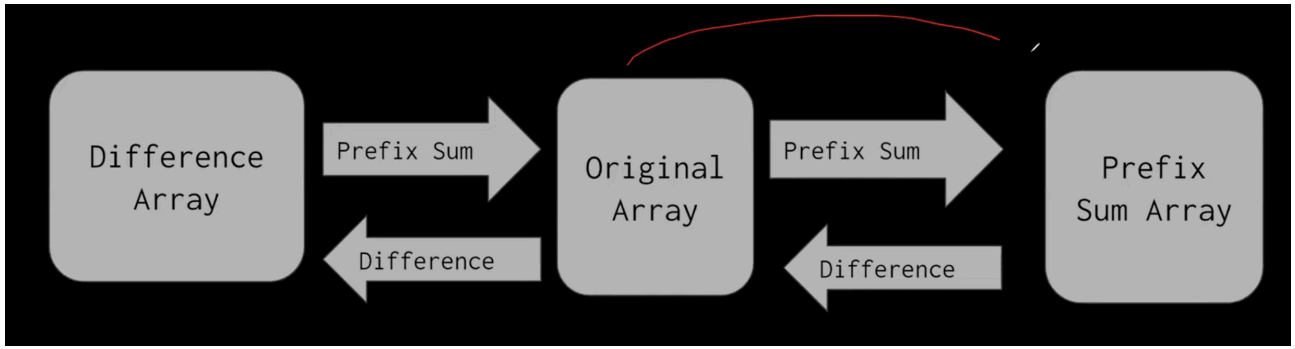
Asymptotic growth analysis

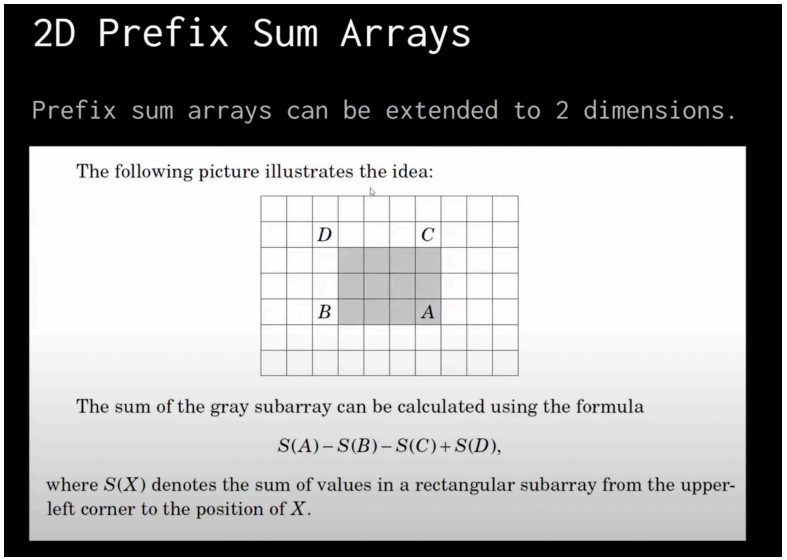
With a naive algorithm, you have to check n elements at worst case each time, taking O(N) time. So with Q queries the final complexity is O(Q*N) = O(QN)

From YouTube series:

Python CP Lesson 2: String Manipulation
Page 1

Python CP Lesson 4: Sort & Search Algorithms
Page 2 and 3





A note on Algorithms

- Every subtask requires its own crafted algorithm to solve it.
- This means that an algorithm that can solve one subtask may not be able to solve the other one.
- The algorithm that you use is very dependent on the bounds of the subtask it is required by

Subtask:

- when you do the problem, many of them will be split into “subtasks”.
- we should first analyse the complexity of each subtask to understand how to solve the question.
- for first and second subtask we can use a naive approach.
- for the next subtask we need a better algorithm, meaning we can use a prefix sum array to solve the problem.
- to use multiple algorithms just use if statements depending on the bounds given.

Online and Offline queries:

- Online query: a query that must be solved by processing the queries one at a time as they arrive.
- Offline query: query that must be solved by preprocessing and solving the queries all at once.