|  |  |  |
| --- | --- | --- |
| Power | Val | Cs speak |
| 0 | 1 |  |
| 1 | 2 |  |
| 2 | 4 |  |
| 3 | 8 |  |
| 4 | 16 |  |
| 5 | 32 |  |
| 6 | 64 |  |
| 7 | 128 |  |
| 8 | 256 |  |
| 9 | 512 |  |
| 10 | 1k | Kilo |
| 20 | 1m | Mega |
| 30 | 1b | Giga |
| 40 | 1t | Tera |
| 50 | 1q | peta |
| 60 | 1z | Exabyte (google had 10 in 2013) |

= num ways to pick k from n (**no order**)

= choose k from n **with order**

n! = permutations

**BAYES RULE:**

½ + ¼ + … = 1

sigma(x) = sigma’(x) = s(x) ( 1 – s(x) )

(on when y=1)y (on when y=0)1-y

**# nodes** **in tree** with depth D and branching factor b

1 + b + b2 + … + bD

= bD+1-1/b-1

**Processes** have their own address space

**Threads** are in a process, share heap but separate stacks

Sorted() is **ASCENDING**

Numerator **OR** denominator can be float

**HEAPS**

Parent = (I – 2) / 2

i

left = i\*2 right = i\*2+1

import heapq

heapq.heappush(list, item)

heapq.heappop(list, item)

heapq.heapify(list)

heapq.pushpop(list, item)

**SEARCH**

import bisect

bisect.bisect\_left(a, x, low=0, high=len(a))

index of leftmost val >= x

**QUEUES**

From collections import deque

D =deque([1, 2, 3])

d.appendleft(x)

d.extendleft([…])

d.pop()

**REGEX**

|  |  |
| --- | --- |
| \d | [0-9] |
| \D | [^0-9] |
| \s | [whitespaces] |
| \S | [^whitespace] |
| \w | Alphanumeric |
| \W | [^alphanumeric] |
| \* | Repeat 0+ |
| + | Repeat 1+ |
| {m, n}  {m,}  {,n} | n-n times |
| ? | 0 or 1 |
| | | Or |
| ^…$ | Start and end of line |
| \b | Word boundary |
| (…) | group |

Import re

Re.search(pattern, string)

Tmp = re.compile(pattern, string)

Tmp.search(string)

Re.findall(pattern, string)

**(Returns STRINGS, not re.match)**

Re.finditer(pattern, string)

Re.sub(pattern, string, replace)

Match.group(group #)

Match.start()

Match.end()

**TREE TRAVERSAL**

Tree depth **D**

Soln is at depth **d**

Branching factor **b**

|  |  |  |
| --- | --- | --- |
|  | Space | Time |
| DFS | O(D) | O(bD) |
| BFS | bd | bd |
| DFS + iterative deepening | O(d) | O(bd) |