Thinking & Coding Algorithms

Kyle Simpson

Recommended Reading

Disclaimer #1

Lab, Not Lecture

Disclaimer #2

Using JS 🤔

TL;DW

1. Ask better clarifying questions
2. Balance CPU vs Memory
3. Shape data structure(s) to the problem, not the other way
4. “premature optimization” vs. “immature optimization”

Quick DSA Primer

Common Data Structures

Array, Stack, Queue

Set, Object, Map

Tree, Graph

Common Algorithms

BubbleSort, QuickSort

Tree Traversal, Path Finding

Binary Search

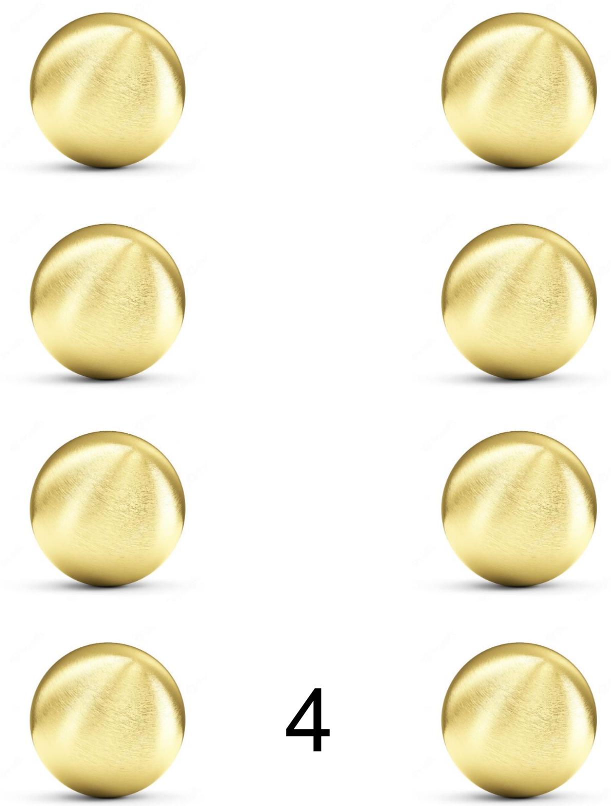
Common Techniques

Iteration

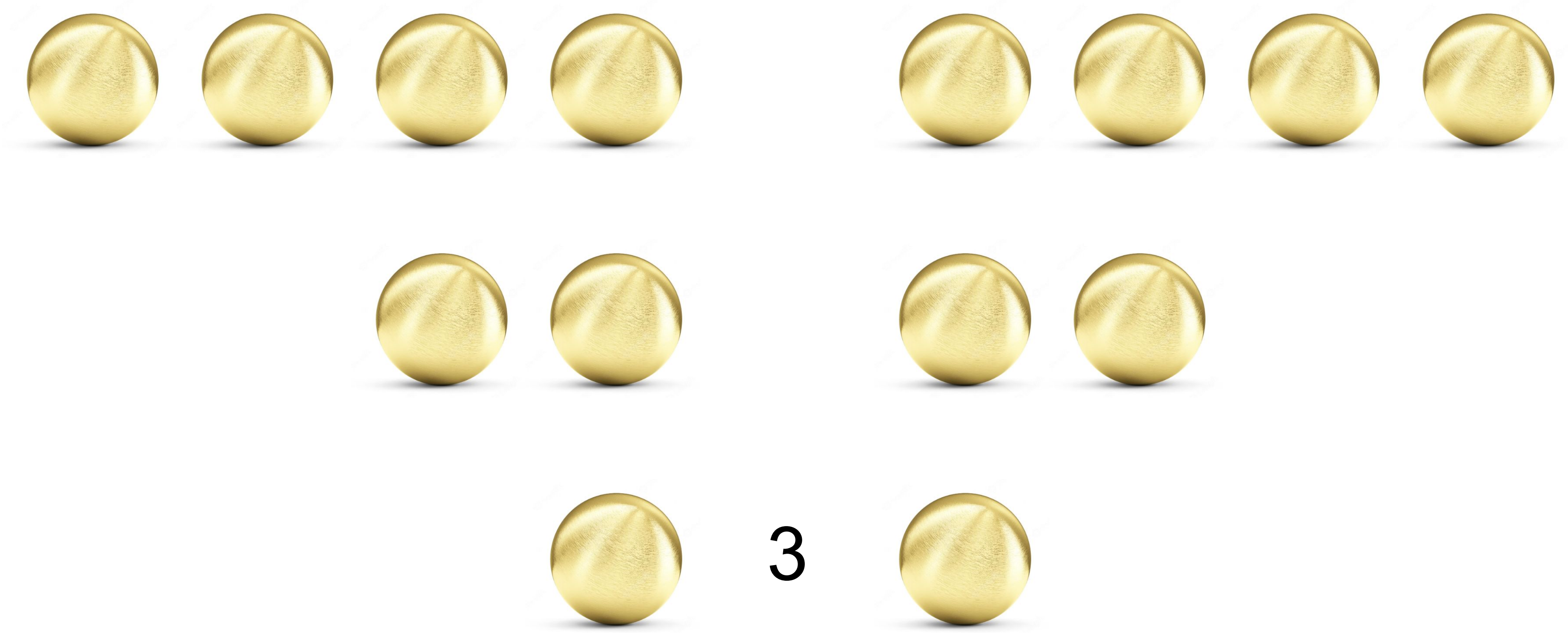
Recursion

Indexes, References

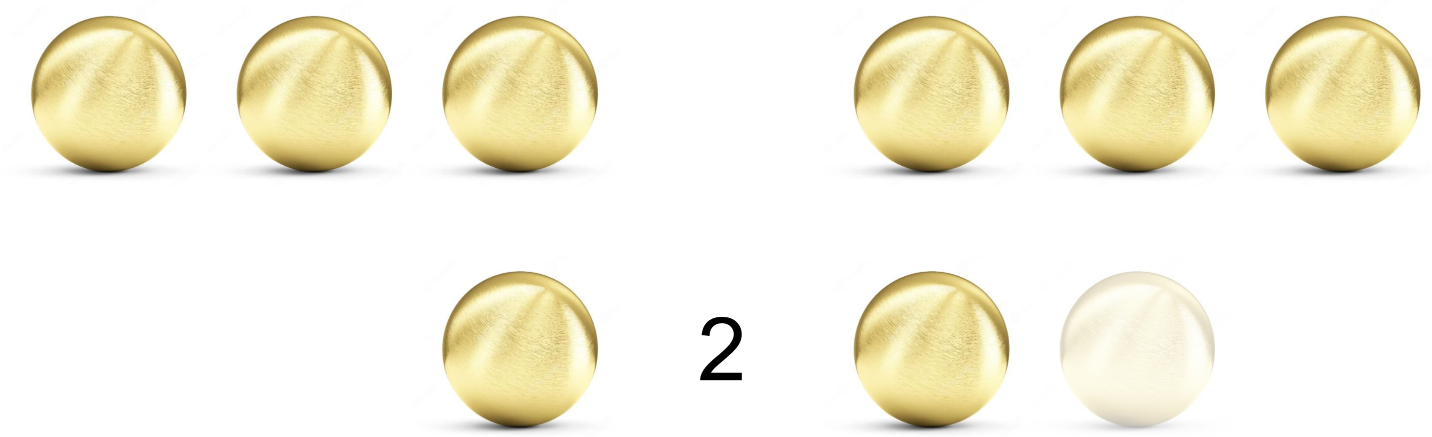
Iteration



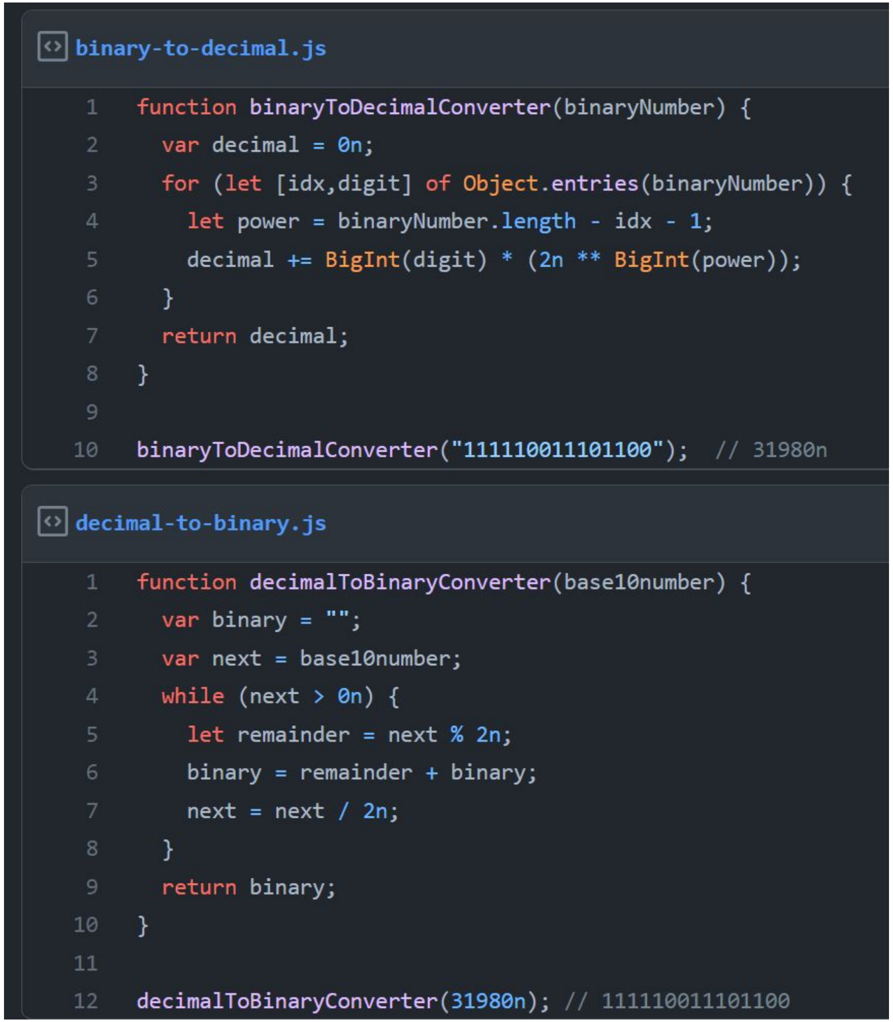
Recursion



Fuzzy?



Warm-ups

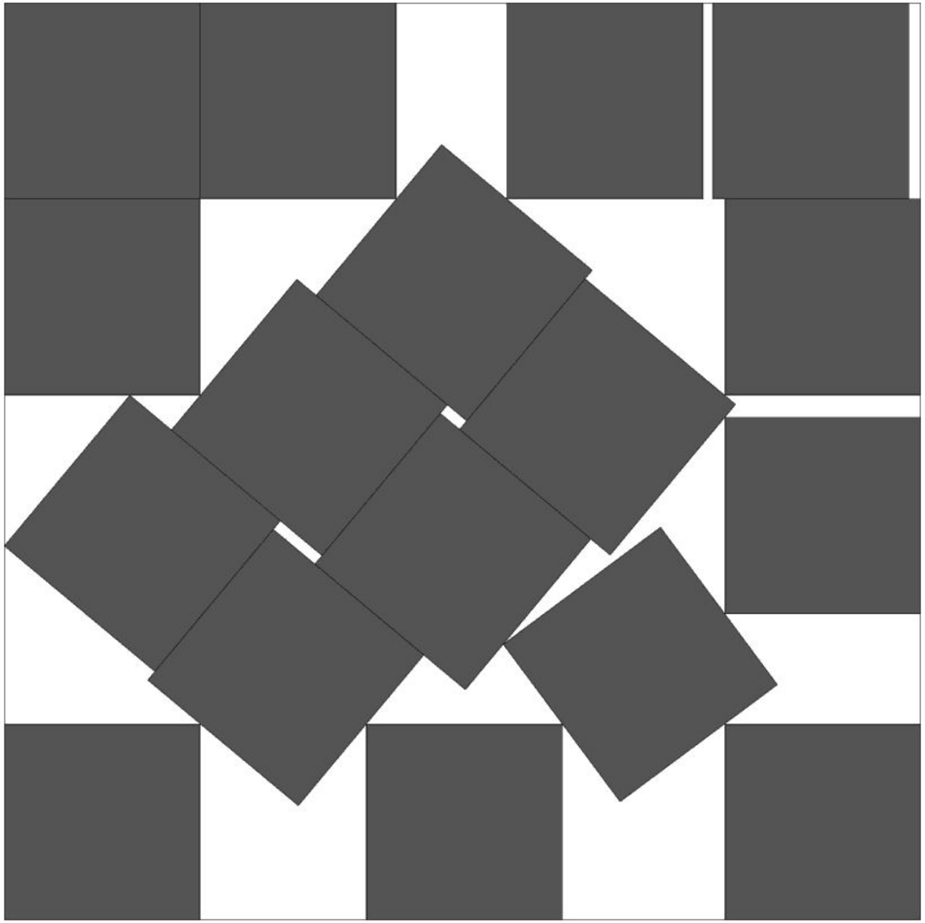


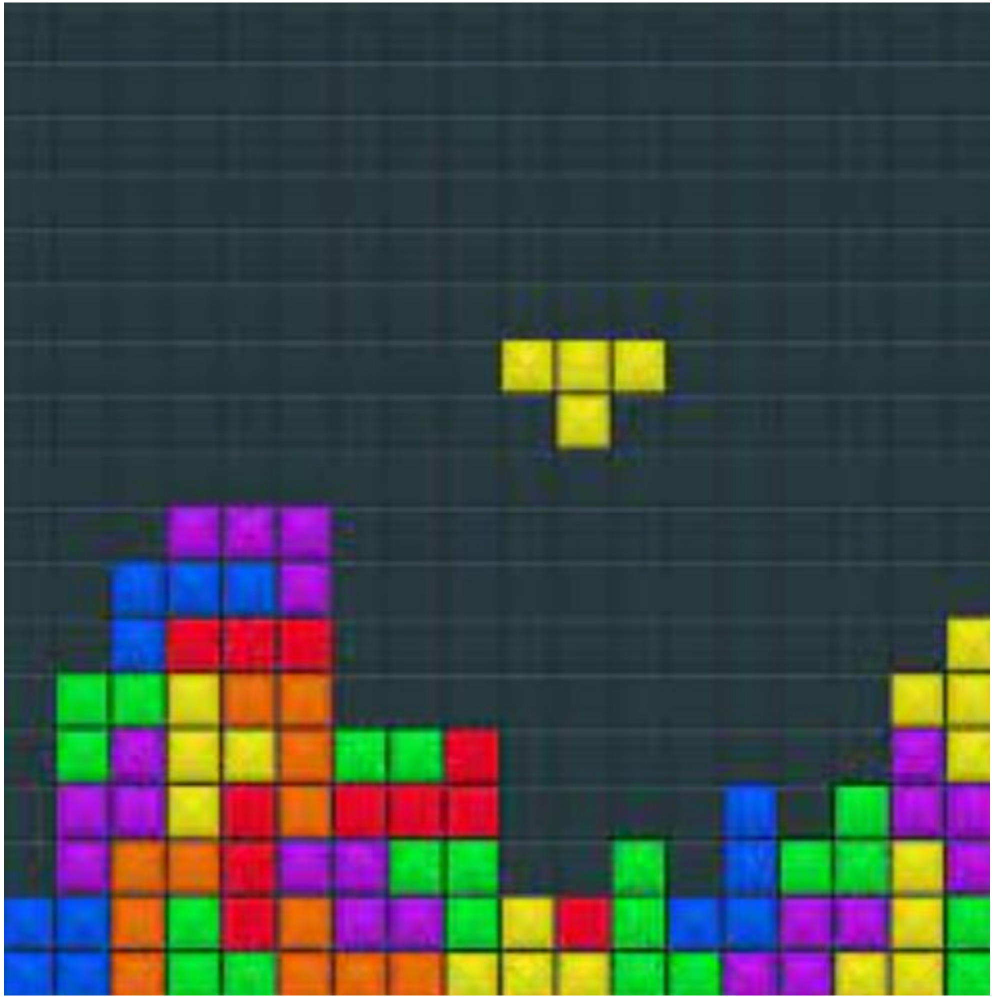
binary-to-decimal.js

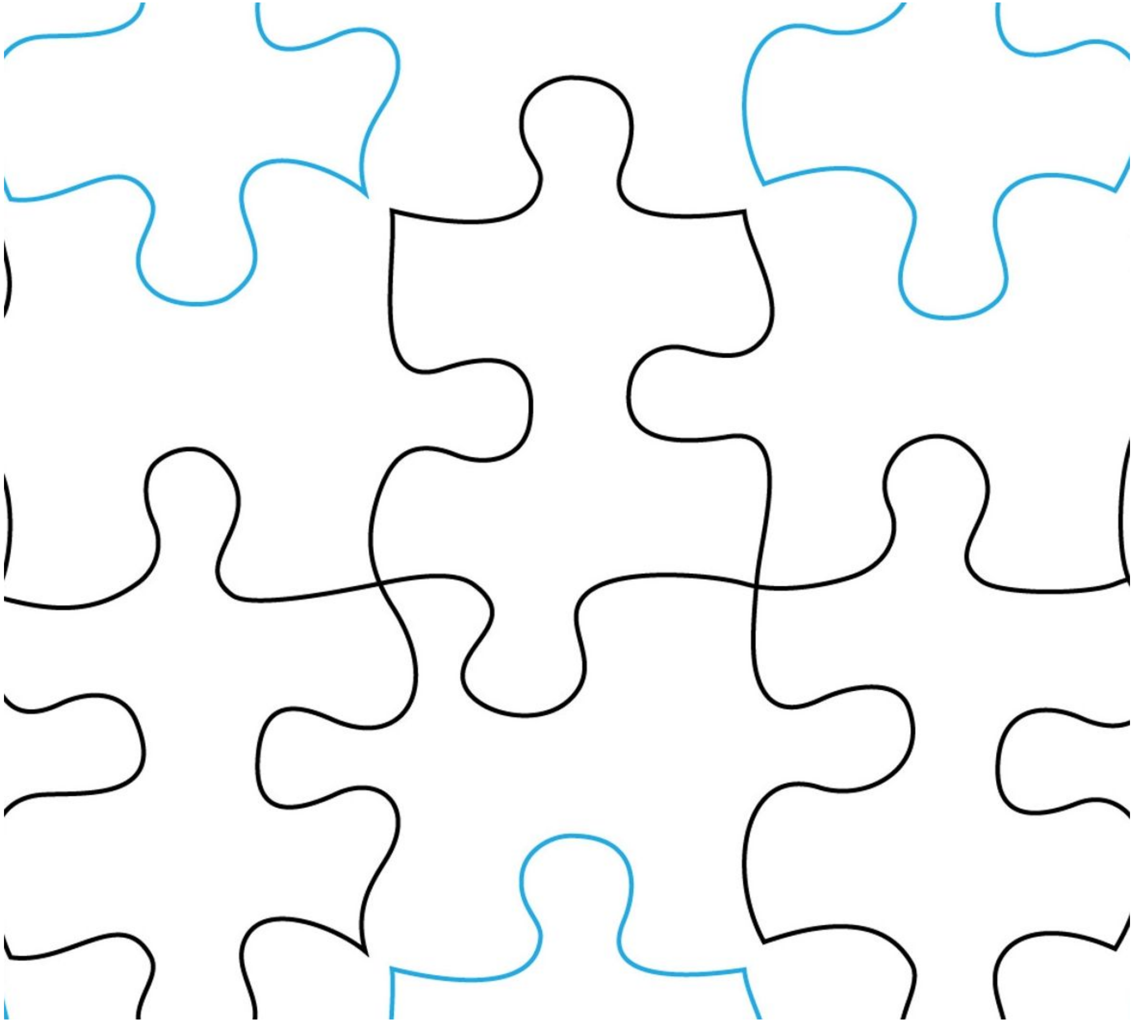
```
function binaryToDecimalConverter(binaryNumber) {
  var decimal = 0n;
  for (let [idx,digit] of Object.entries(binaryNumber)) {
    let power = binaryNumber.length - idx - 1;
    decimal += BigInt(digit) * (2n ** BigInt(power));
  }
  return decimal;
}

binaryToDecimalConverter("111110011101100");  // 31980n
```

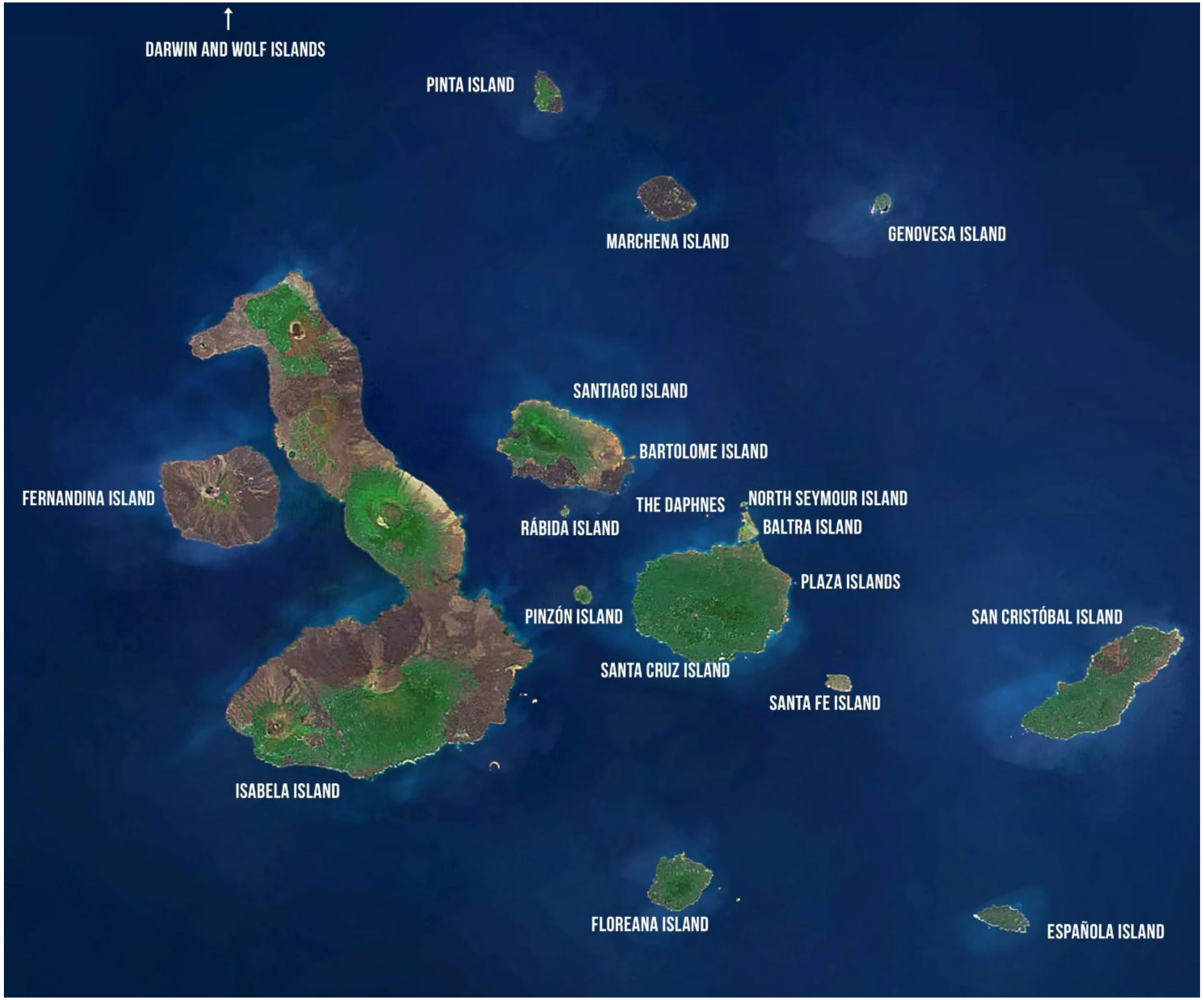
decimal-to-binary.js

```
function decimalToBinaryConverter(base10number) {
  var binary = "";
  var next = base10number;
  while (next > 0n) {
    let remainder = next % 2n;
    binary = remainder + binary;
    next = next / 2n;
  }
  return binary;
}

decimalToBinaryConverter(31980n); // 111110011101100
```

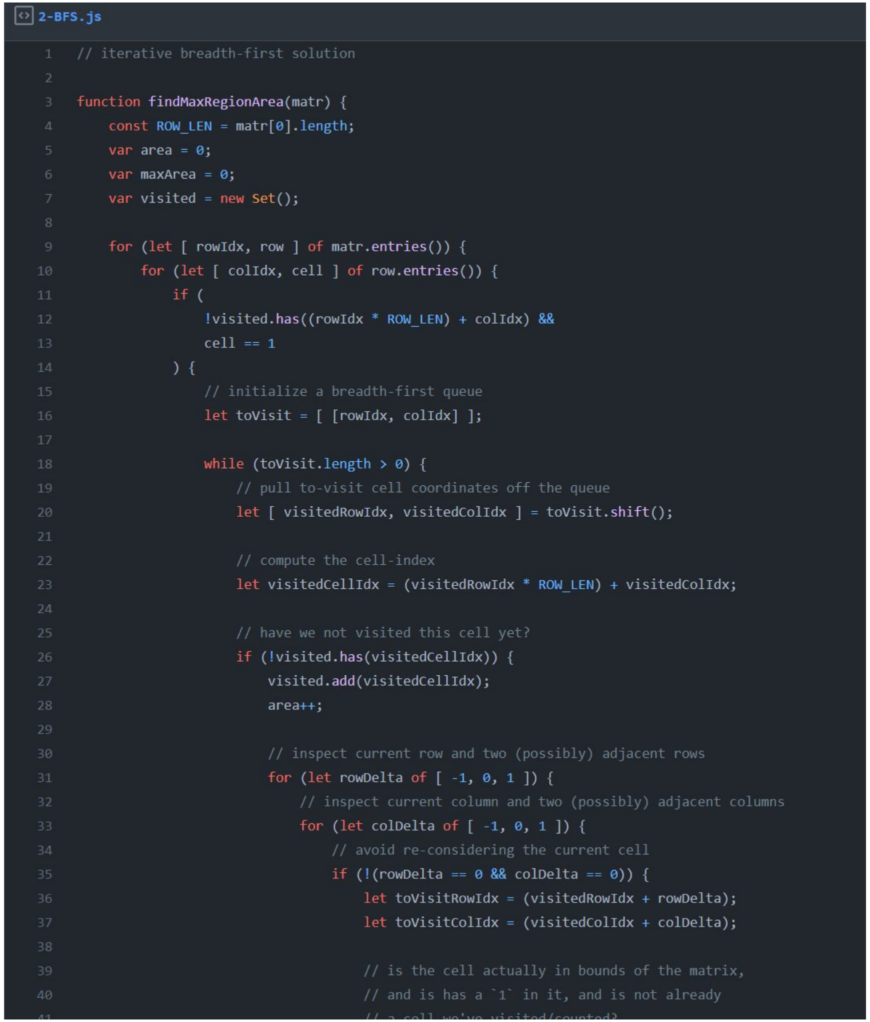








DARWIN AND WOLF ISLANDS
PINTA ISLAND
MARCHENA ISLAND
GENOVESA ISLAND
SANTIAGO ISLAND
BARTOLOME ISLAND
FERNANDINA ISLAND
THE DAPHNES
NORTH SEYMOUR ISLAND
RÁBIDA ISLAND
BALTRA ISLAND
PLAZA ISLANDS
PINZÓN ISLAND
SAN CRISTÓBAL ISLAND
SANTA CRUZ ISLAND
SANTA FE ISLAND
ISABELA ISLAND
FLOREANA ISLAND
ESPAÑOLA ISLAND



2-BFS.js

```javascript
// iterative breadth-first solution

function findMaxRegionArea(matr) {
    const ROW_LEN = matr[0].length;
    var area = 0;
    var maxArea = 0;
    var visited = new Set();

    for (let [ rowIdx, row ] of matr.entries()) {
        for (let [ colIdx, cell ] of row.entries()) {
            if (
                !visited.has((rowIdx * ROW_LEN) + colIdx) &&
                cell == 1
            ) {
                // initialize a breadth-first queue
                let toVisit = [ [rowIdx, colIdx] ];

                while (toVisit.length > 0) {
                    // pull to-visit cell coordinates off the queue
                    let [ visitedRowIdx, visitedColIdx ] = toVisit.shift();

                    // compute the cell-index
                    let visitedCellIdx = (visitedRowIdx * ROW_LEN) + visitedColIdx;

                    // have we not visited this cell yet?
                    if (!visited.has(visitedCellIdx)) {
                        visited.add(visitedCellIdx);
                        area++;

                        // inspect current row and two (possibly) adjacent rows
                        for (let rowDelta of [ -1, 0, 1 ]) {
                            // inspect current column and two (possibly) adjacent columns
                            for (let colDelta of [ -1, 0, 1 ]) {
                                // avoid re-considering the current cell
                                if (!(rowDelta == 0 && colDelta == 0)) {
                                    let toVisitRowIdx = (visitedRowIdx + rowDelta);
                                    let toVisitColIdx = (visitedColIdx + colDelta);

                                    // is the cell actually in bounds of the matrix,
                                    // and is has a `1` in it, and is not already
                                    // a cell we've visited/counted?
```

https://gist.github.com/getify/59ab7443723564eb40d20ab7c45d5f0a

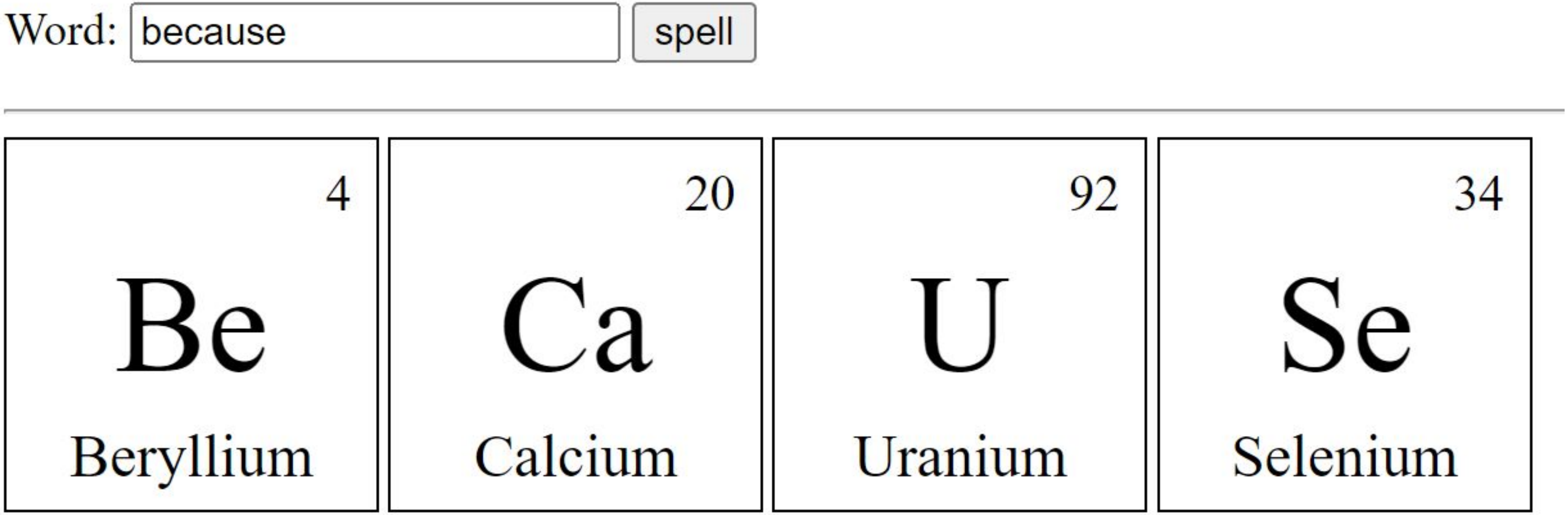
Exercise:
Periodic Table Speller

https://github.com/getify/workshop-periodic-table

Foxx
Life Sciences
Our Innovation. Your Opportunity.

PERIODIC TABLE OF ELEMENTS

1 I A	2 II A	3 III B	4 IV B	5 V B	6 VI B	7 VII B	8 VIII B	9 VIII B	10 VIII B	11 I B	12 II B	13 III A	14 IV A	15 V A	16 VI A	17 VII A	18 VIII A
1 H Hydrogen 1.00794																	2 He Helium 4.002602
3 Li Lithium 6.941	4 Be Beryllium 9.012182											5 B Boron 10.811	6 C Carbon 12.0107	7 N Nitrogen 14.0067	8 O Oxygen 15.9994	9 F Fluorine 18.9984032	10 Ne Neon 20.1797
11 Na Sodium 22.98976928	12 Mg Magnesium 24.305											13 Al Aluminium 26.9815386	14 Si Silicon 28.0855	15 P Phosphorus 30.973762	16 S Sulfur 32.065	17 Cl Chlorine 35.453	18 Ar Argon 39.948
19 K Potassium 39.0983	20 Ca Calcium 40.078	21 Sc Scandium 44.9559	22 Ti Titanium 47.867	23 V Vanadium 50.9415	24 Cr Chromium 51.9961	25 Mn Manganese 54.938045	26 Fe Iron 55.845	27 Co Cobalt 58.933195	28 Ni Nickel 58.6934	29 Cu Copper 63.546	30 Zn Zinc 65.38	31 Ga Gallium 69.729	32 Ge Germanium 72.64	33 As Arsenic 74.9216	34 Se Selenium 78.96	35 Br Bromine 79.904	36 Kr Krypton 83.798
37 Rb Rubidium 85.4678	38 Sr Strontium 87.62	39 Y Yttrium 88.90585	40 Zr Zirconium 91.224	41 Nb Niobium 92.9063	42 Mo Molybdenum 95.96	43 Tc Technetium [98]	44 Ru Ruthenium 101.07	45 Rh Rhodium 102.9055	46 Pd Palladium 106.42	47 Ag Silver 107.8682	48 Cd Cadmium 112.411	49 In Indium 114.818	50 Sn Tin 118.71	51 Sb Antimony 121.76	52 Te Tellurium 127.6	53 I Iodine 126.90447	54 Xe Xenon 131.293
55 Cs Caesium 132.9054519	56 Ba Barium 137.327	57-71 Lanthanoids	72 Hf Hafnium 178.49	73 Ta Tantalum 180.94788	74 W Tungsten 183.84	75 Re Rhenium 186.207	76 Os Osmium 190.23	77 Ir Iridium 192.217	78 Pt Platinum 195.084	79 Au Gold 196.966569	80 Hg Mercury 200.59	81 Tl Thallium 204.3833	82 Pb Lead 207.2	83 Bi Bismuth 208.9804	84 Po Polonium [209]	85 At Astatine [210]	86 Rn Radon [222]
87 Fr Francium [223]	88 Ra Radium [226]	89-103 Actinoids	104 Rf Rutherfordium [267]	105 Db Dubnium [268]	106 Sg Seaborgium [271]	107 Bh Bohrium [272]	108 Hs Hassium [270]	109 Mt Meitnerium [276]	110 Ds Darmstadtium [281]	111 Rg Roentgenium [280]	112 Cn Copernicium [285]	113 Nh Nihonium [286]	114 Fl Flerovium [289]	115 Mc Moscovium [288]	116 Lv Livermorium [293]	117 Ts Tennessine [294]	118 Og Oganesson [294]
		57 La Lanthanum 138.90547	58 Ce Cerium 140.116	59 Pr Praseodymium 140.90765	60 Nd Deodymium 144.242	61 Pm Promethium [145]	62 Sm Samarium 150.36	63 Eu Europium 151.964	64 Gd Gadolinium 157.25	65 Tb Terbium 158.9253	66 Dy Dysprosium 162.5	67 Ho Holmium 164.93032	68 Er Erbium 167.259	69 Tm Thulium 168.93421	70 Yb Ytterbium 173.054	71 Lu Lutetium 174.9668	
		89 Ac Actinium [227]	90 Th Thorium 232.03806	91 Pa Protactinium 231.03588	92 U Uranium 238.02891	93 Np Neptunium [237]	94 Pu Plutonium [244]	95 Am Americium [243]	96 Cm Curium [247]	97 Bk Berkelium [247]	98 Cf Californium [251]	99 Es Einsteinium [252]	100 Fm Fermium [257]	101 Md Mendelevium [258]	102 No Nobelium [262]	103 Lr Lawrencium [262]	



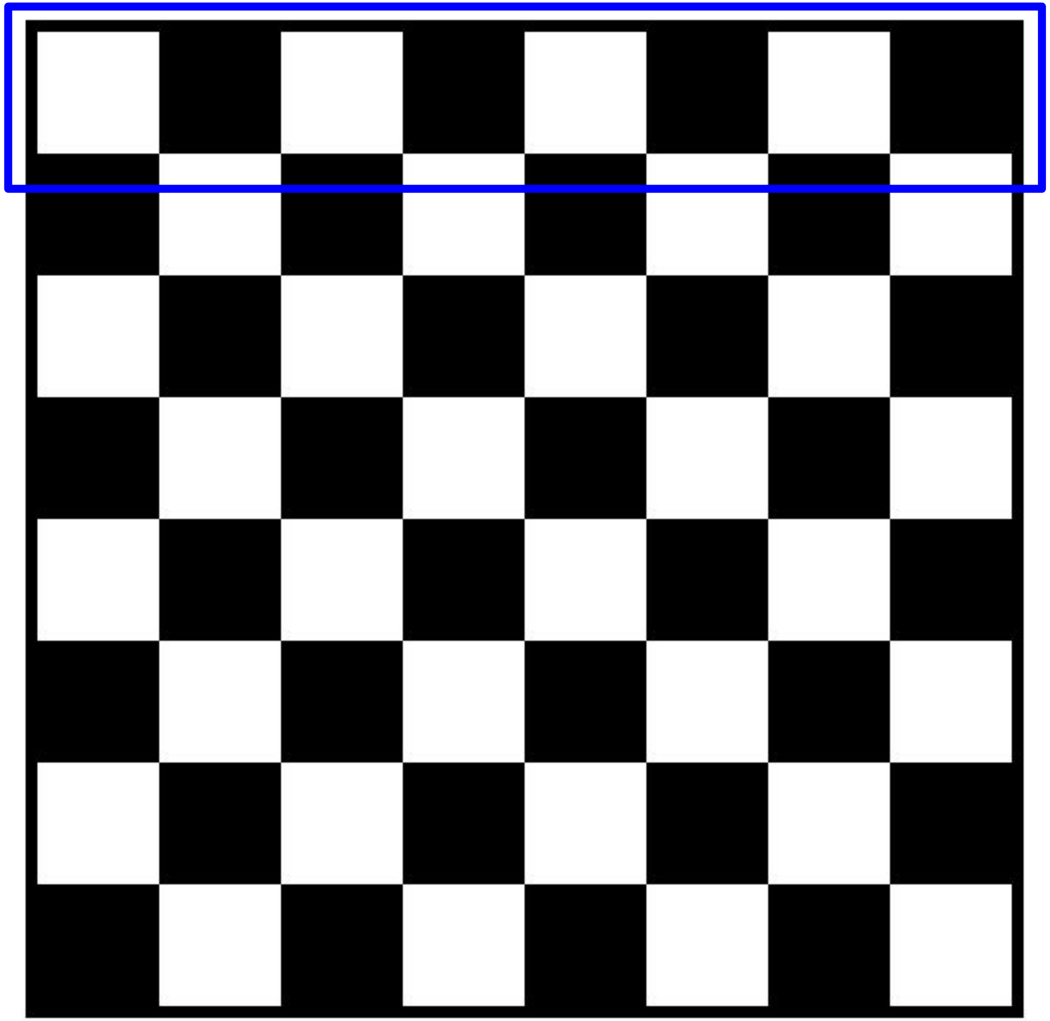
Word: because

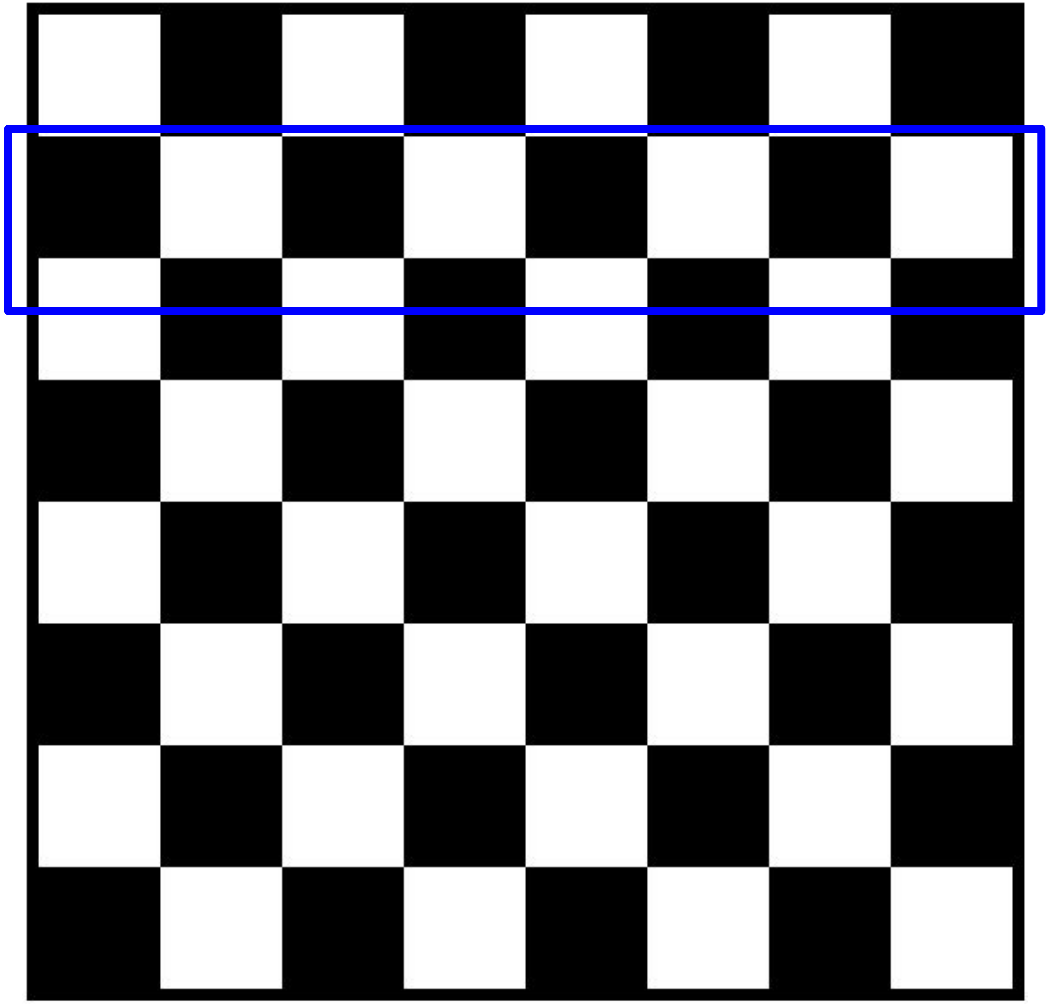
4	20	92	34
Be	Ca	U	Se
Beryllium	Calcium	Uranium	Selenium

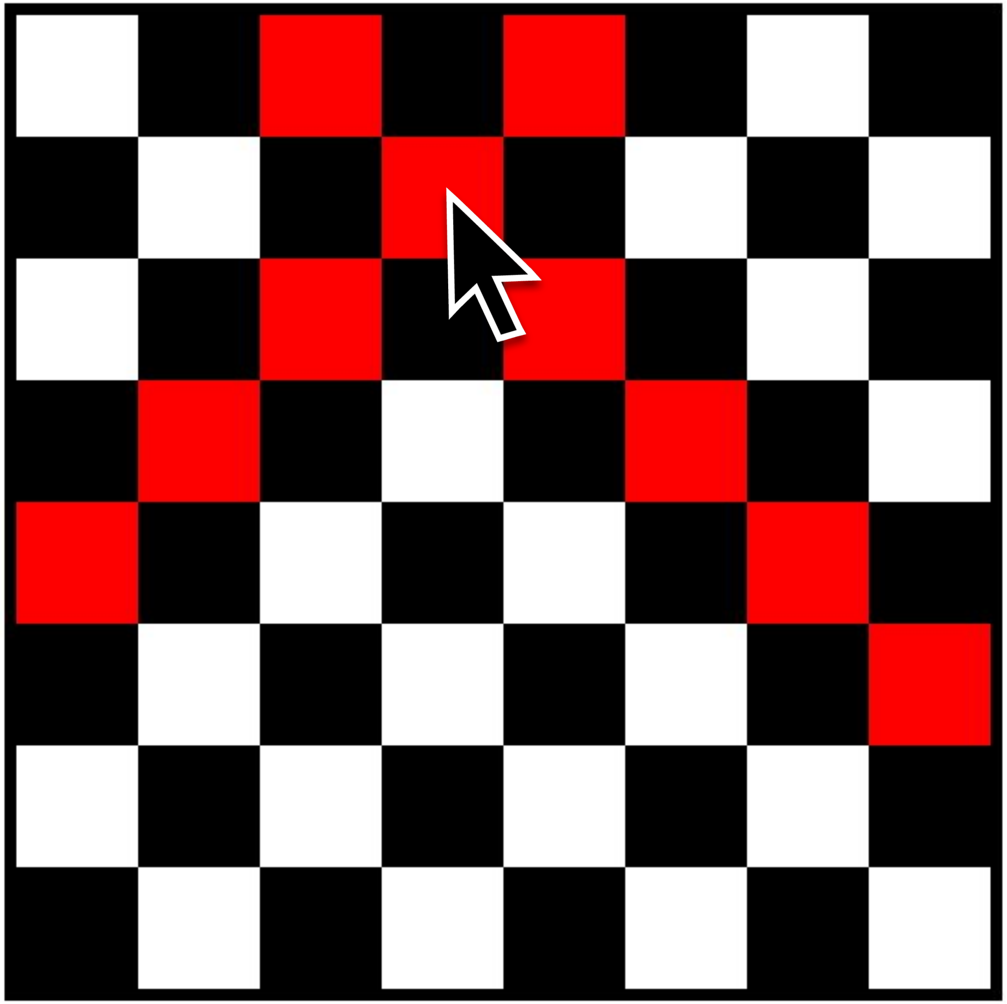
Periodic Table Speller

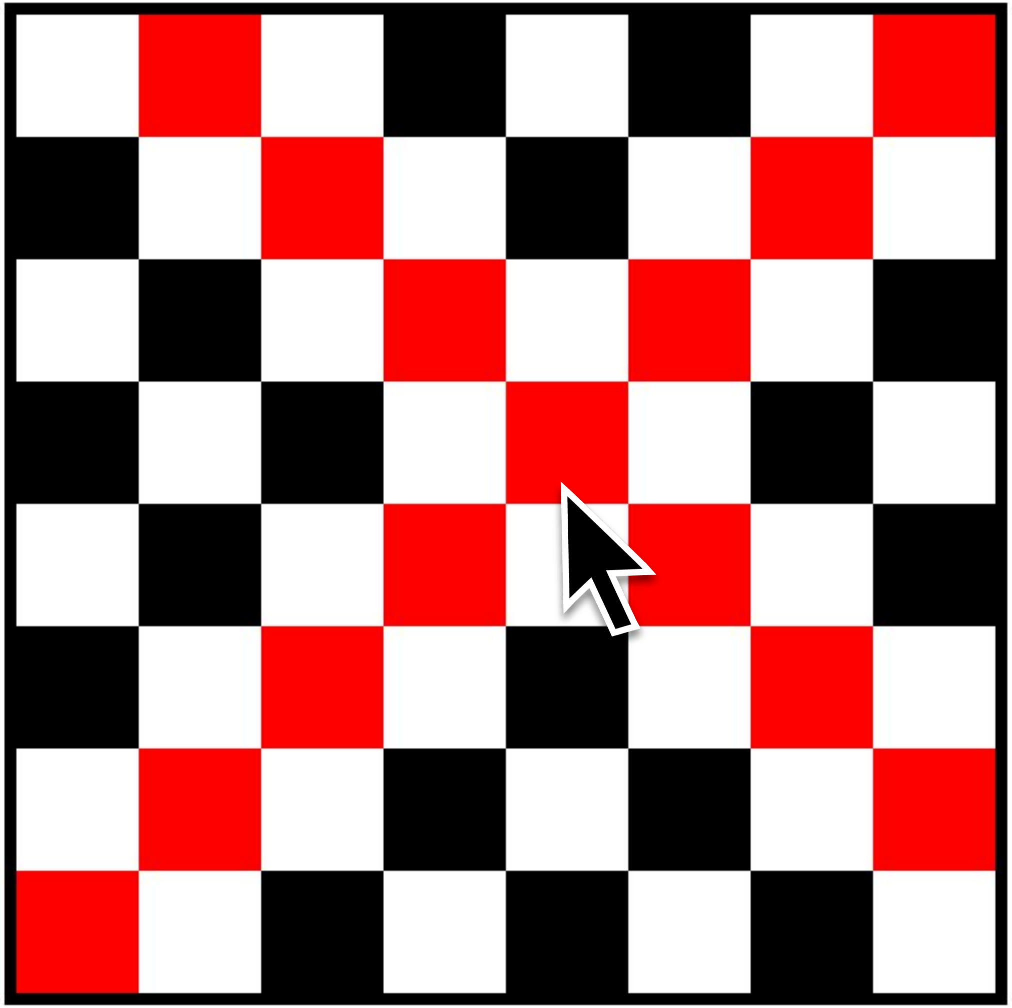
Exercise:
Chessboard Diagonals

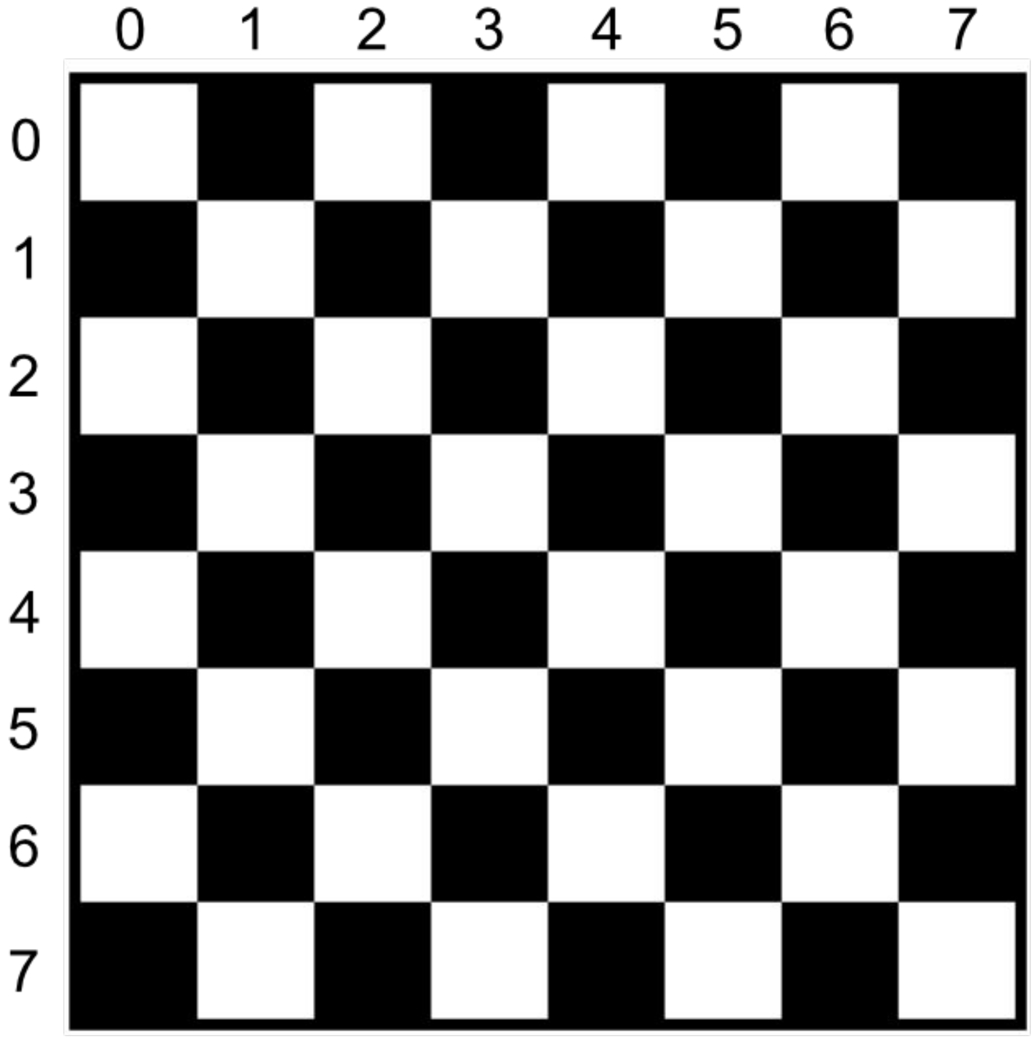
https://github.com/getify/workshop-chess-diagonals



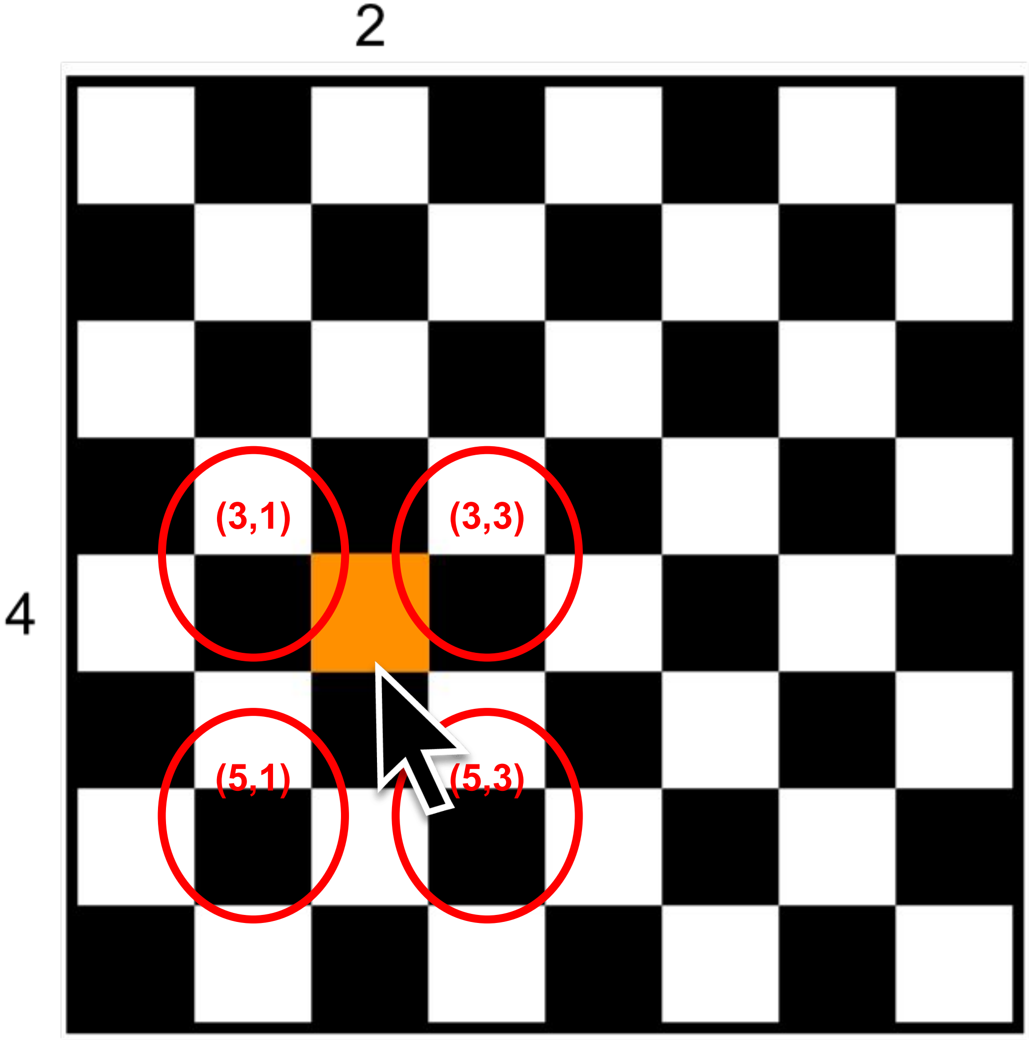




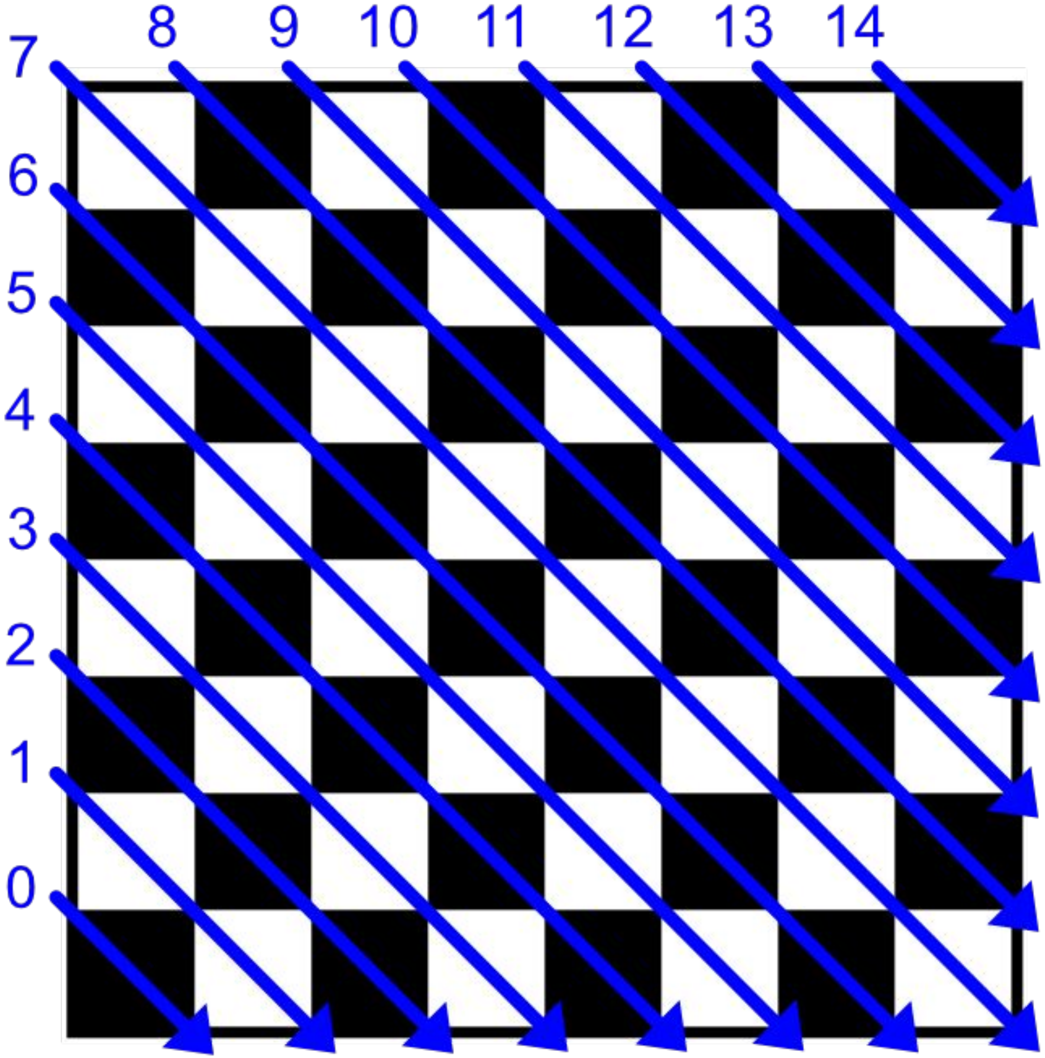



0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7

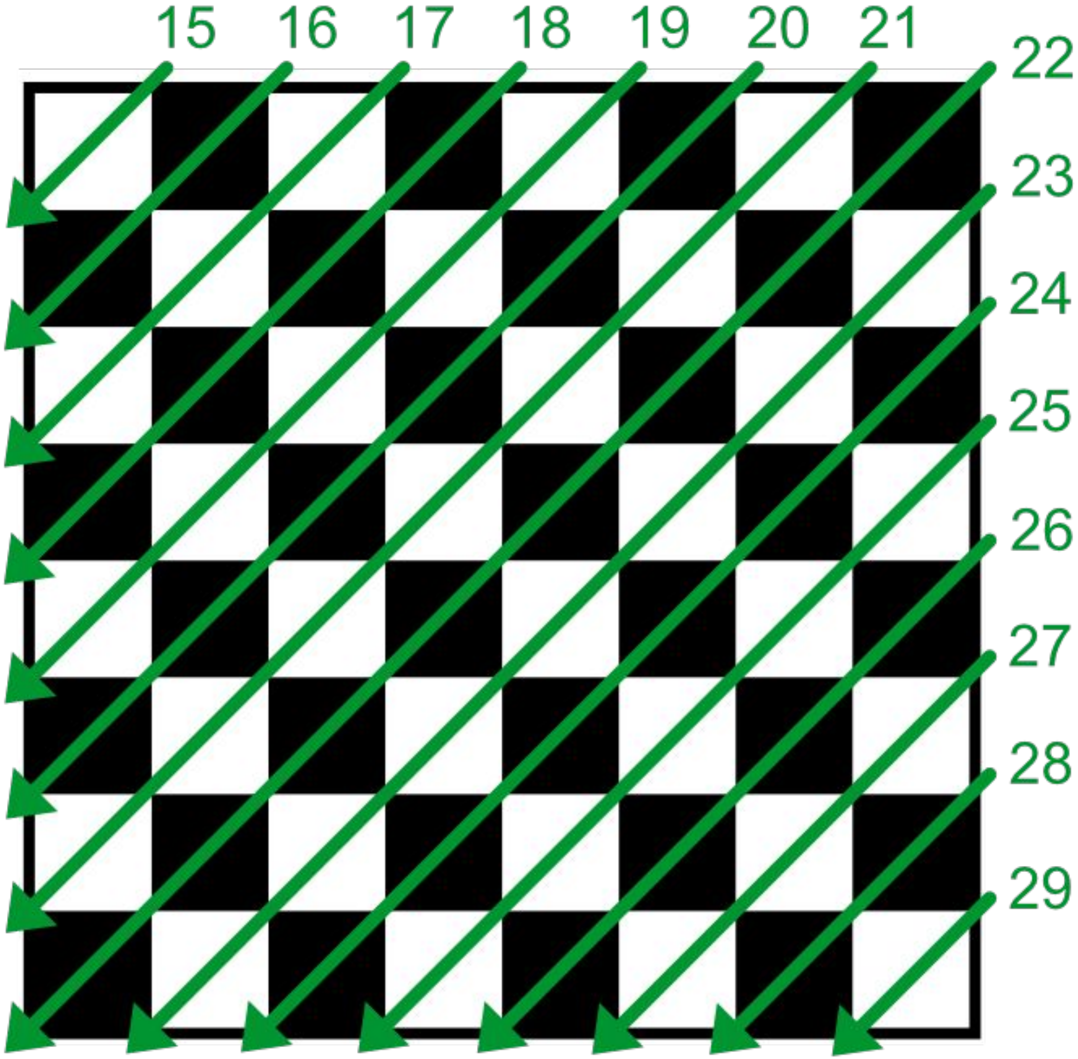
(4,2)

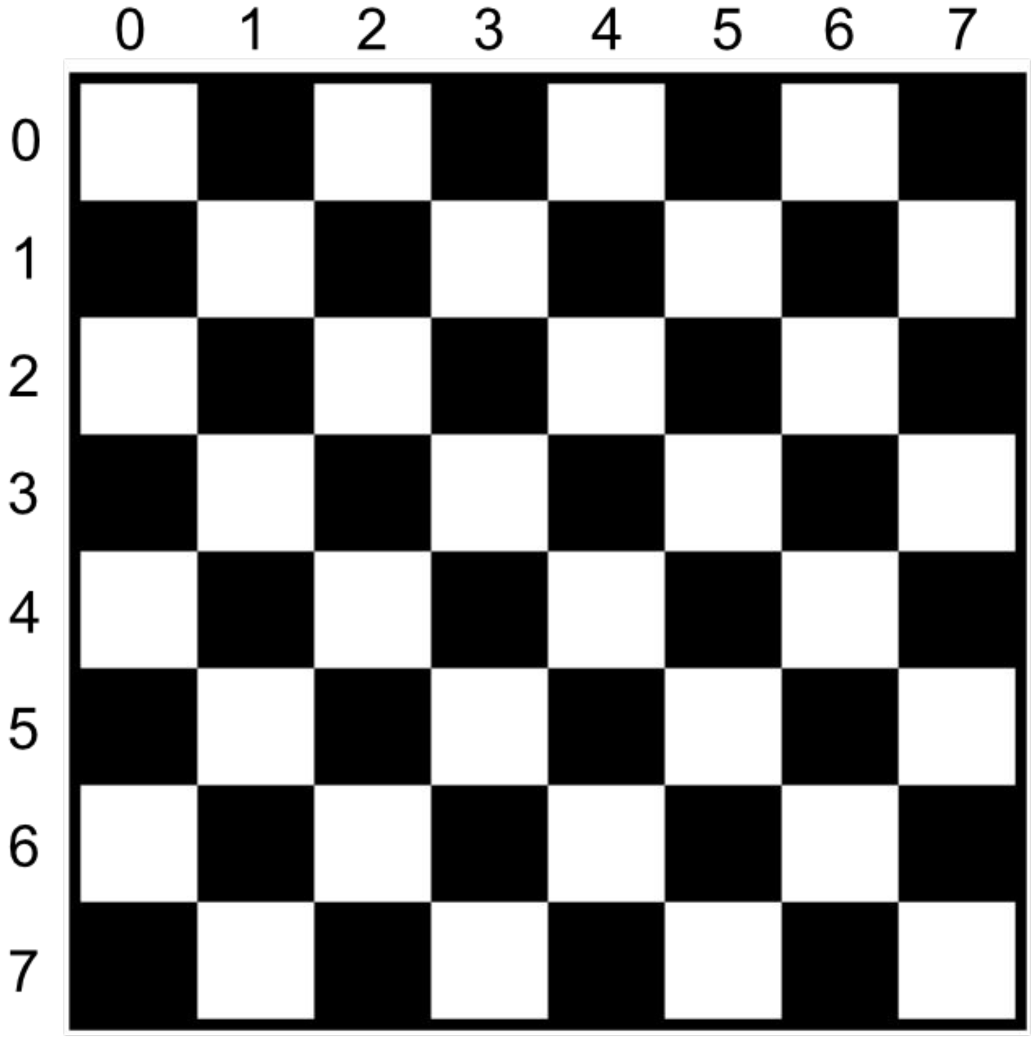


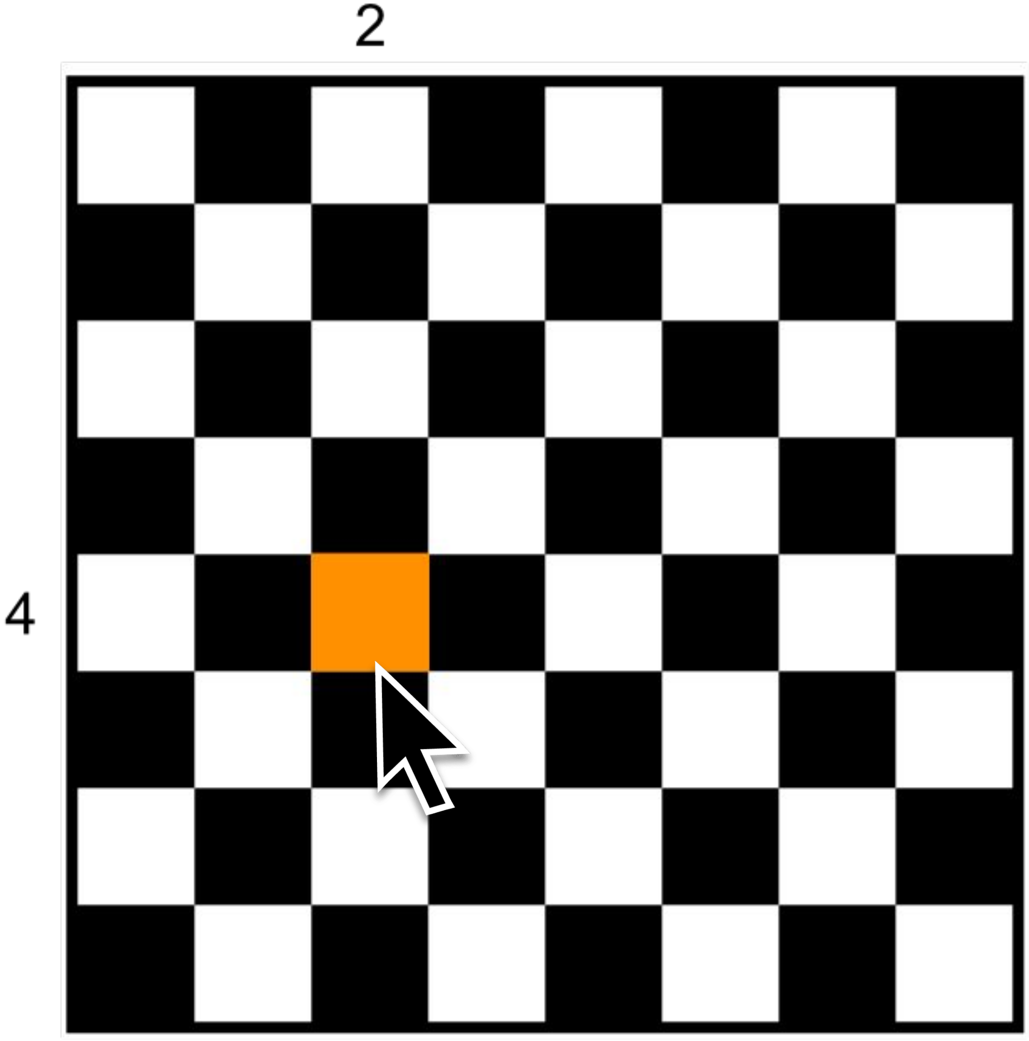
Major Diagonals



Minor Diagonals




0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7

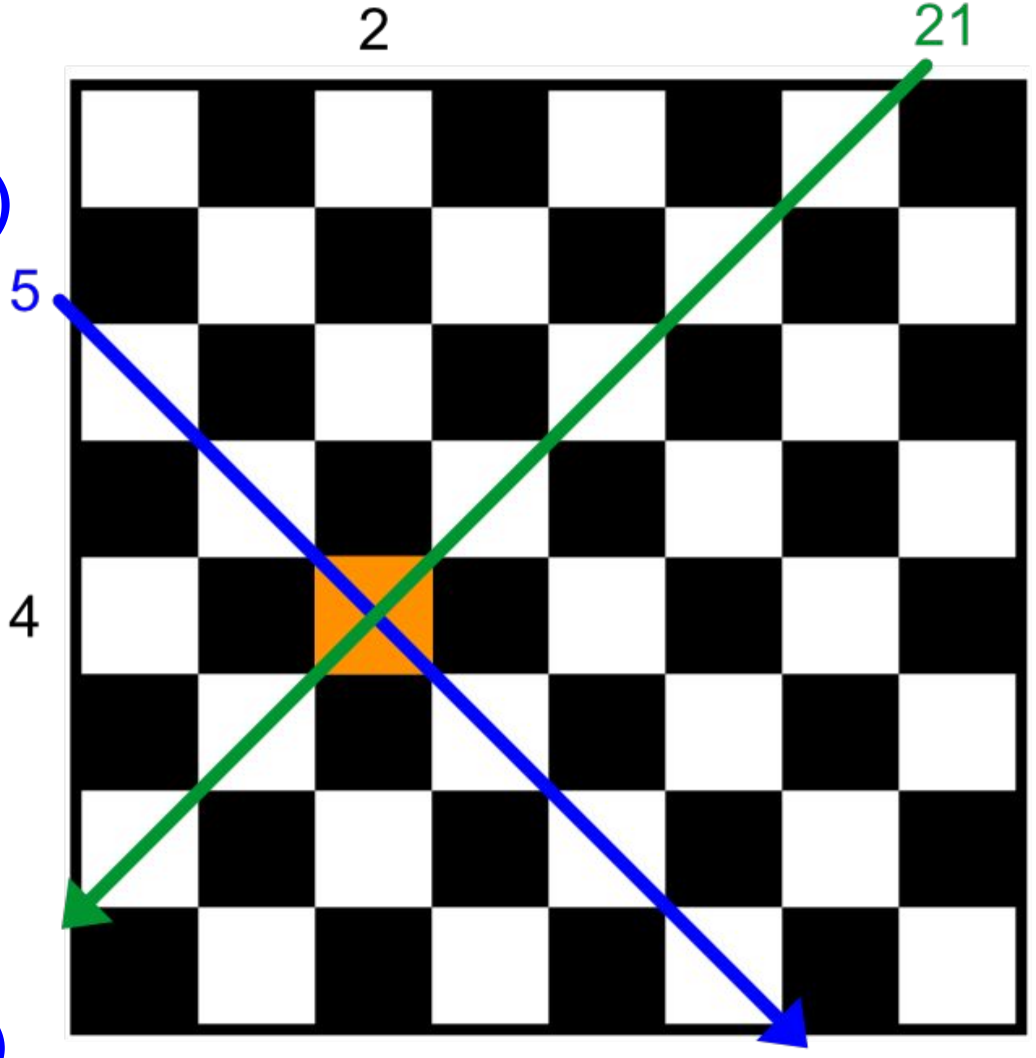


(4,2)

5 = 7 - (4 - 2)

5 = 7 - (2 - 0)
5 = 7 - (3 - 1)
5 = 7 - (4 - 2)
5 = 7 - (5 - 3)
5 = 7 - (6 - 4)
5 = 7 - (7 - 5)

7 - (row - col)



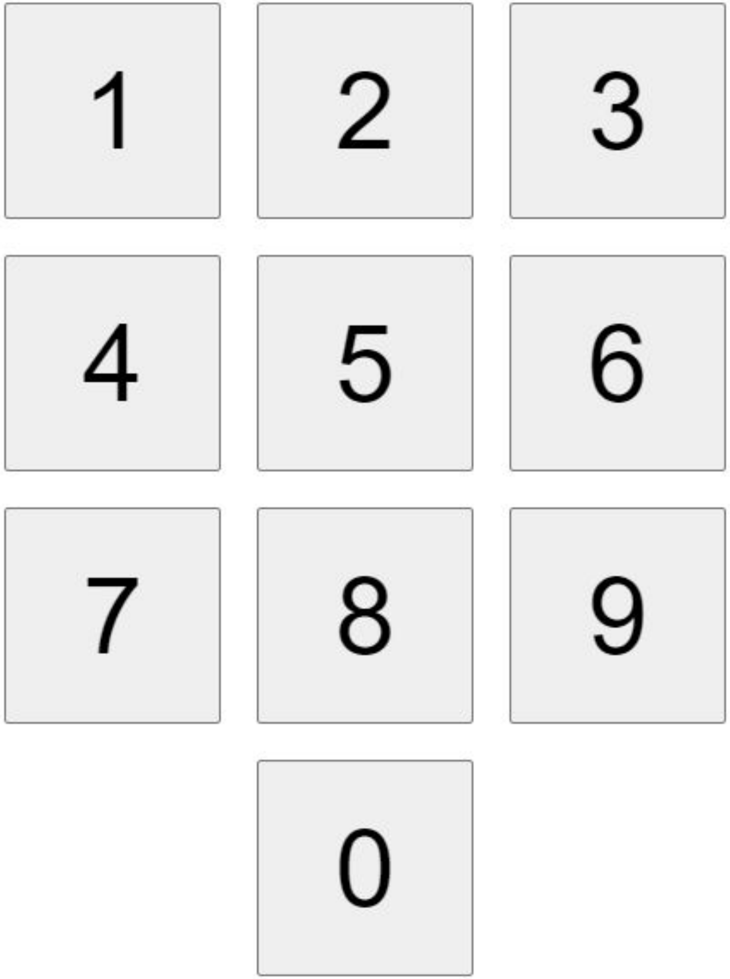
21 = 15 + (4 + 2)

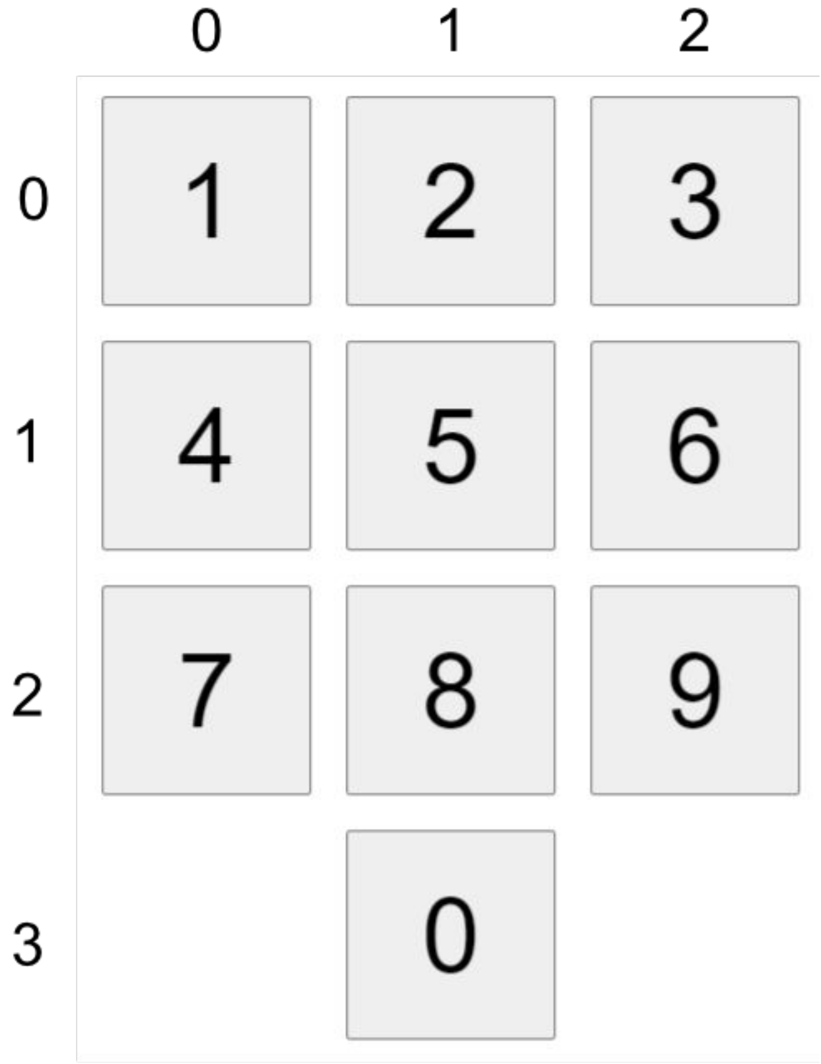
21 = 15 + (6 + 0)
21 = 15 + (5 + 1)
21 = 15 + (4 + 2)
21 = 15 + (3 + 3)
21 = 15 + (2 + 4)
21 = 15 + (1 + 5)
21 = 15 + (0 + 6)

15 + (row + col)

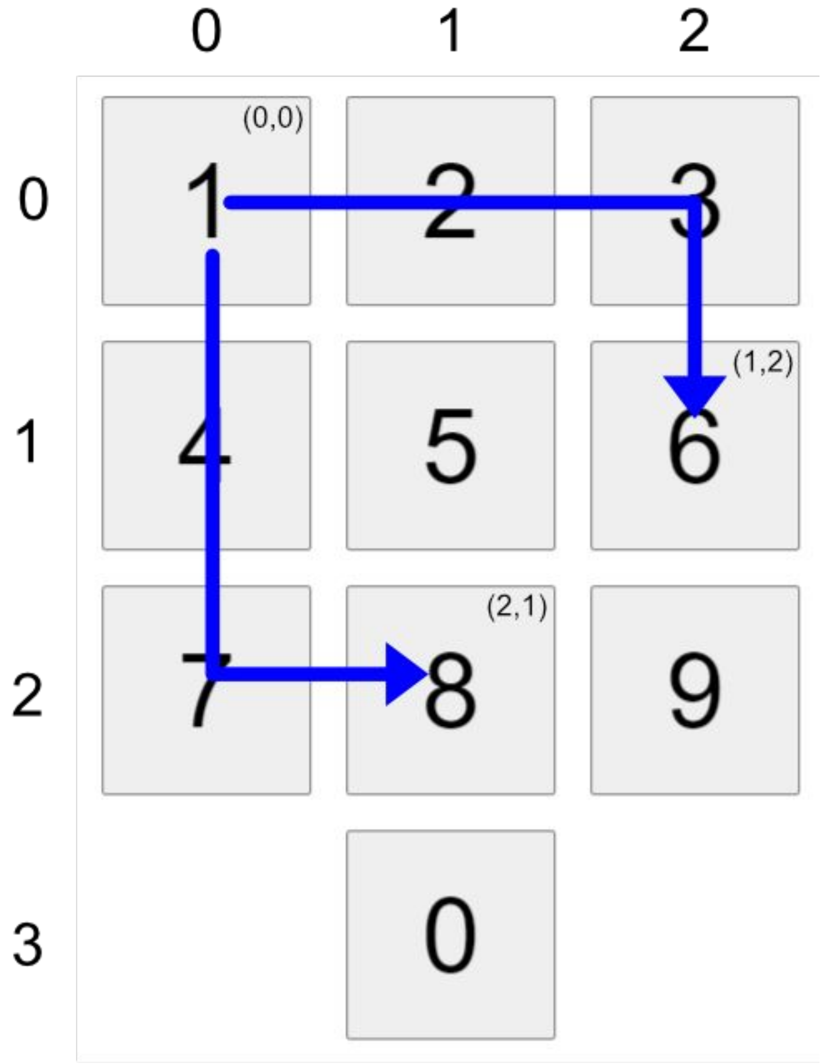
Exercise:
Knight’s Dialer

https://github.com/getify/workshop-knights-dialer


1
2
3
4
5
6
7
8
9
0

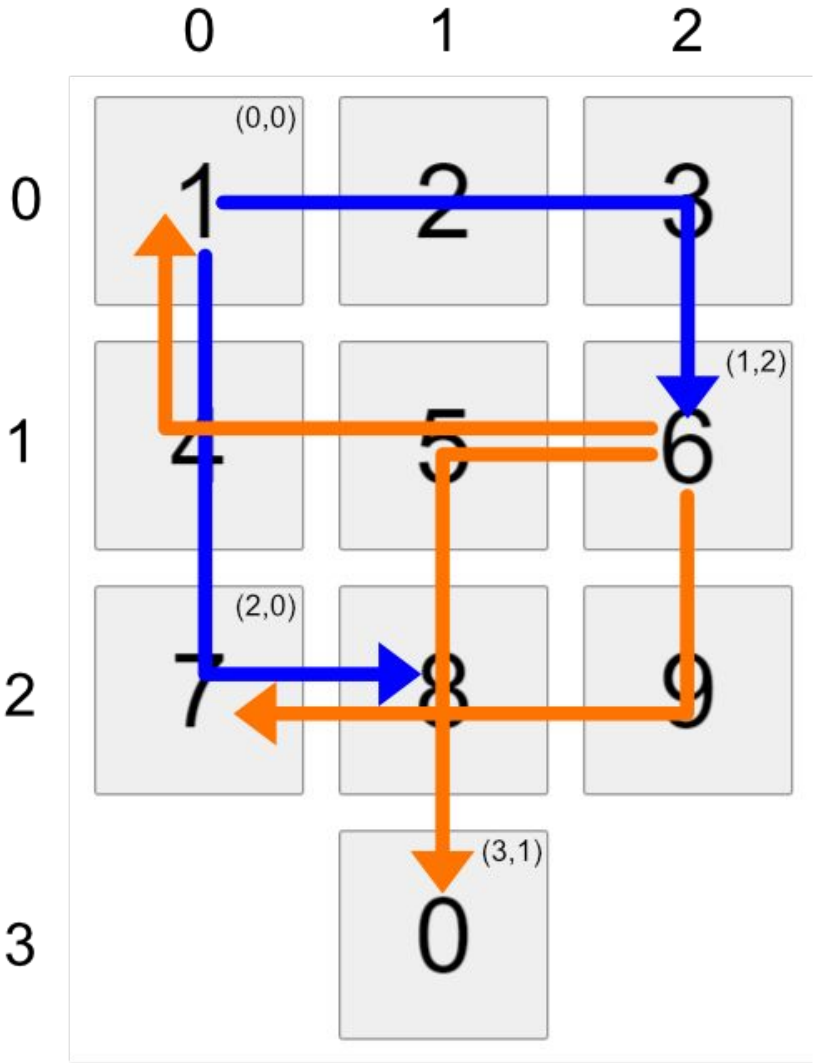


	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3		0	



row +/- 2
col +/- 1

row +/- 1
col +/- 2


0
1
2
0
1
2
3
(0,0)
1
2
3
(1,2)
4
5
6
(2,0)
7
8
9
(3,1)
0

countPaths(4,6) = 168

countPaths(3,5) + 52
countPaths(9,5) + 52
countPaths(0,5) 64

Recursion

countPaths(3,5) =	52
countPaths(4,4) +	32
countPaths(8,4)	20

Recursion

Breadth-First

Depth-First

Traversal

Breadth-First Traversal Iterative+Queue

Depth-First Traversal Recursion

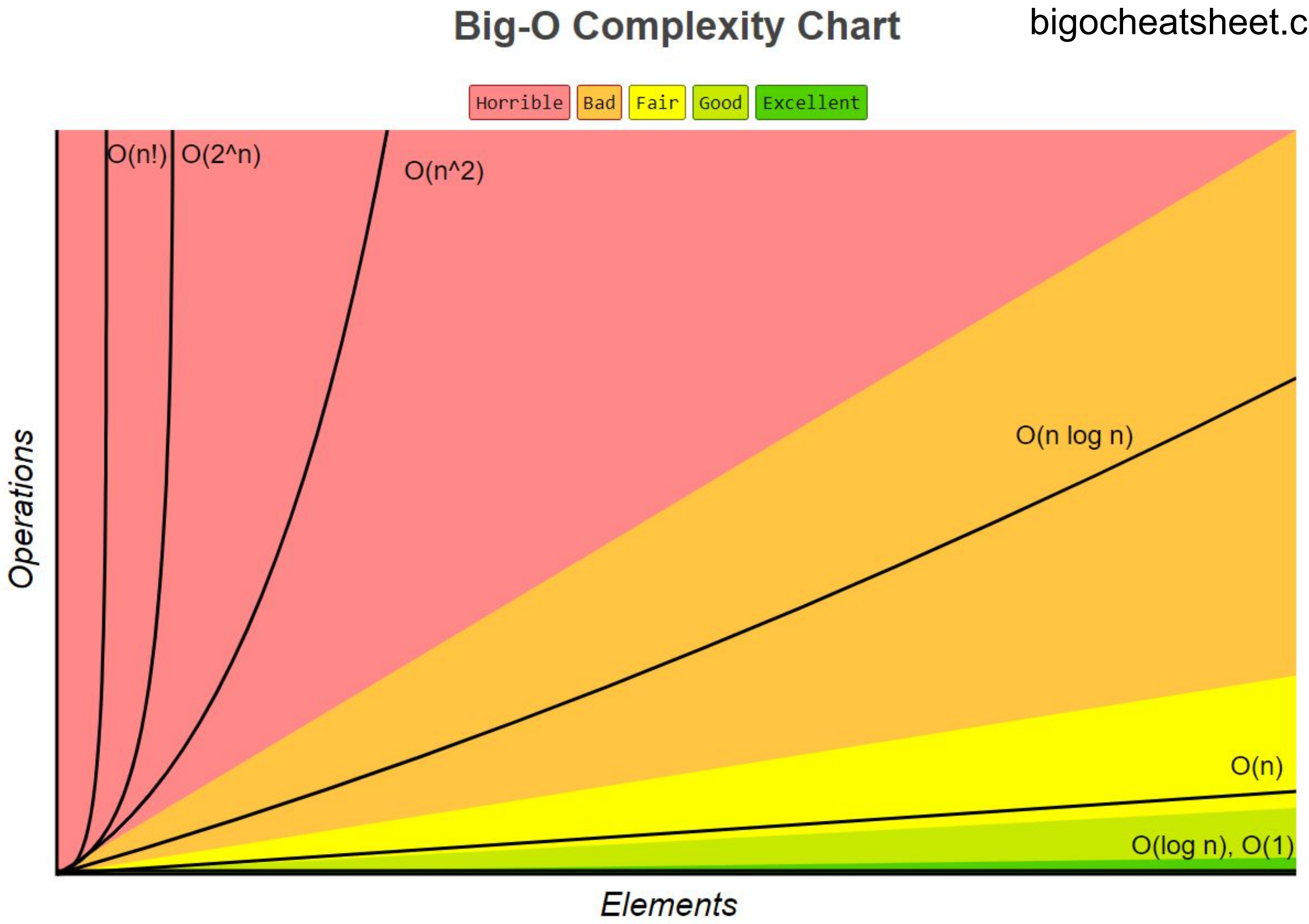
O(n): ~2.222^n

O(6): **~120 ops**   O(7): **~268 ops**   O(8): **~595 ops**

Exponential

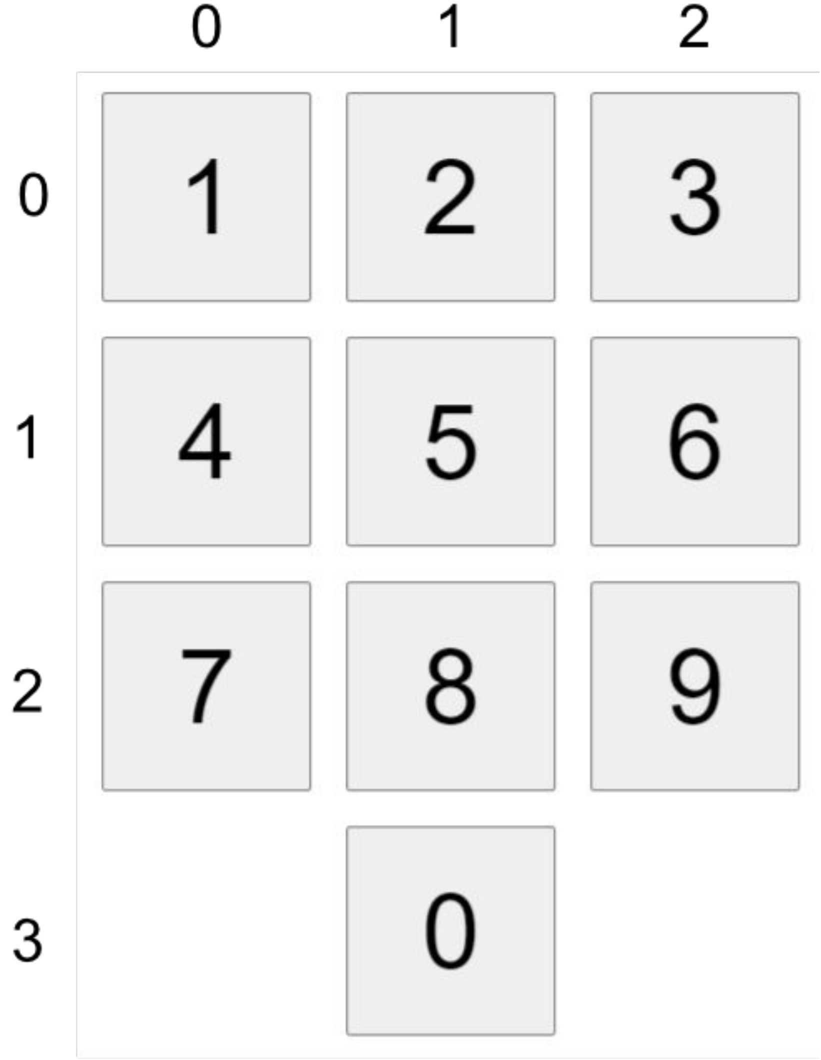
Recursion

Factorial
Exponential
Polynomic
Linear * Logarithmic
Linear
Logarithmic
Constant



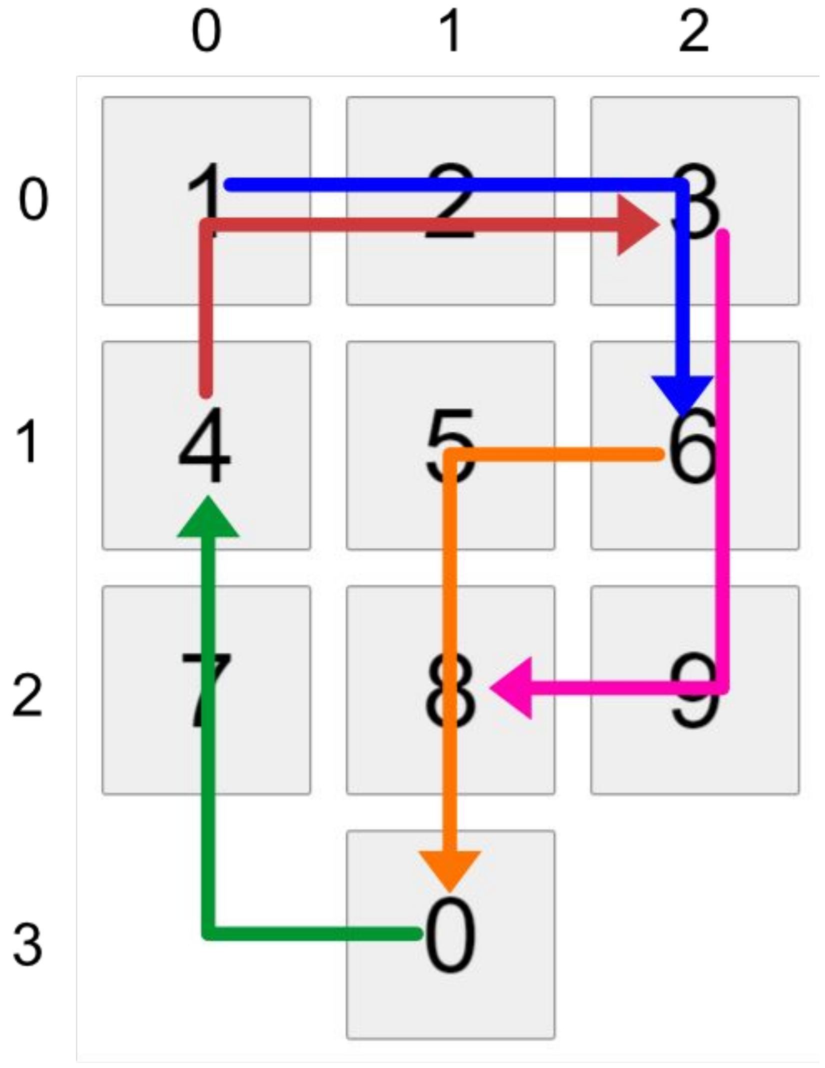
bigocheatsheet.com

Acyclic Paths



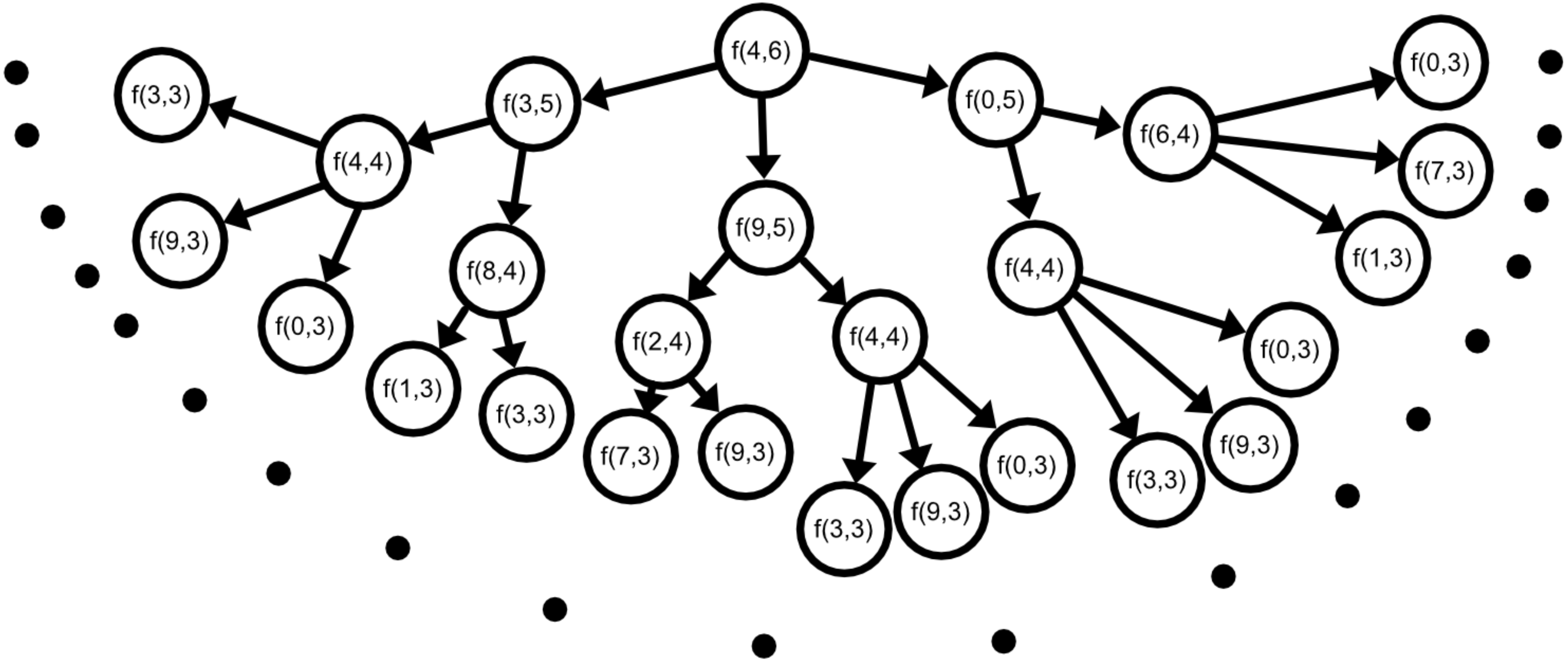
	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3		0	

Acyclic Paths

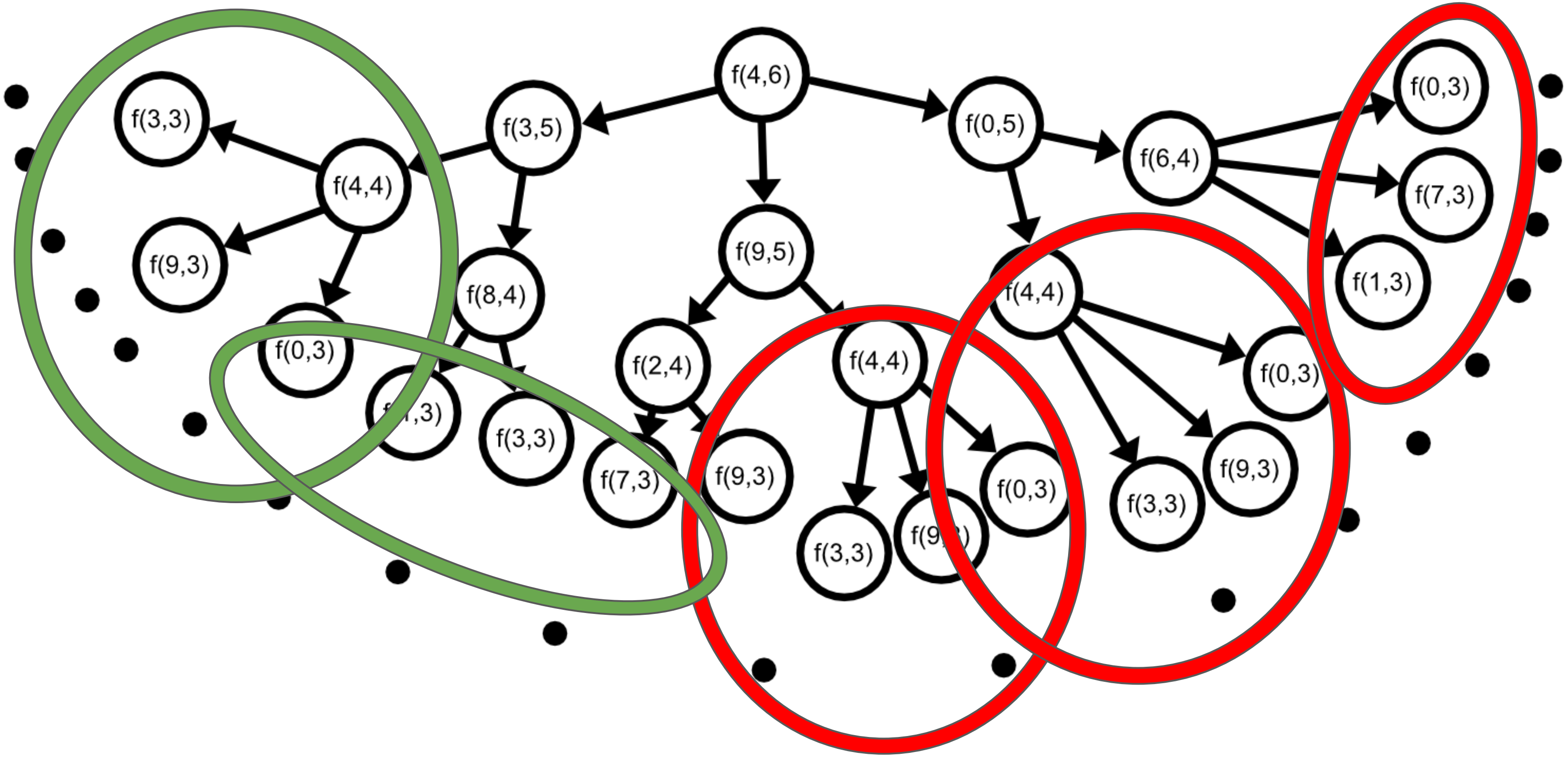


	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3		0	

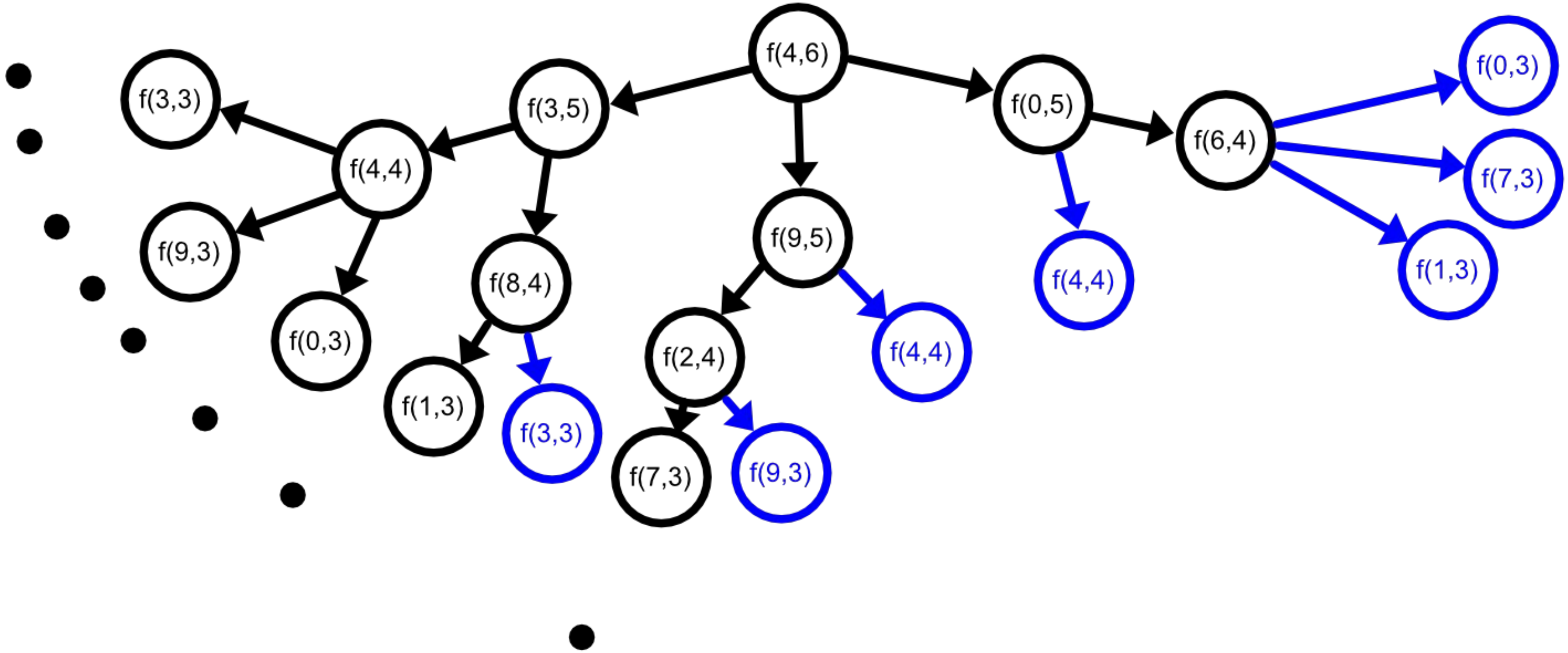
1 → 6 → 0 → 4 → 3 → 8



(Optimizing) Recursion



Recursion + Memoization



Recursion + Memoization

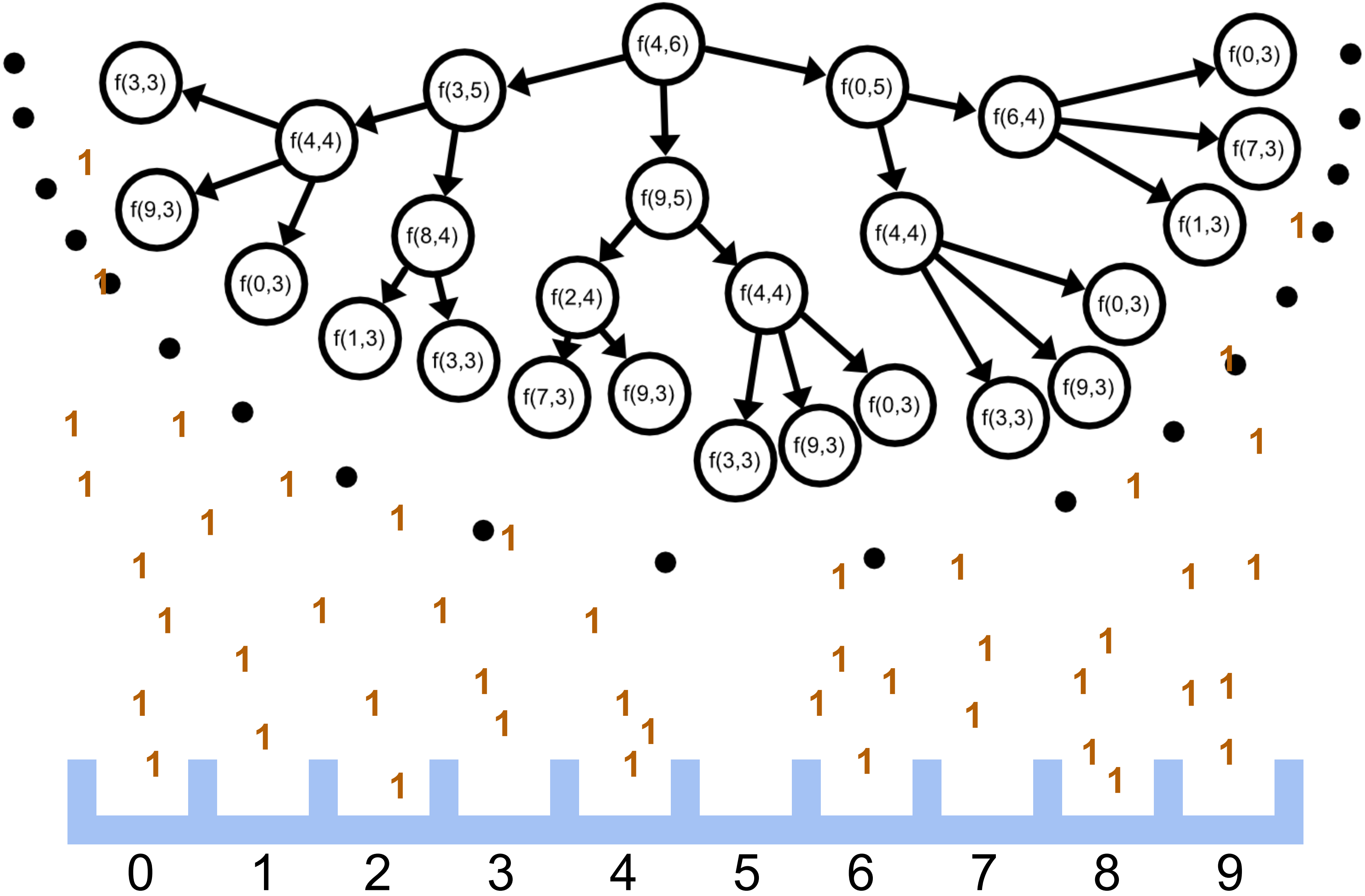
Top-Down

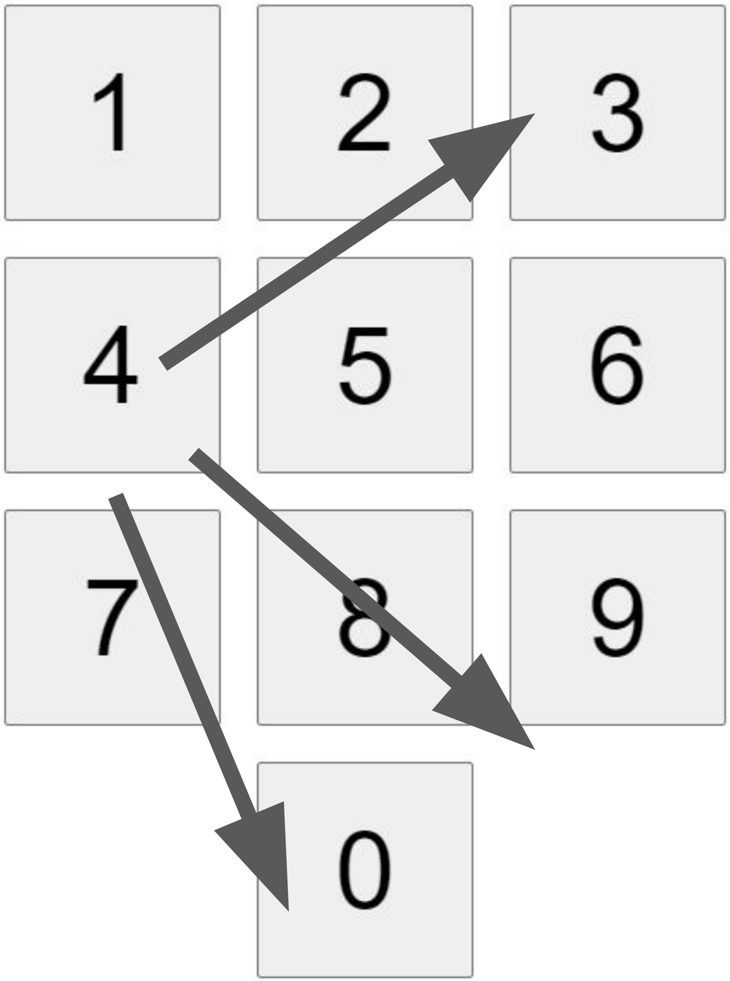
(memoization)

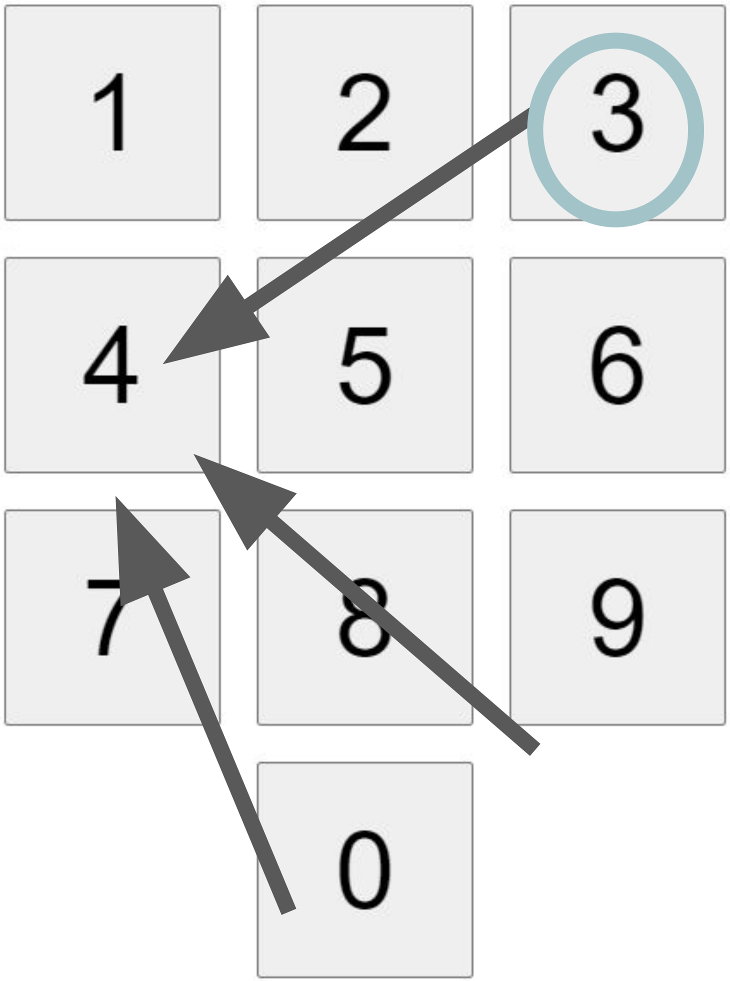
Bottom-Up

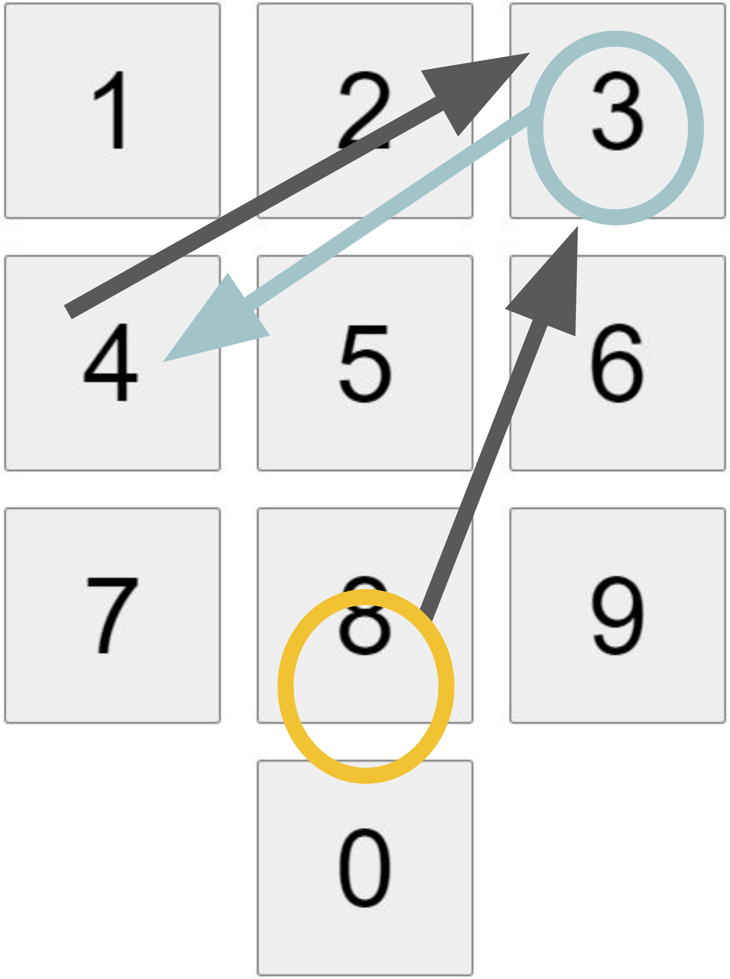
(tabulation)

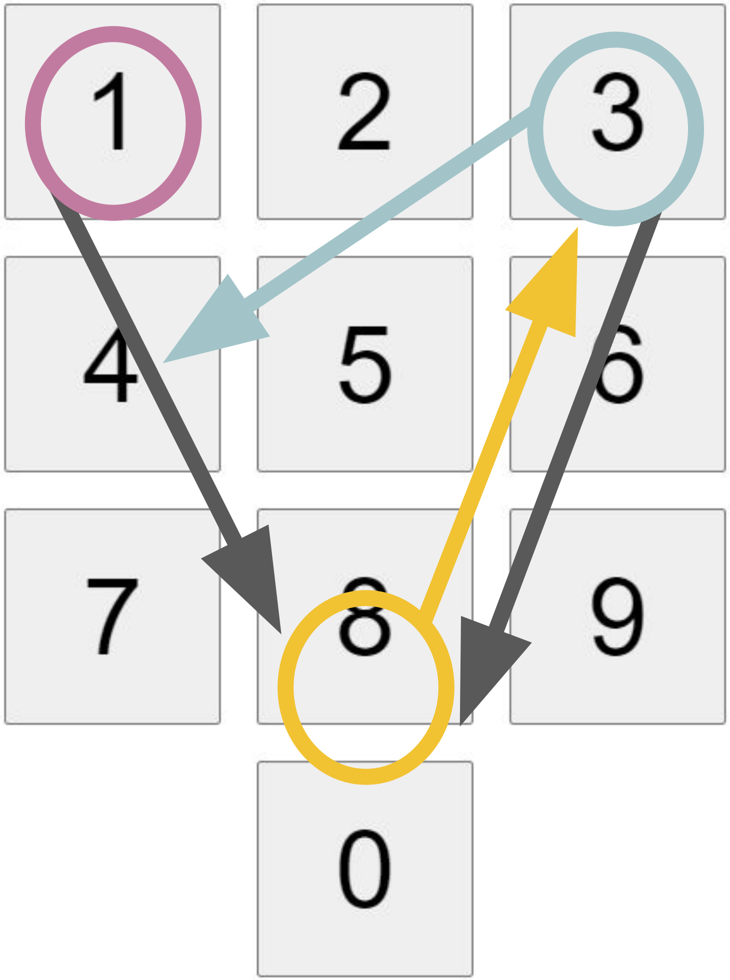
Dynamic Programming


f(4,6)
f(3,5)
f(0,5)
f(3,3)
f(4,4)
f(9,3)
f(0,3)
f(8,4)
f(1,3)
f(3,3)
f(9,5)
f(2,4)
f(7,3)
f(9,3)
f(4,4)
f(3,3)
f(9,3)
f(0,3)
f(6,4)
f(0,3)
f(7,3)
f(1,3)
f(4,4)
f(0,3)
f(9,3)
f(3,3)
0
1
2
3
4
5
6
7
8
9

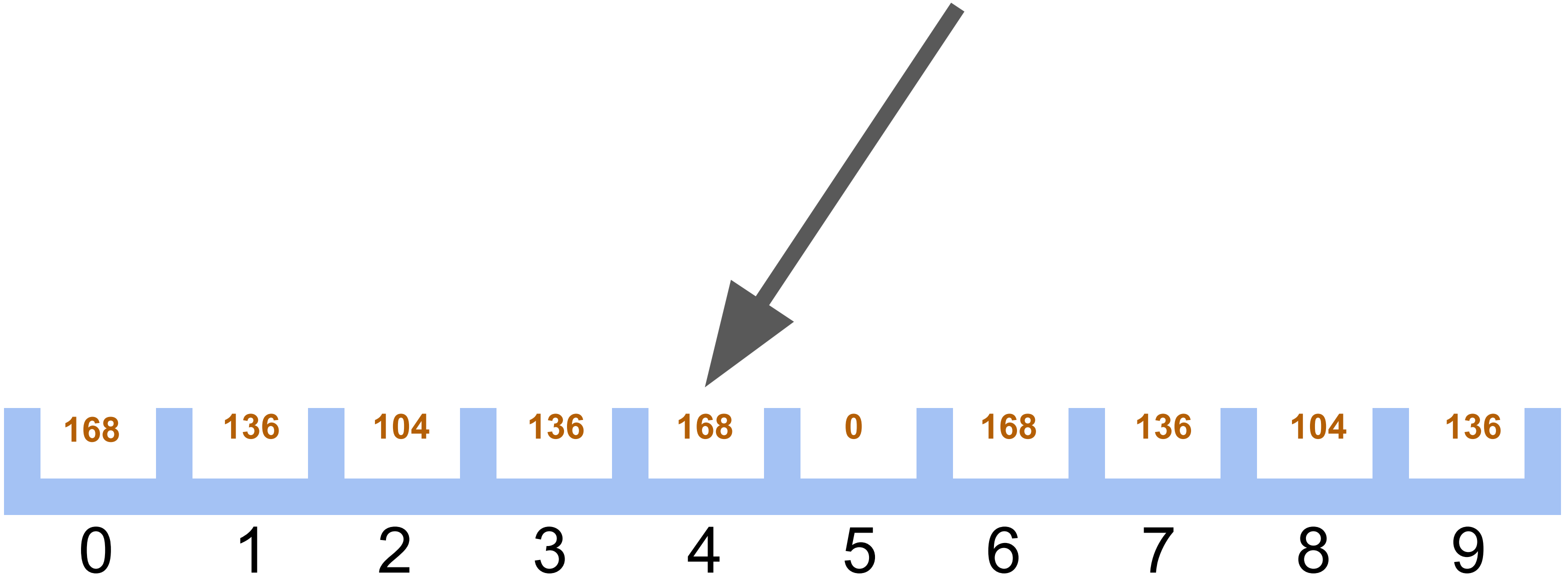

1
2
3
4
5
6
7
8
9
0


1
2
3
4
5
6
7
8
9
0


1
2
3
4
5
6
7
8
9
0

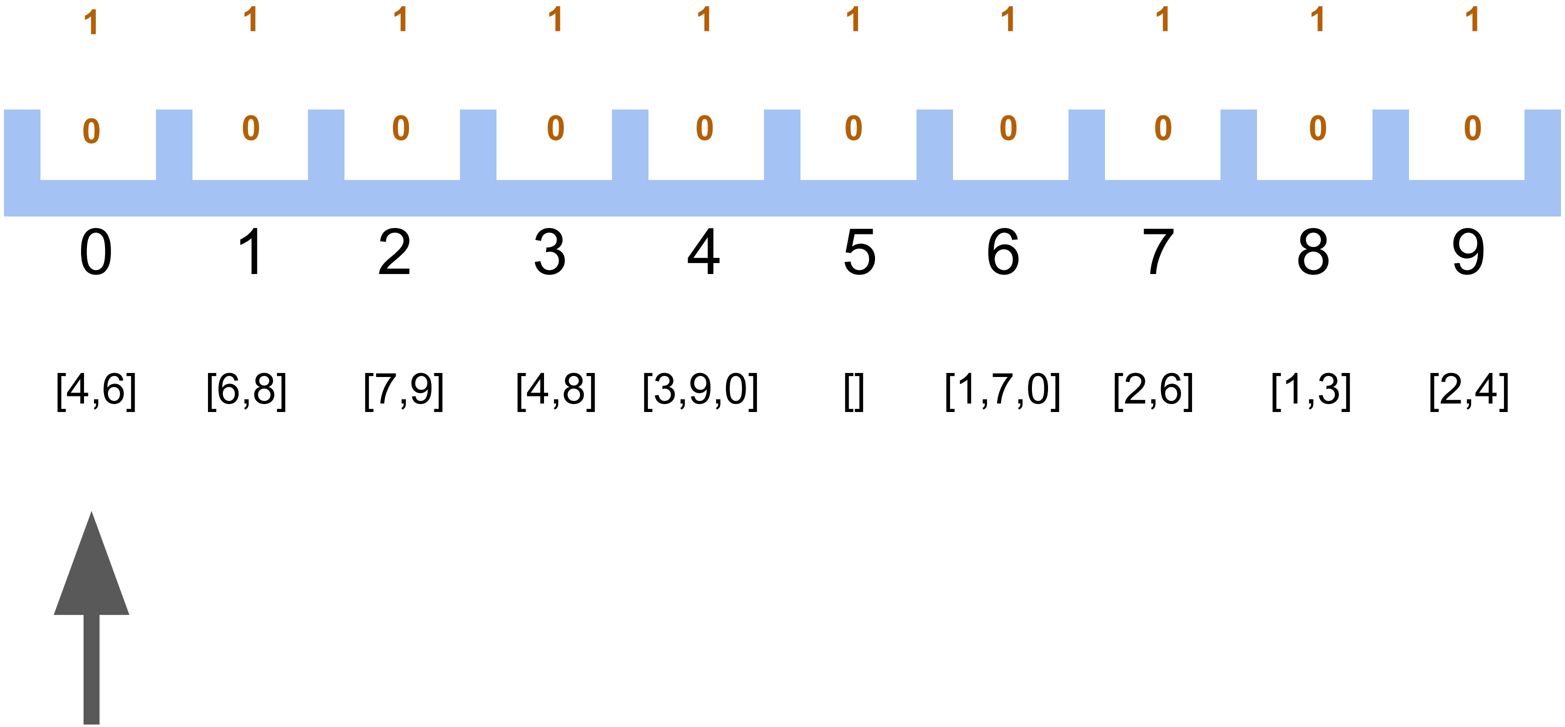

1
2
3
4
5
6
7
8
9
0

countPaths(4,6)



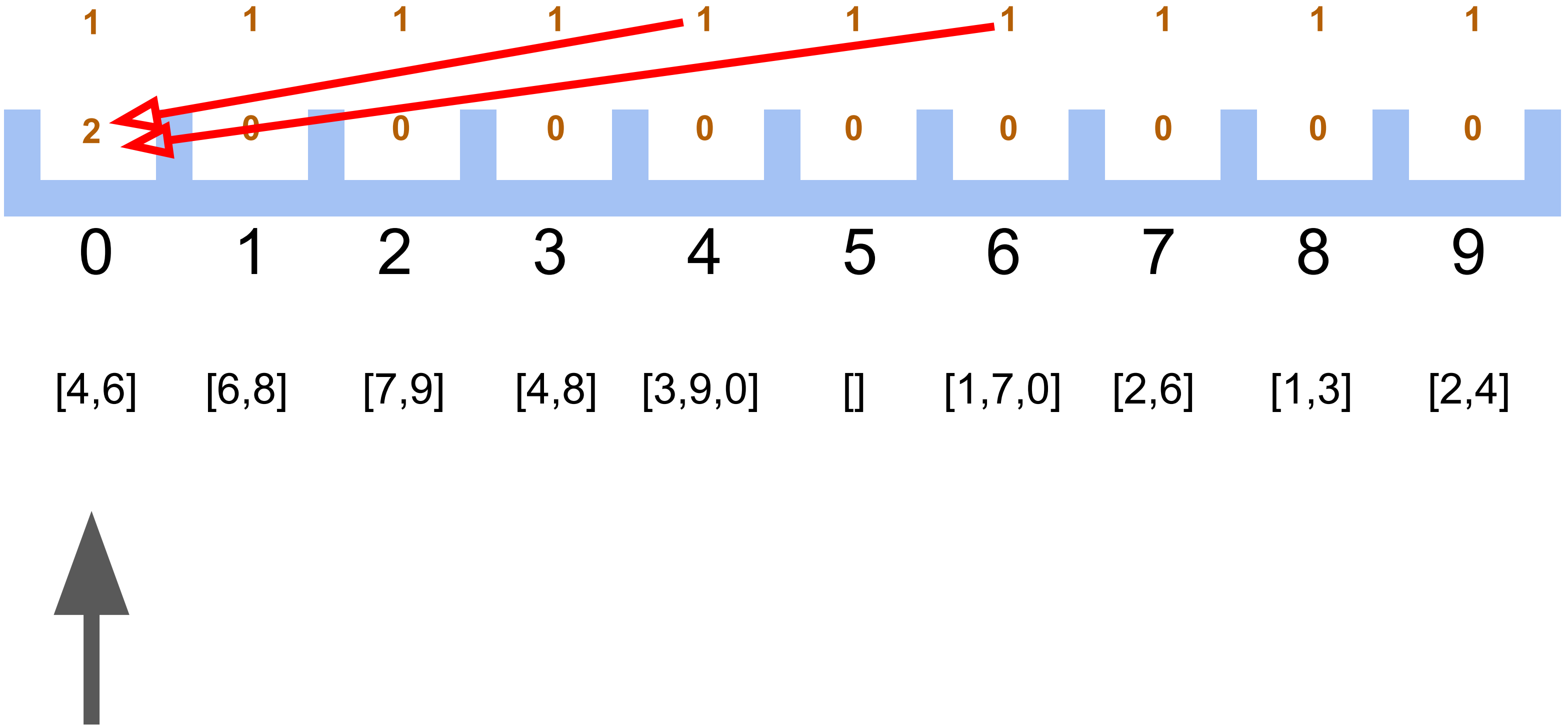
168	136	104	136	168	0	168	136	104	136
0	1	2	3	4	5	6	7	8	9

Iteration: 1
hopCount: 6

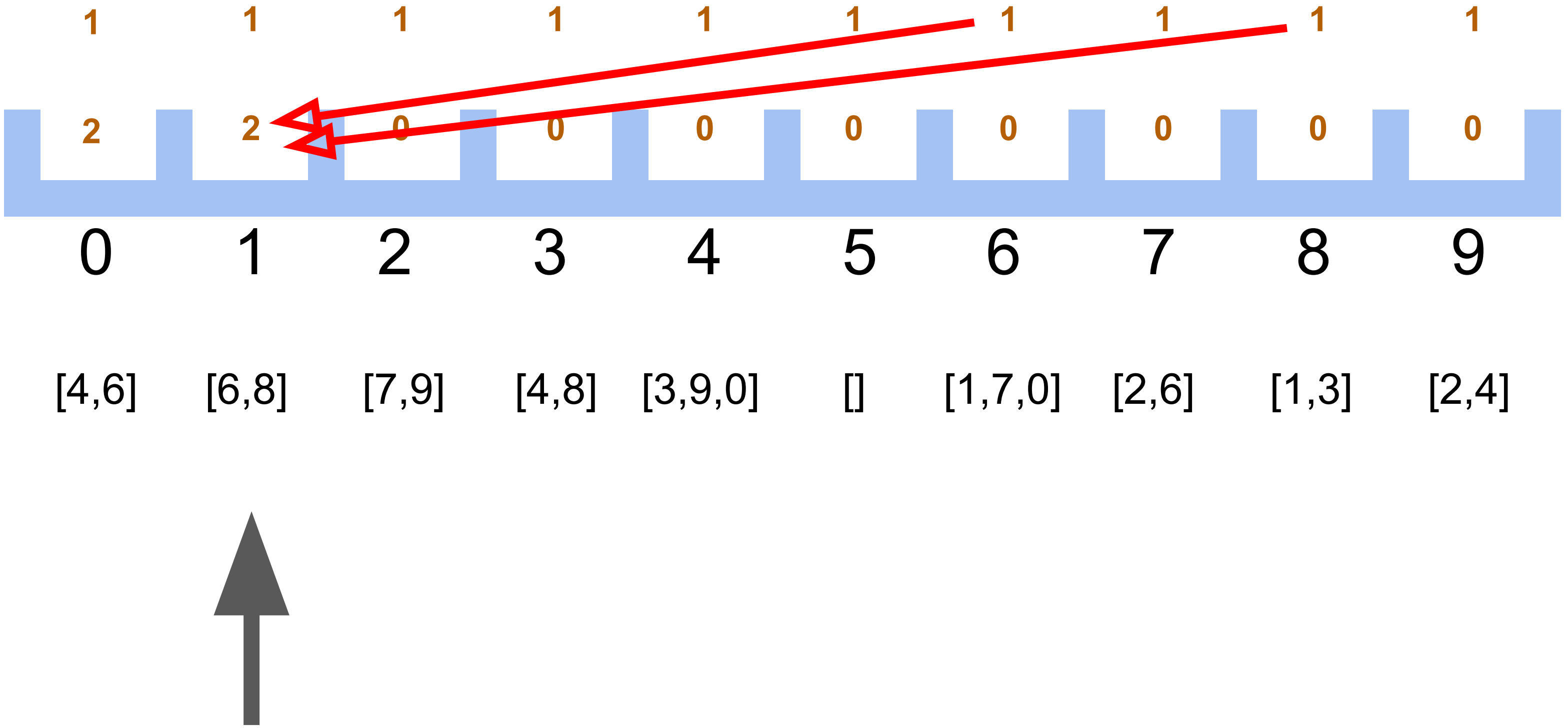


1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

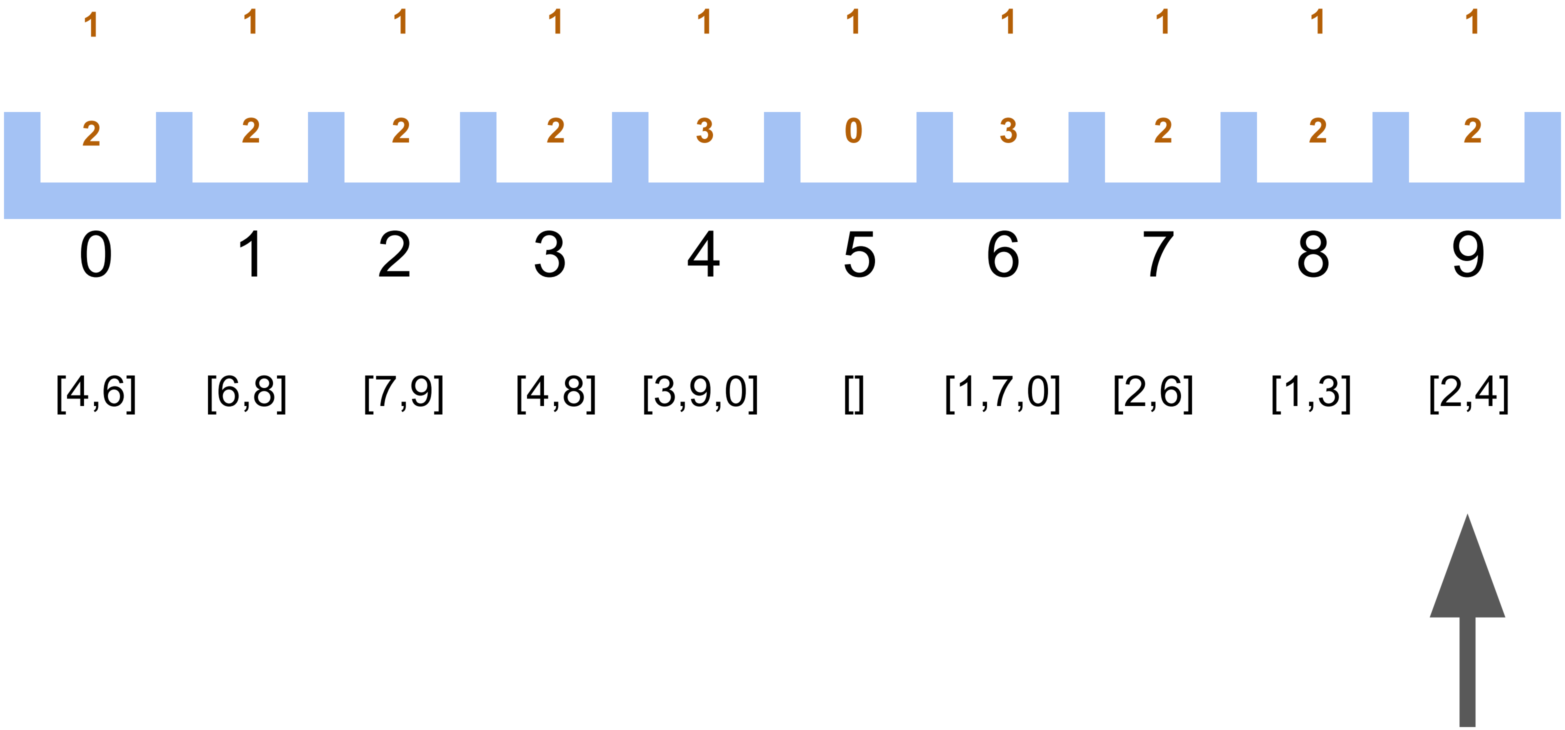
Iteration: 1
hopCount: 6



Iteration: 1
hopCount: 6

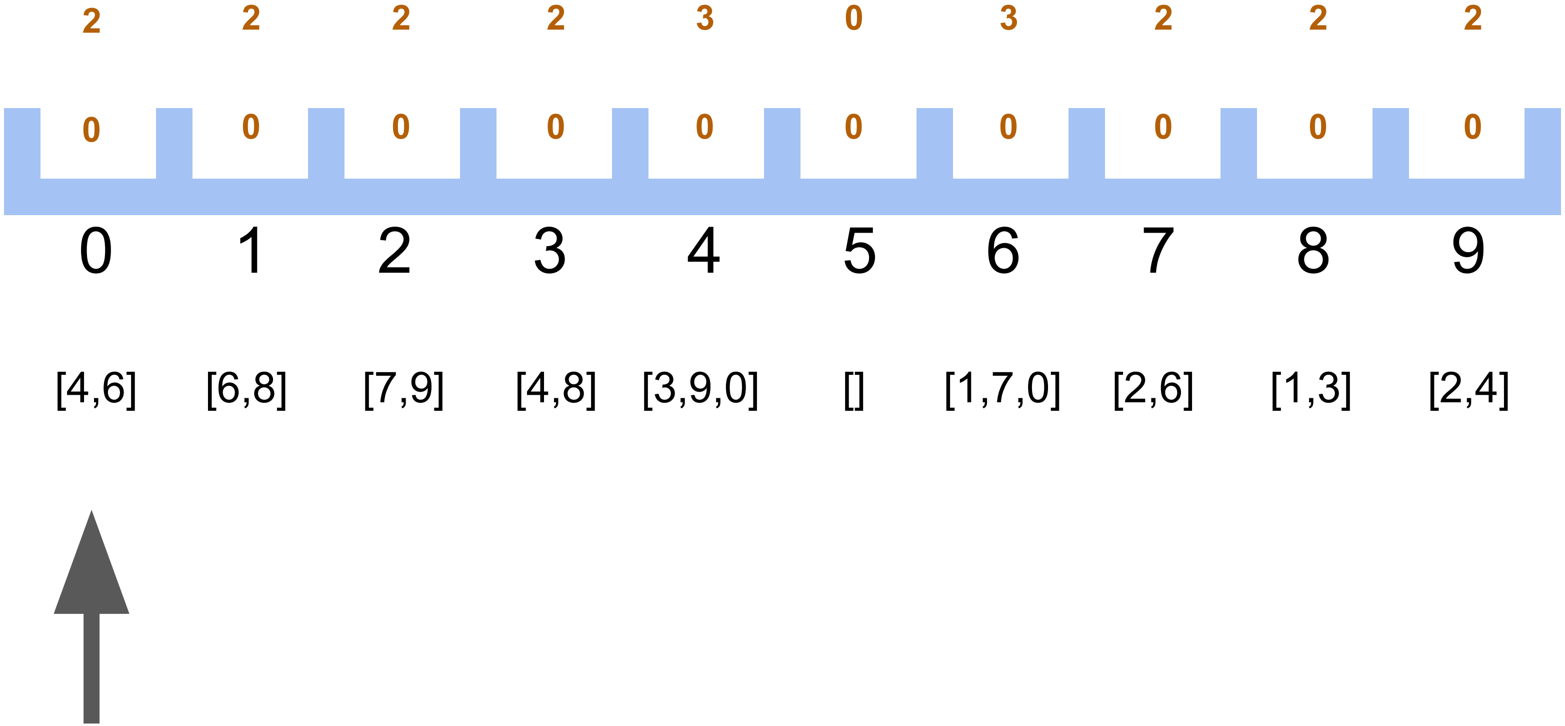


Iteration: 1
hopCount: 6



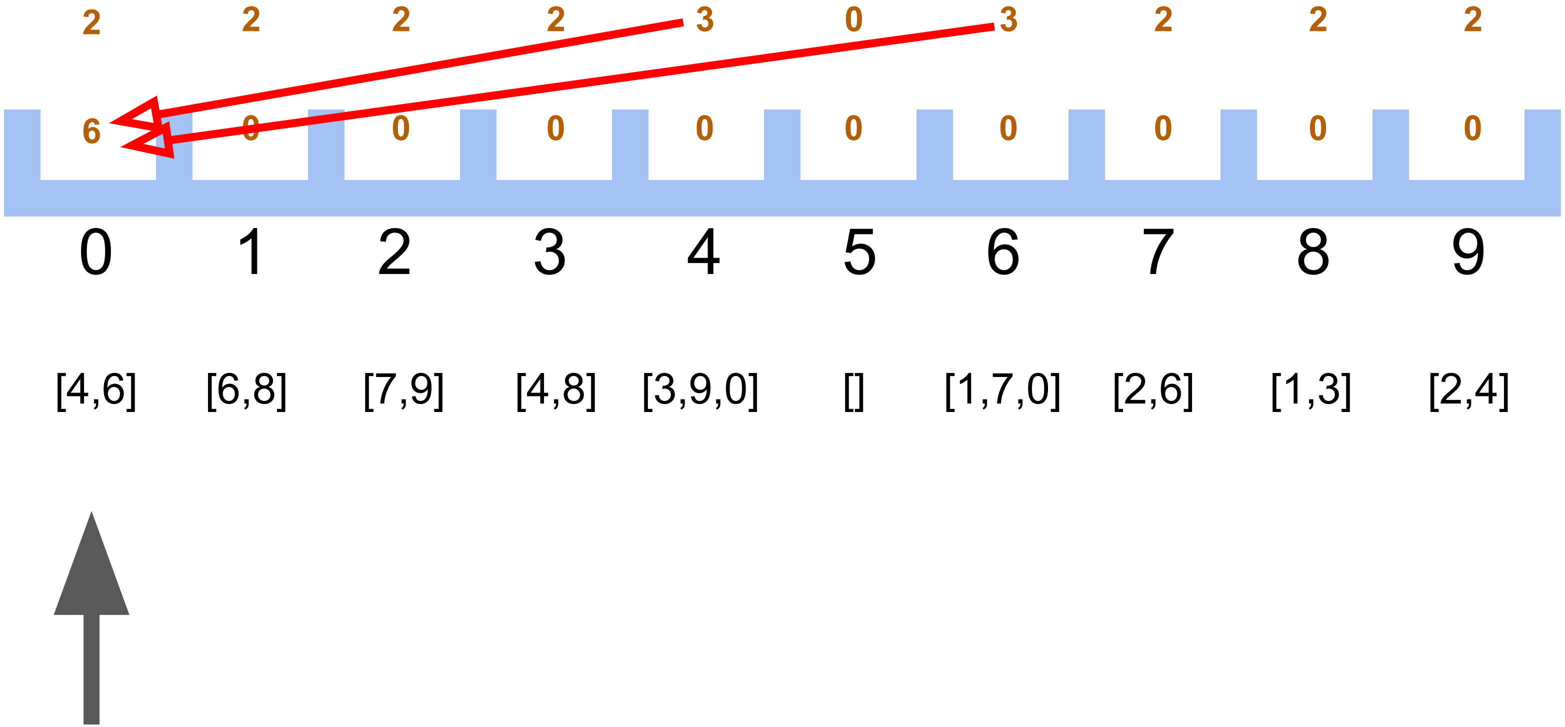
1	1	1	1	1	1	1	1	1	1
2	2	2	2	3	0	3	2	2	2
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

Iteration: 2
hopCount: 5



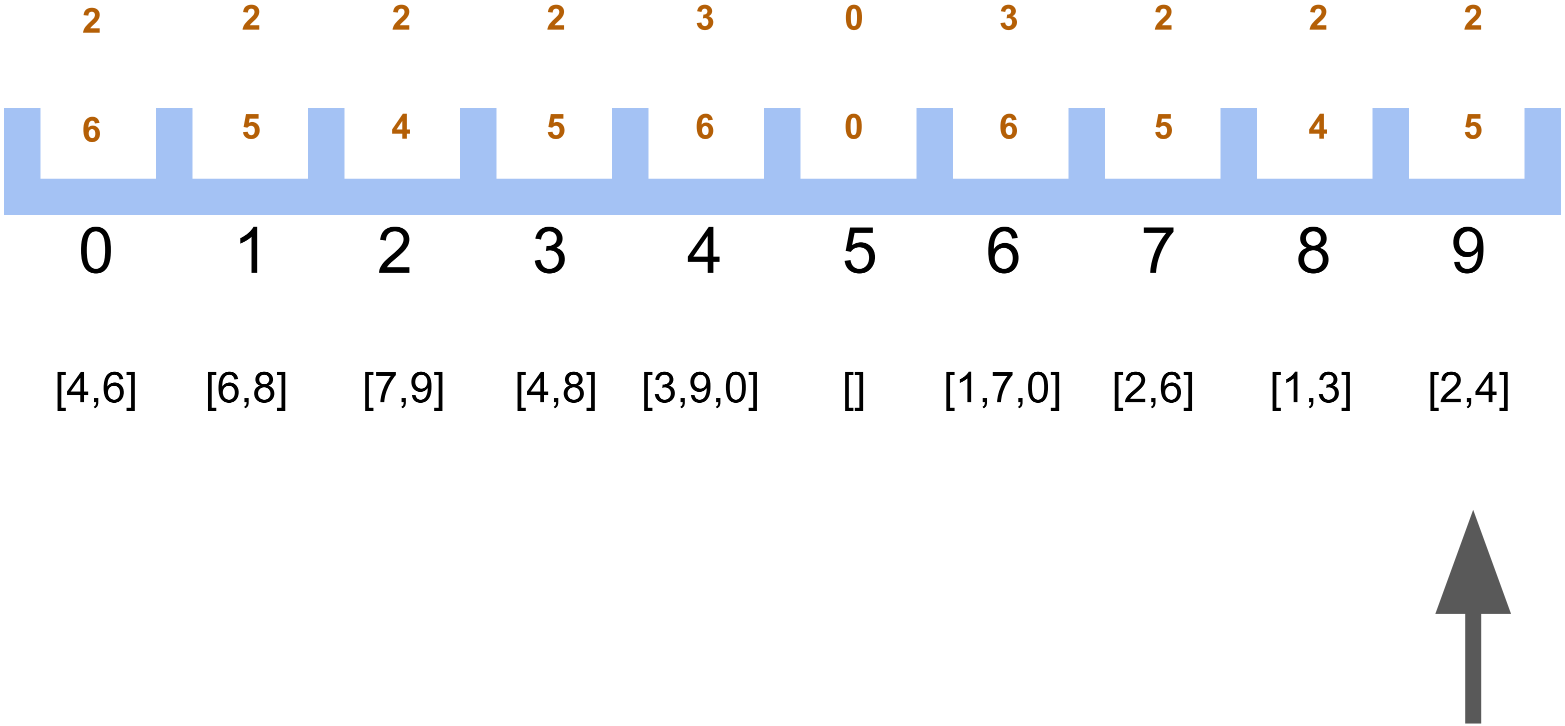
2	2	2	2	3	0	3	2	2	2
0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

Iteration: 2
hopCount: 5



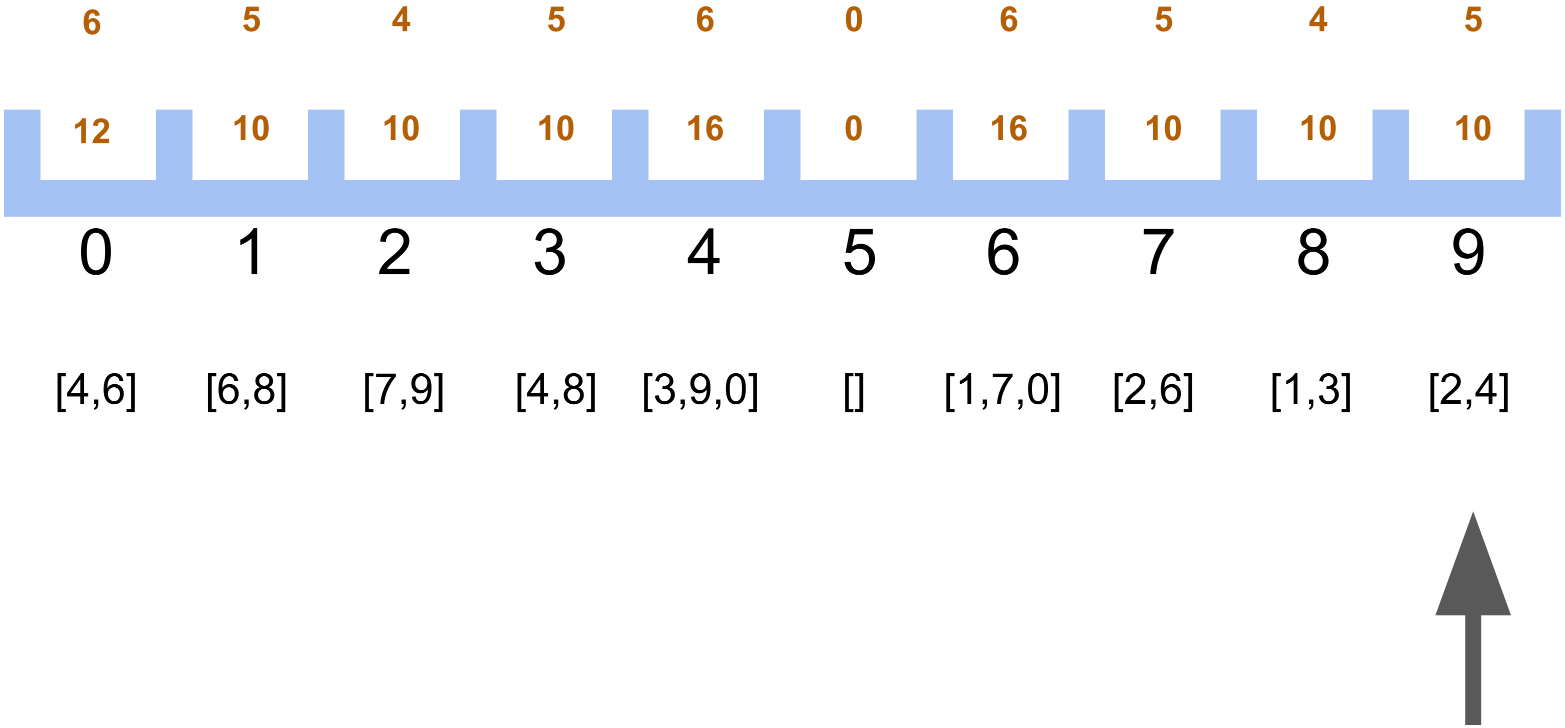
2	2	2	2	3	0	3	2	2	2
6	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

Iteration: 2
hopCount: 5



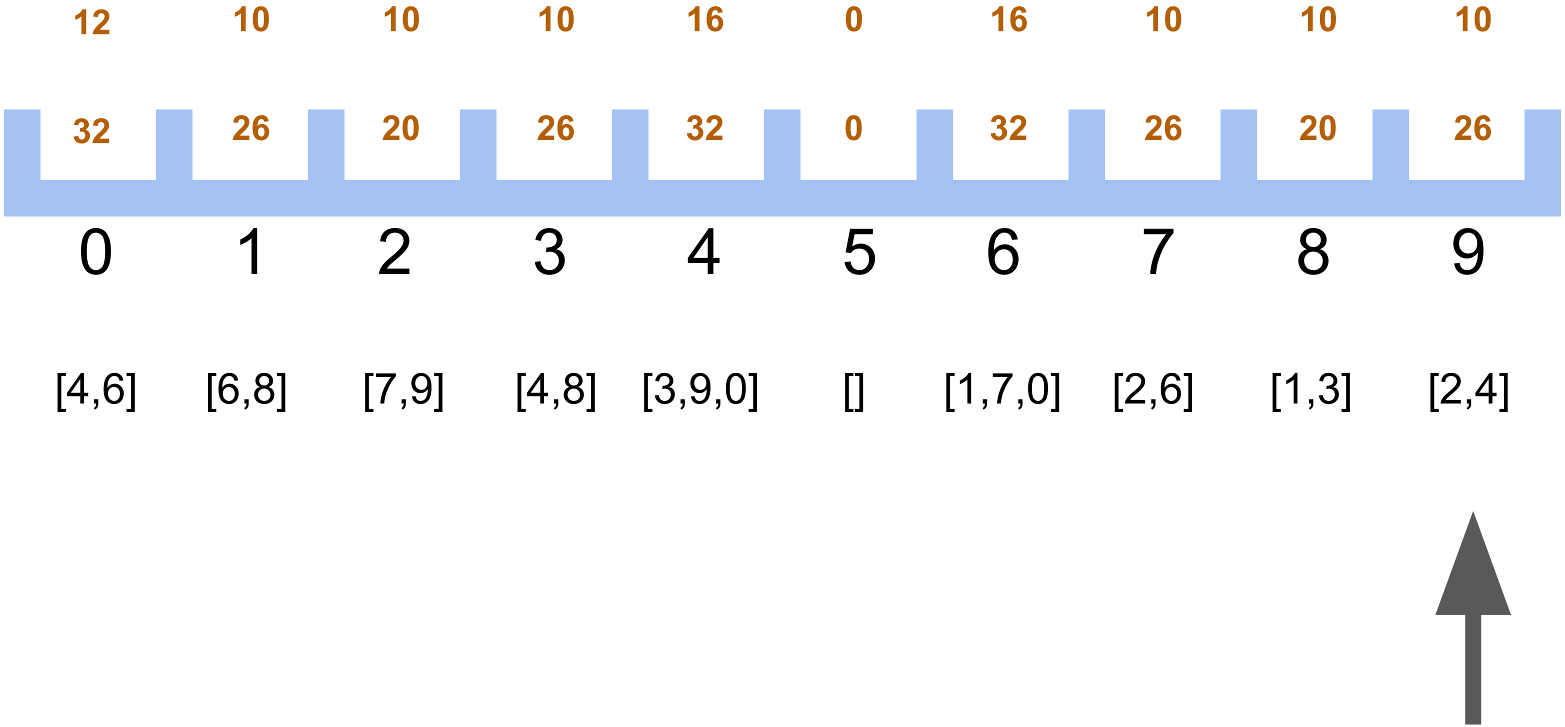
2	2	2	2	3	0	3	2	2	2
6	5	4	5	6	0	6	5	4	5
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

Iteration: 3
hopCount: 4



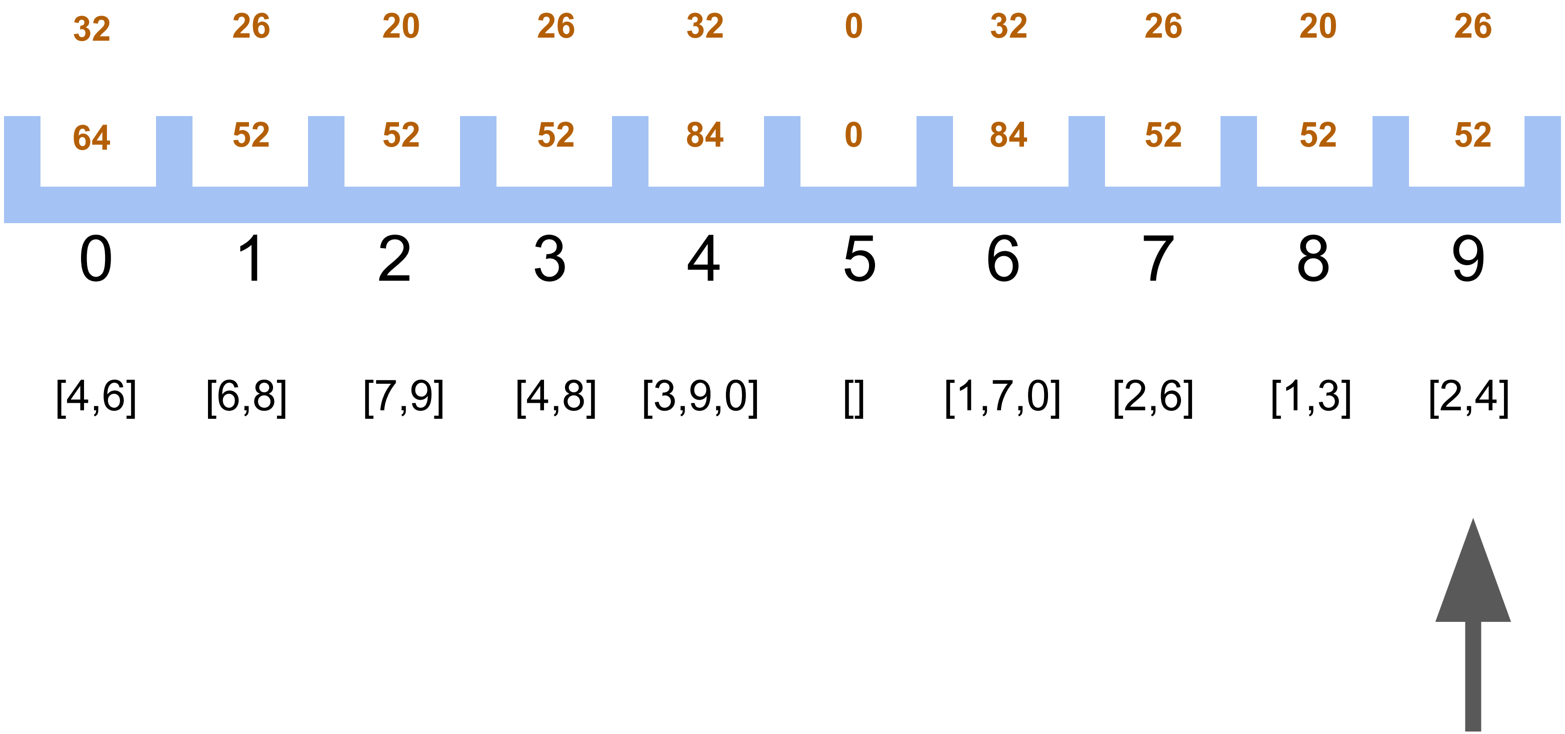
6	5	4	5	6	0	6	5	4	5
12	10	10	10	16	0	16	10	10	10
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

Iteration: 4
hopCount: 3



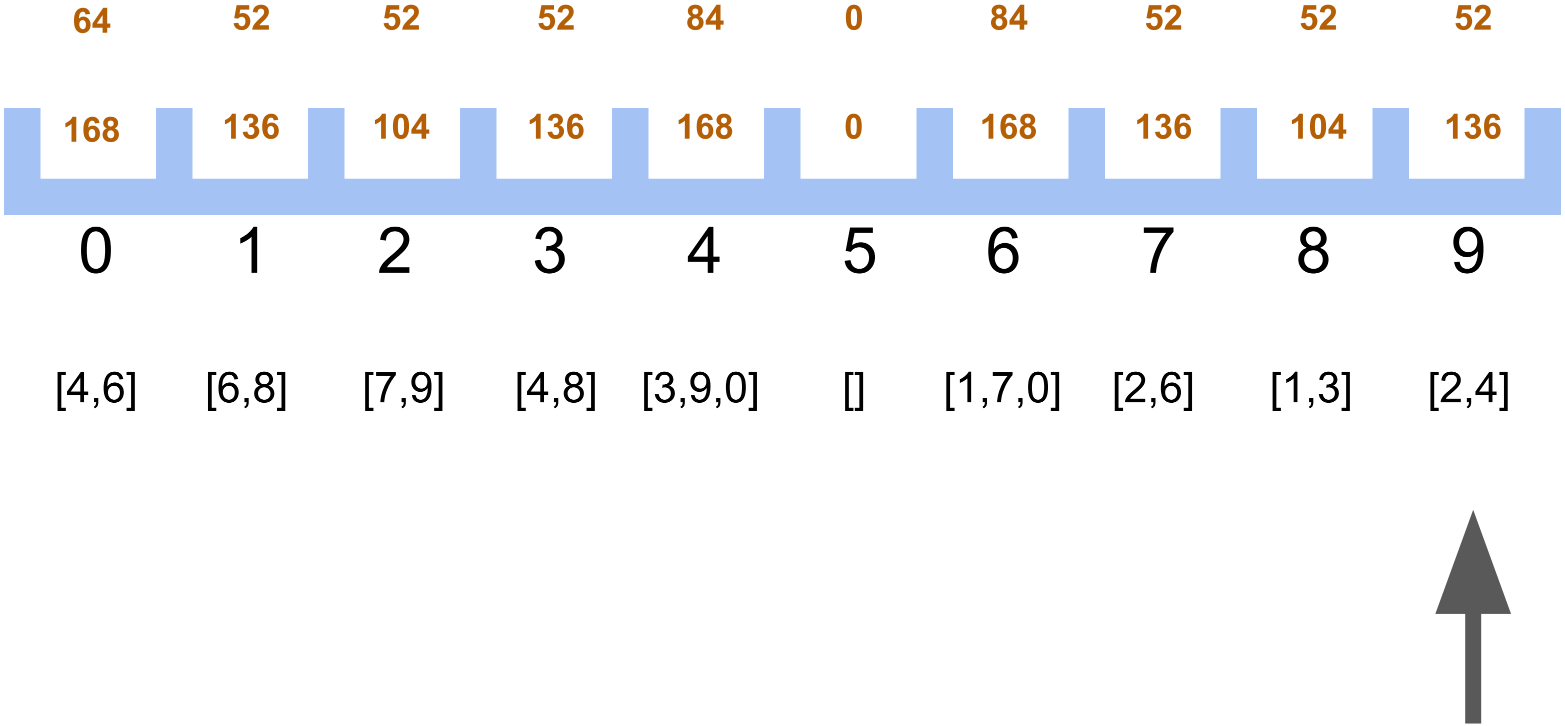
12	10	10	10	16	0	16	10	10	10
32	26	20	26	32	0	32	26	20	26
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

Iteration: 5
hopCount: 2



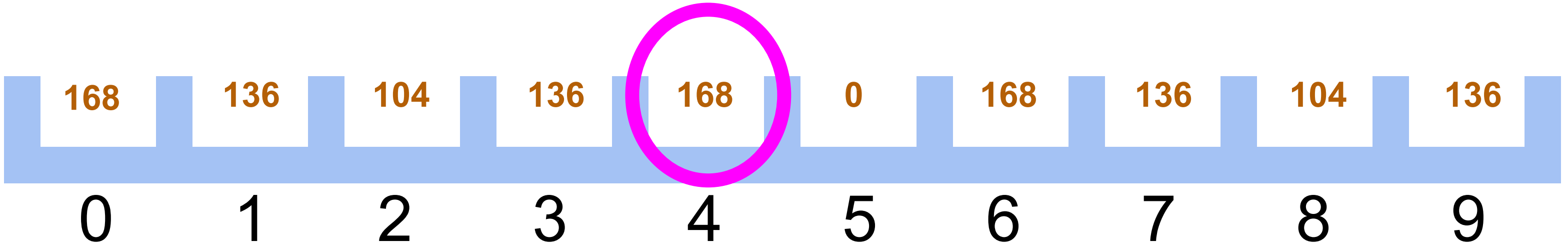
32	26	20	26	32	0	32	26	20	26
64	52	52	52	84	0	84	52	52	52
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

Iteration: 6
hopCount: 1



64	52	52	52	84	0	84	52	52	52
168	136	104	136	168	0	168	136	104	136
0	1	2	3	4	5	6	7	8	9
[4,6]	[6,8]	[7,9]	[4,8]	[3,9,0]	[]	[1,7,0]	[2,6]	[1,3]	[2,4]

countPaths(4,6)



168	136	104	136	168	0	168	136	104	136
0	1	2	3	4	5	6	7	8	9

O(n): n * 20 => n

O(6): **120 ops** O(60): **1,200 ops** O(600): **12,000 ops**

Linear

Exercise:
Wordy Unscrambler

https://github.com/getify/workshop-wordy-unscrambler

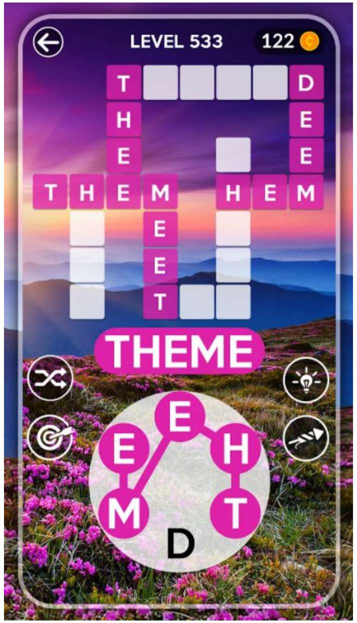
Short Dictionary (~2400 words) (entries: 2426) (**0.0** ms)

Letters: PCERA unscramble (**28.1** ms)

Input: **PCERA**

ACRE
ARC
ARE
CAP
CAPE
CAR
CARE
EAR
PER
RACE
REAP

Wordscapes



CAN

CAP

CAPE

CAR

CARE

CLAP

EACH

EAR

PCERA

CAN

✓ CAP

✓ CAPE

✓ CAR

✓ CARE

CLAP

EACH

✓ EAR

PCERA

✓ APPEAR

CAN

✓ CAP

✓ CAPE

✓ CAR

✓ CARE

CLAP

EACH

✓ EAR

PCERA

APPEAR

CAN

✓ CAP

✓ CAPE

✓ CAR

✓ CARE

CLAP

EACH

EAR

✓

PCERA

CAN

CAP

CAPE

CAR

CARE

CLAP

EACH

EAR

PCERA

PCERA CPERA EPCRA PECRA… (5! = 120)

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN CAP

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN CAP CAPE

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN CAP CAPE CAR

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN CAP CAPE CAR CARE

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN CAP CAPE CAR CARE CLAP

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN CAP CAPE CAR CARE CLAP EACH

PCERA

PCERA CPERA EPCRA PECRA CEPRA ECPRA RCPEA CRPEA PRCEA RPCEA
CPREA PCREA PERCA EPRCA RPECA PRECA ERPCA REPCA RECPA ERCPA
CREPA RCEPA ECRPA CERPA AERPC EARPC RAEPC AREPC ERAPC REAPC
PEARC EPARC APERC PAERC EAPRC AEPRC ARPEC RAPEC PAREC APREC
RPAEC PRAEC PREAC RPEAC EPRAC PERAC REPAC ERPAC CRPAE RCPAE
PCRAE CPRAE RPCAE PRCAE ARCPE RACPE CARPE ACRPE RCAPE CRAPE
CPARE PCARE ACPRE CAPRE PACRE APCRE APRCE PARCE RAPCE ARPCE
PRACE RPACE EPACR PEACR AEPCR EAPCR PAECR APECR CPEAR PCEAR
ECPAR CEPAR PECAR EPCAR EACPR AECPR CEAPR ECAPR ACEPR CAEPR
CAPER ACPER PCAER CPAER APCER PACER RACEP ARCEP CRAEP RCAEP
ACREP CAREP EARCP AERCP REACP ERACP ARECP RAECP RCEAP CREAP
ERCAP RECAP CERAP ECRAP ECARP CEARP AECRP EACRP CAERP ACERP

CAN CAP CAPE CAR CARE CLAP EACH EAR

PCERA

PCERA CPERA EPCRA PECRA… (5! = 120)

NPCERA

NPCERA PNCERA CNPERA NCPERA… (6! = 720)

NPCERAH

NPCERAH PNCERAH CNPERAH NCPERAH… (7! = 5,040)

O(k): k!

Factorial

4! = 24

5! = 120

6! = 720

7! = 5,040

8! = 40,320

9! = 362,880

10! = 3,628,800

11! = 39,916,800

12! = 479,001,600

13! = 6,227,020,800

14! = 87,178,291,200

15! = 1,307,674,368,000

The worst part is, we can’t just do this permutation once.

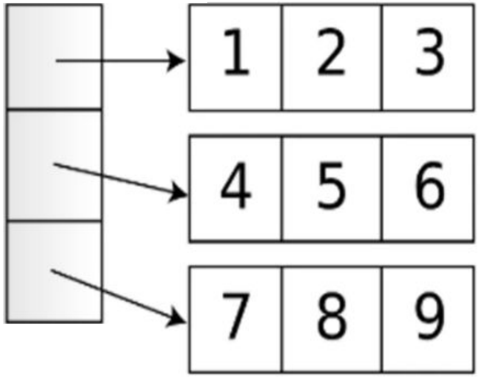
Above 11 or 12 characters, that’d be too many strings to hold in memory all at once.

So we’re going to have to re-permute input for each word we check. O(n * k!)

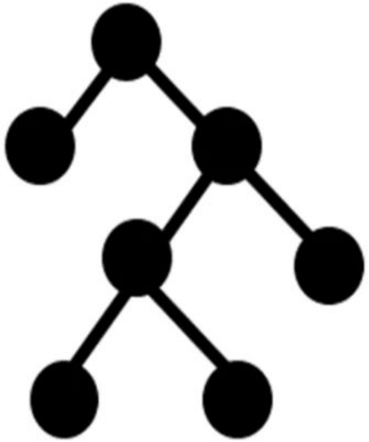
Actually, it’s not that bad. (but it’s still bad)

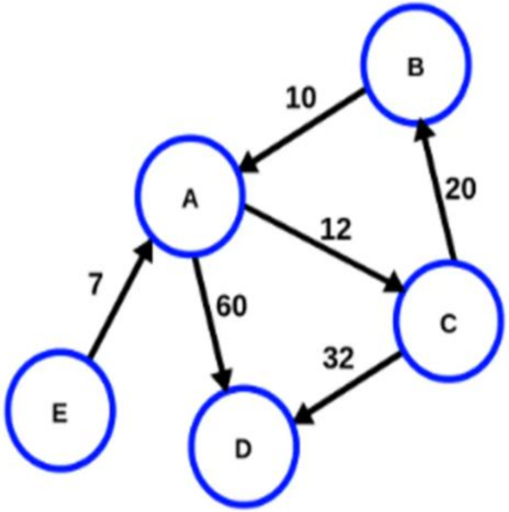
Permutation of the input can stop once the length of the word is reached.

Also, we can abandon any partial permutation result that doesn’t match the beginning of each word.



1	2	3
4	5	6
7	8	9

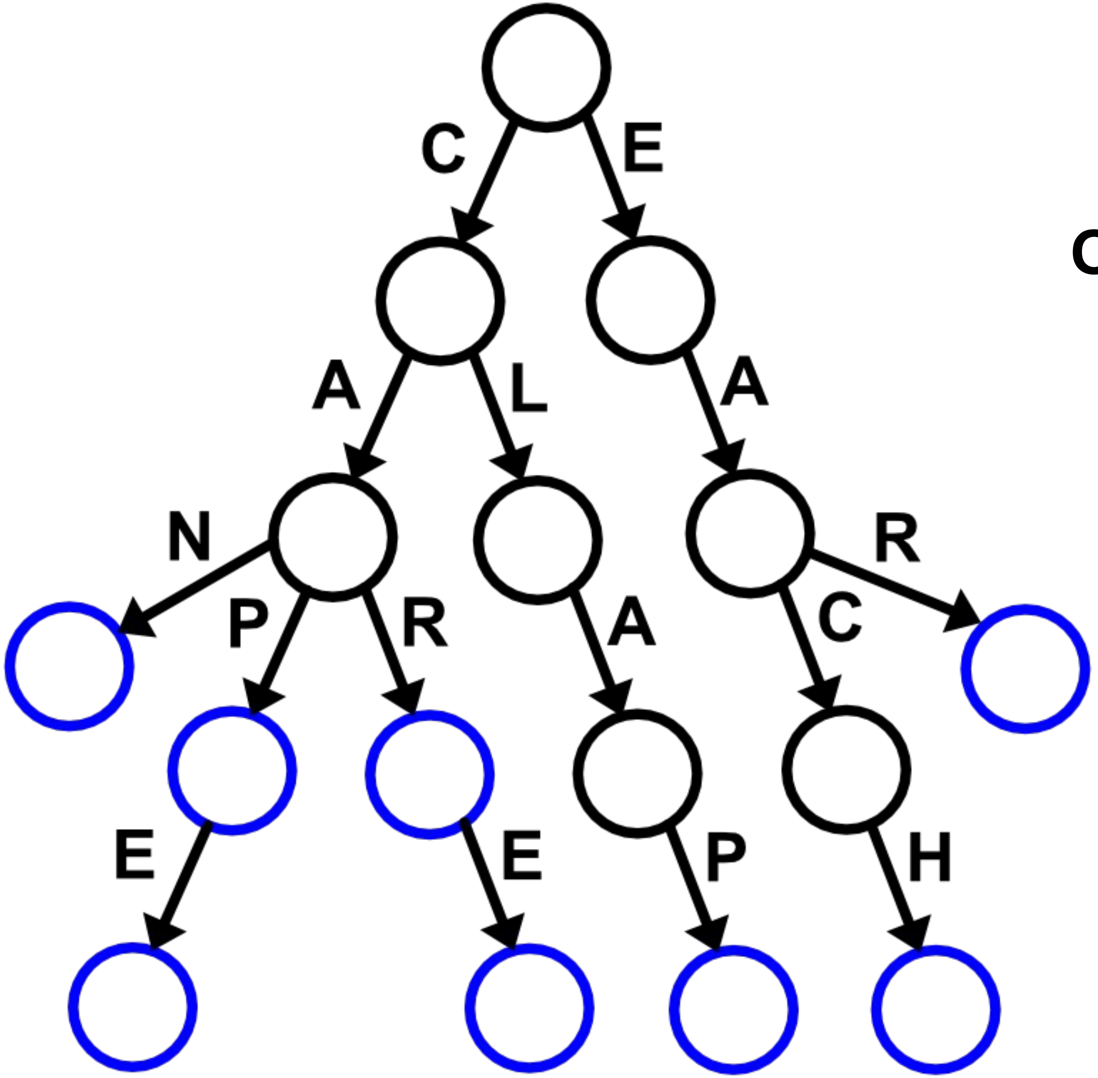




Data Structures

Trie

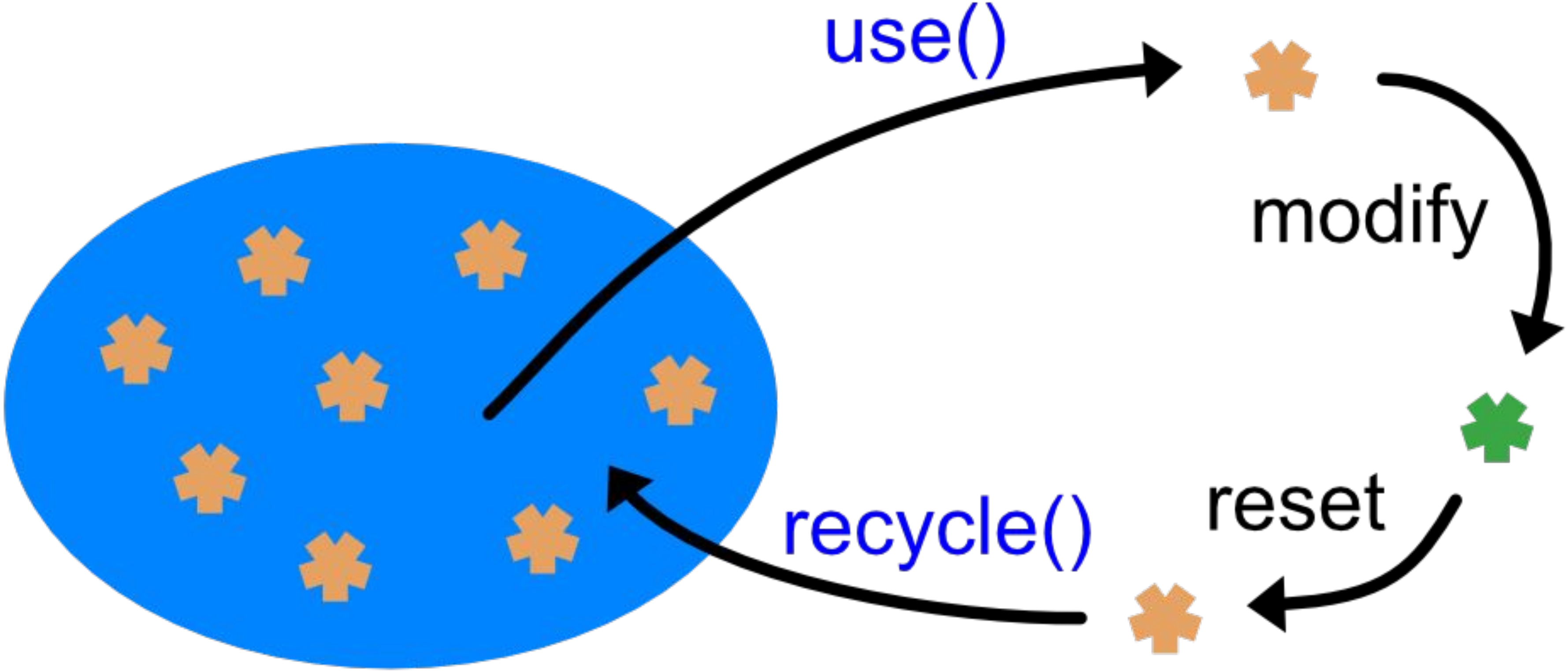
CAN
CAP
CAPE
CAR
CARE
CLAP
EACH
EAR



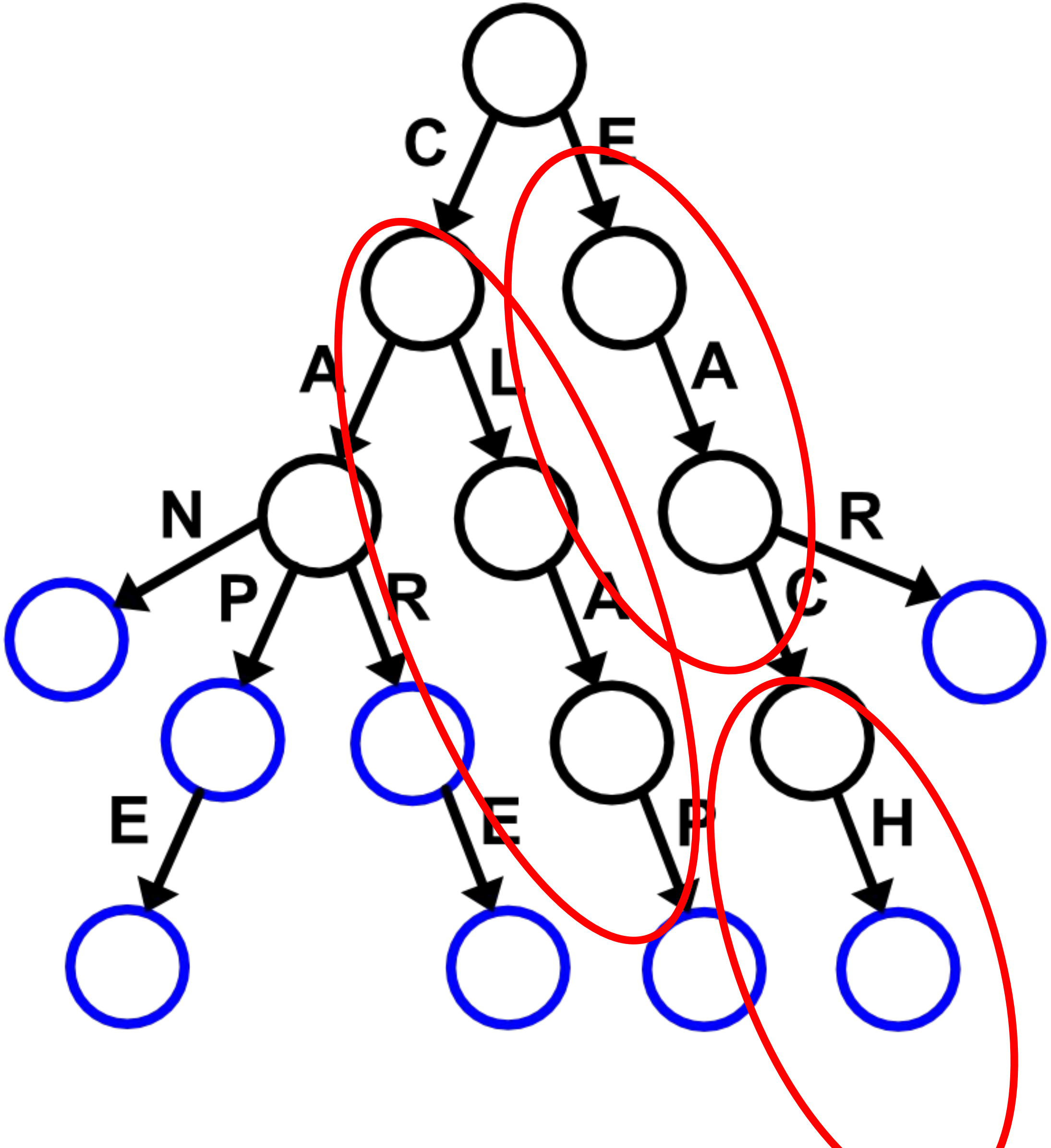
O(k): k * m

Linear

Minimizing Garbage Collection



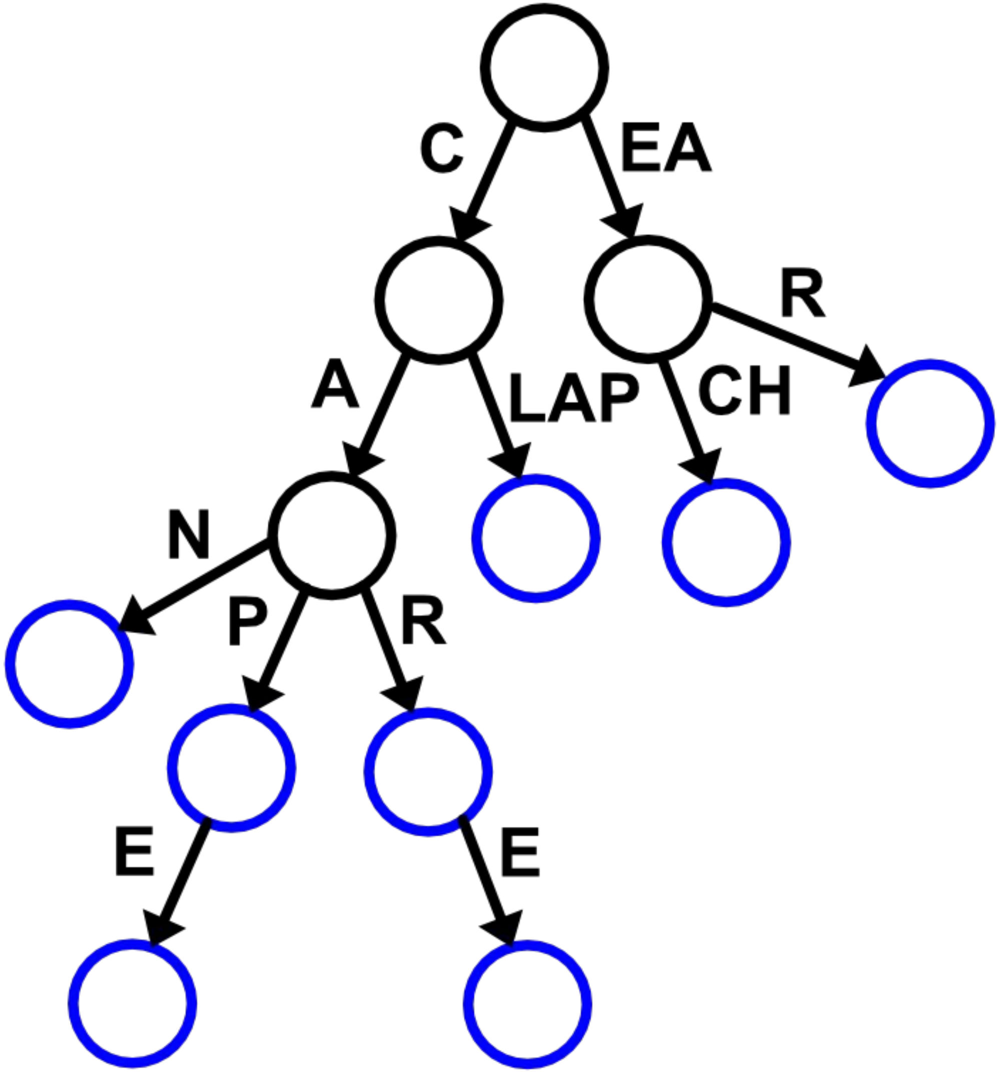
Object Pool



C
E
A
L
A
N
P
R
A
C
R
E
E
P
H

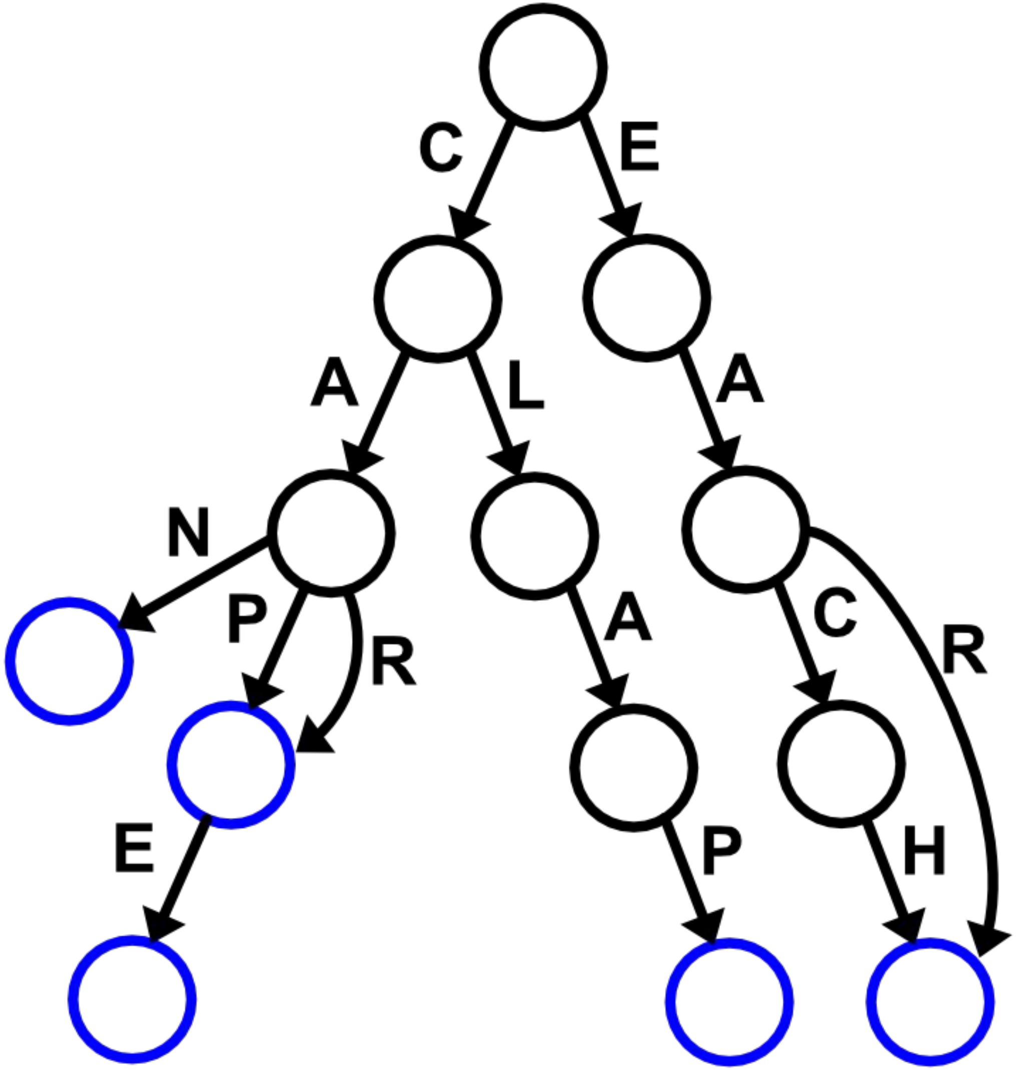
Radix Tree

CAN
CAP
CAPE
CAR
CARE
CLAP
EACH
EAR

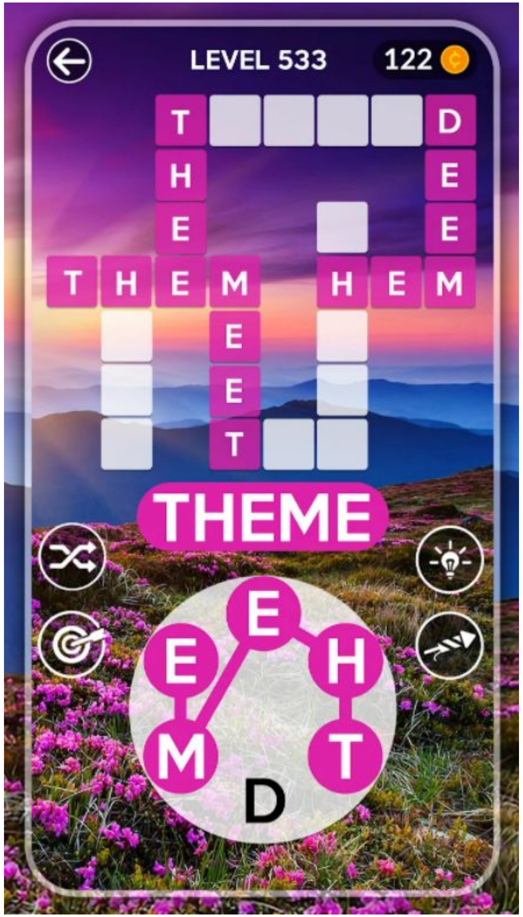


CAN
CAP
CAPE
CAR
CARE
CLAP
EACH
EAR

DAFSA / DAWG



_H_ _ _



explain

e xplain

xe plain

pxe lain

lpxe ain

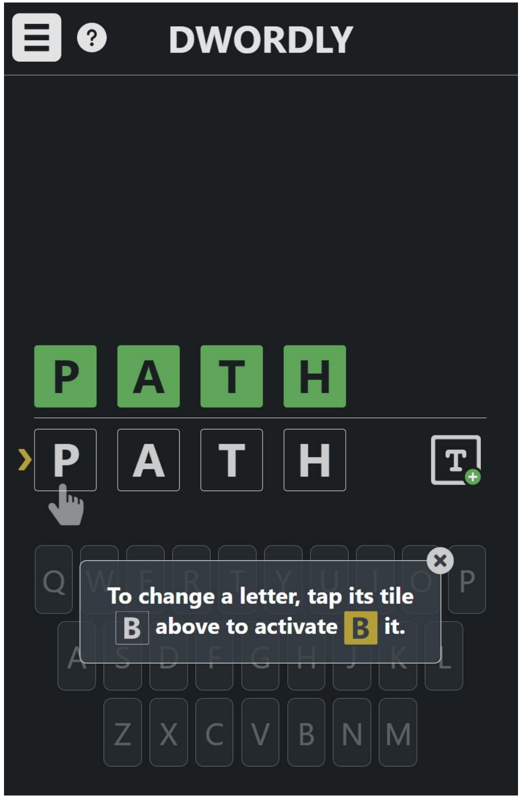
alpxe in

ialpxe n

nialpxe

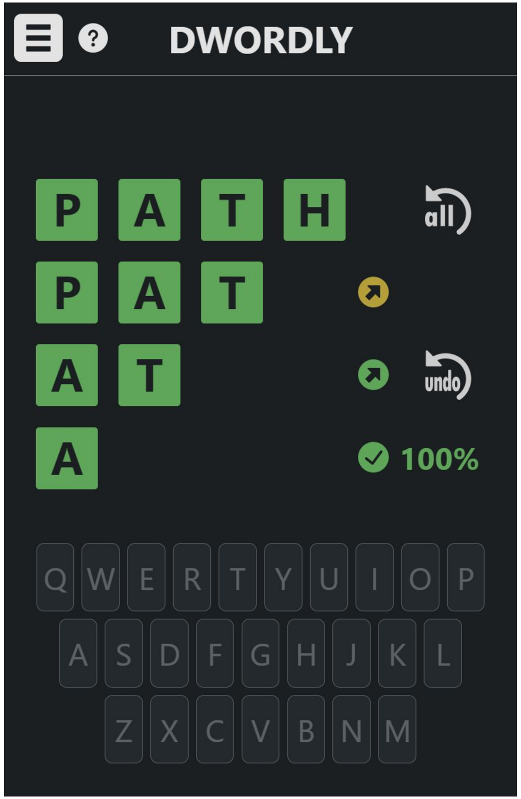
GADDAG

https://en.wikipedia.org/wiki/GADDAG



To change a letter, tap its tile B above to activate B it.

Bonus: Dwordly



https://dwordly.fun

The End