

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



4



Recursion



3



Fuzzy?



2



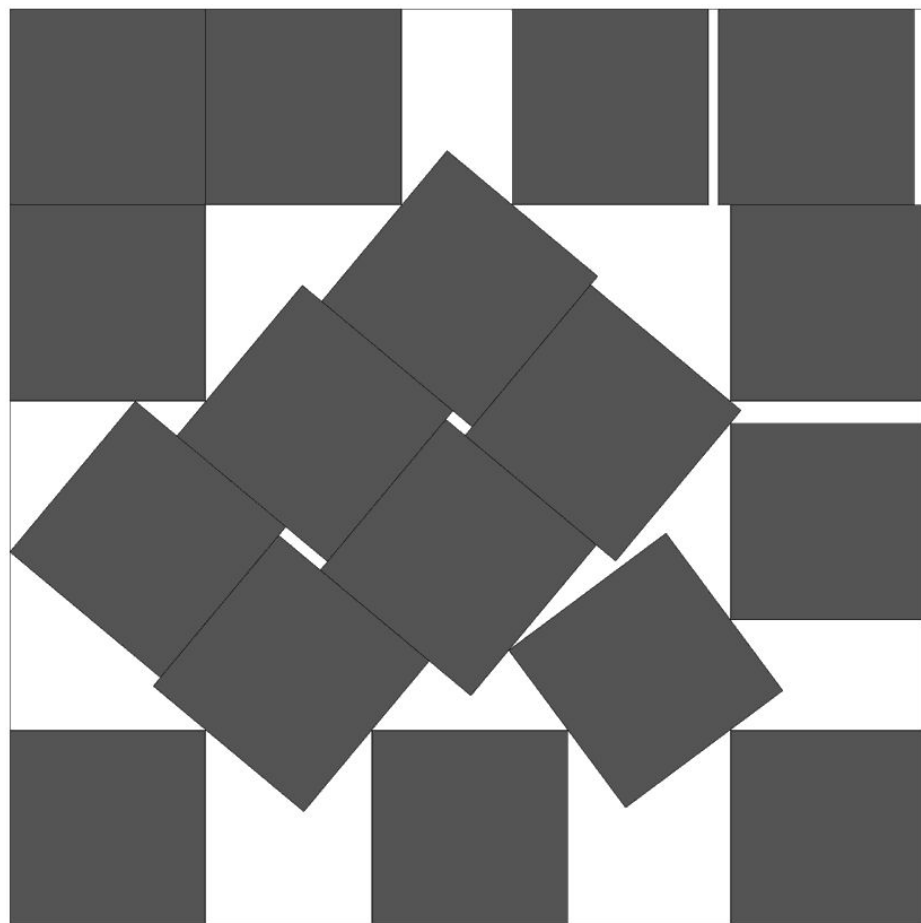
Warm-ups

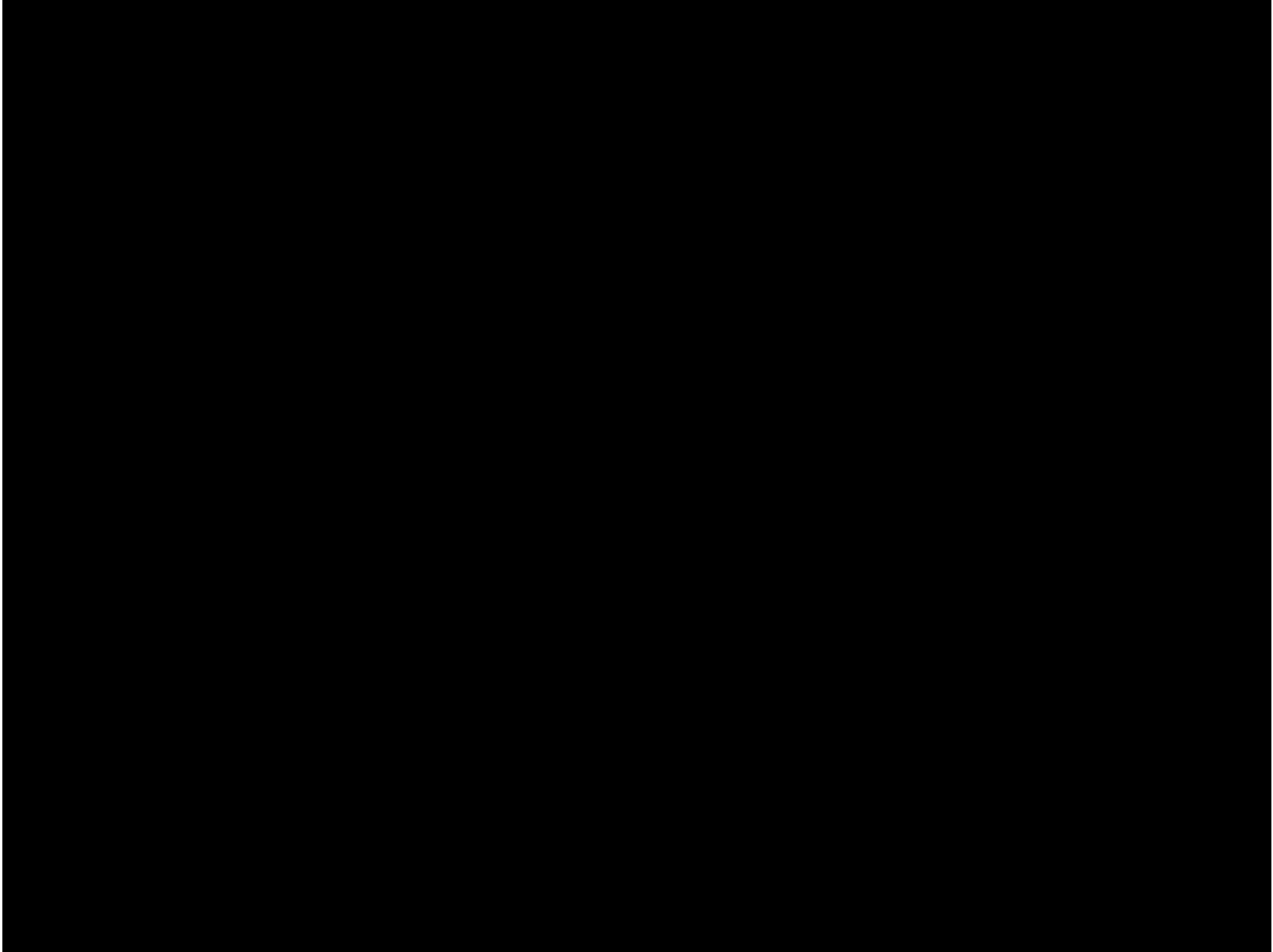
binary-to-decimal.js

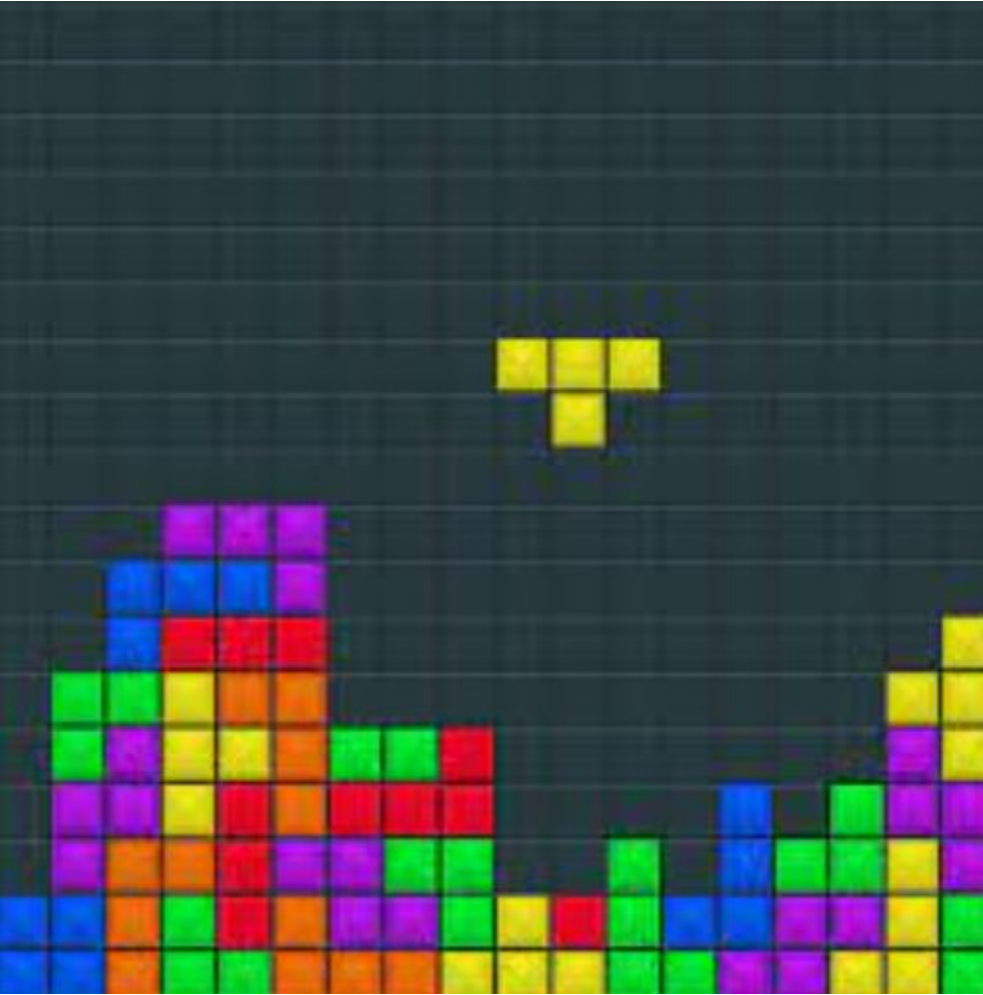
```
1  function binaryToDecimalConverter(binaryNumber) {
2    var decimal = 0n;
3    for (let [idx,digit] of Object.entries(binaryNumber)) {
4      let power = binaryNumber.length - idx - 1;
5      decimal += BigInt(digit) * (2n ** BigInt(power));
6    }
7    return decimal;
8  }
9
10  binaryToDecimalConverter("111110011101100"); // 31980n
```

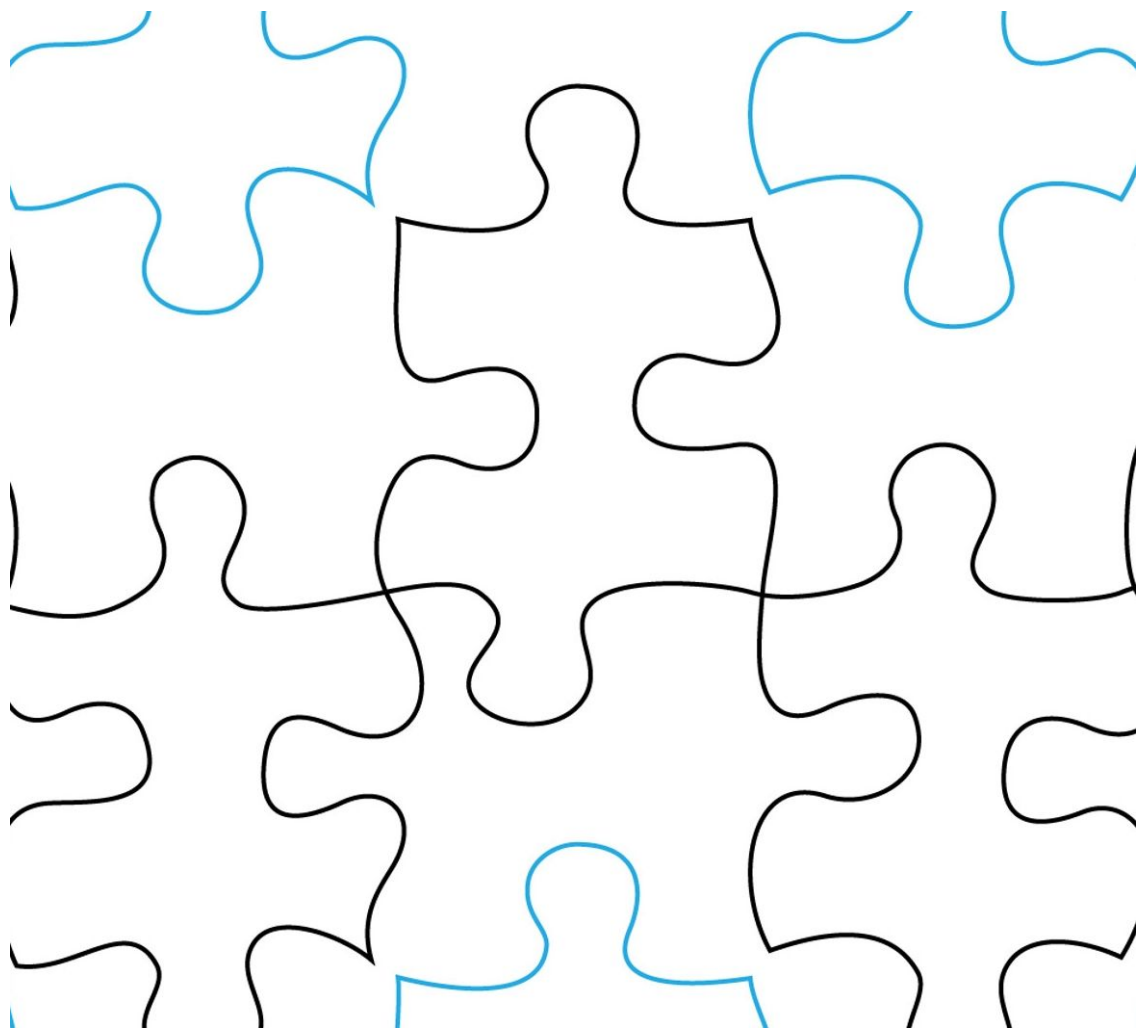
decimal-to-binary.js

```
1  function decimalToBinaryConverter(base10number) {
2    var binary = "";
3    var next = base10number;
4    while (next > 0n) {
5      let remainder = next % 2n;
6      binary = remainder + binary;
7      next = next / 2n;
8    }
9    return binary;
10 }
11
12 decimalToBinaryConverter(31980n); // 111110011101100
```











2-BFS.js

```
1 // iterative breadth-first solution
2
3 function findMaxRegionArea(matr) {
4     const ROW_LEN = matr[0].length;
5     var area = 0;
6     var maxArea = 0;
7     var visited = new Set();
8
9     for (let [ rowIdx, row ] of matr.entries()) {
10         for (let [ colIdx, cell ] of row.entries()) {
11             if (
12                 !visited.has((rowIdx * ROW_LEN) + colIdx) &&
13                 cell == 1
14             ) {
15                 // initialize a breadth-first queue
16                 let toVisit = [ rowIdx, colIdx ];
17
18                 while (toVisit.length > 0) {
19                     // pull to-visit cell coordinates off the queue
20                     let [ visitedRowIdx, visitedColIdx ] = toVisit.shift();
21
22                     // compute the cell-index
23                     let visitedCellIdx = (visitedRowIdx * ROW_LEN) + visitedColIdx;
24
25                     // have we not visited this cell yet?
26                     if (!visited.has(visitedCellIdx)) {
27                         visited.add(visitedCellIdx);
28                         area++;
29
30                         // inspect current row and two (possibly) adjacent rows
31                         for (let rowDelta of [ -1, 0, 1 ]) {
32                             // inspect current column and two (possibly) adjacent columns
33                             for (let colDelta of [ -1, 0, 1 ]) {
34                                 // avoid re-considering the current cell
35                                 if (!(rowDelta == 0 && colDelta == 0)) {
36                                     let toVisitRowIdx = (visitedRowIdx + rowDelta);
37                                     let toVisitColIdx = (visitedColIdx + colDelta);
38
39                                     // is the cell actually in bounds of the matrix,
40                                     // and is has a `1` in it, and is not already
41                                     // a cell we've visited/counted?
```


Exercise: Periodic Table Speller

[workshop repo](#)

PERIODIC TABLE OF ELEMENTS

1 IA																	18 VIIIA
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	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 IB	12 IIB	13 Al Aluminum 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 69.723	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 Cesium 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 [261]	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 Neodymium 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 231.03688	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]			

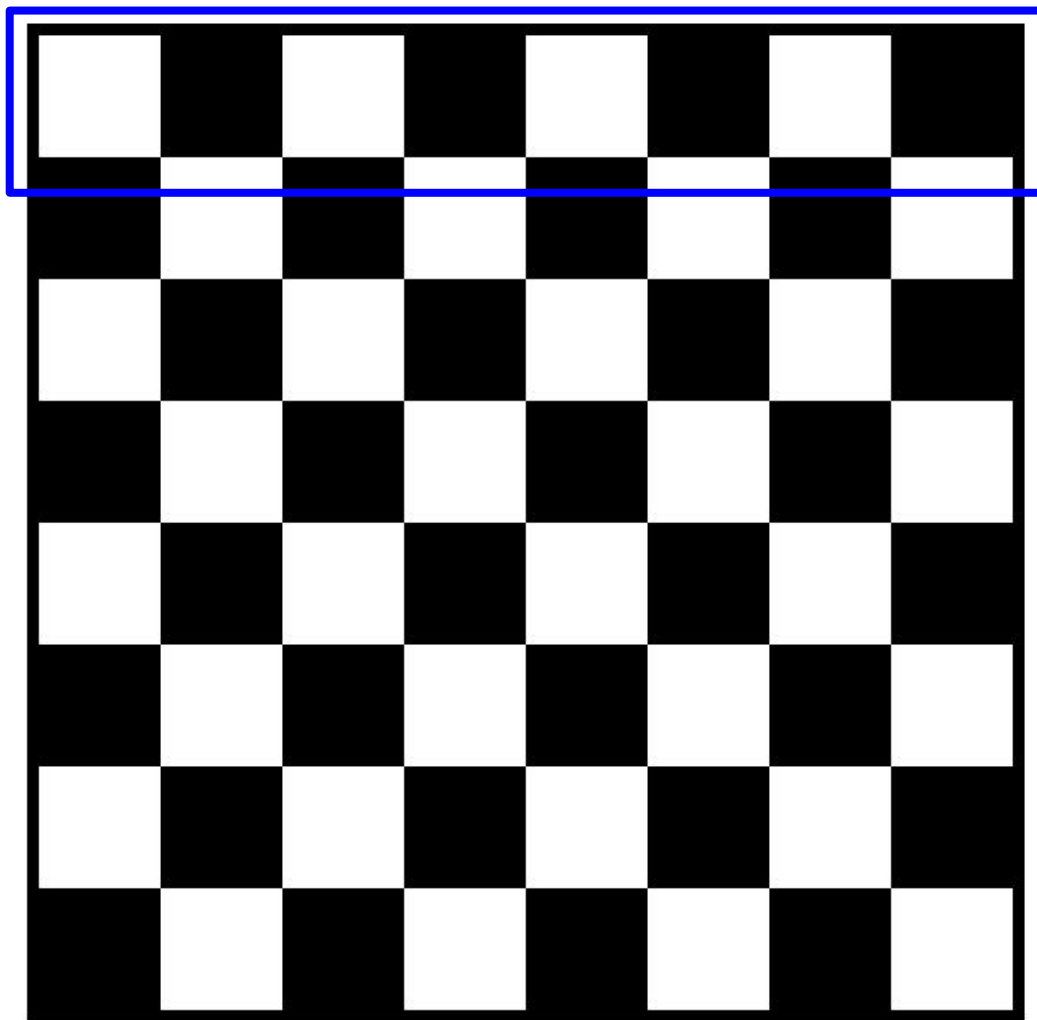
Word:

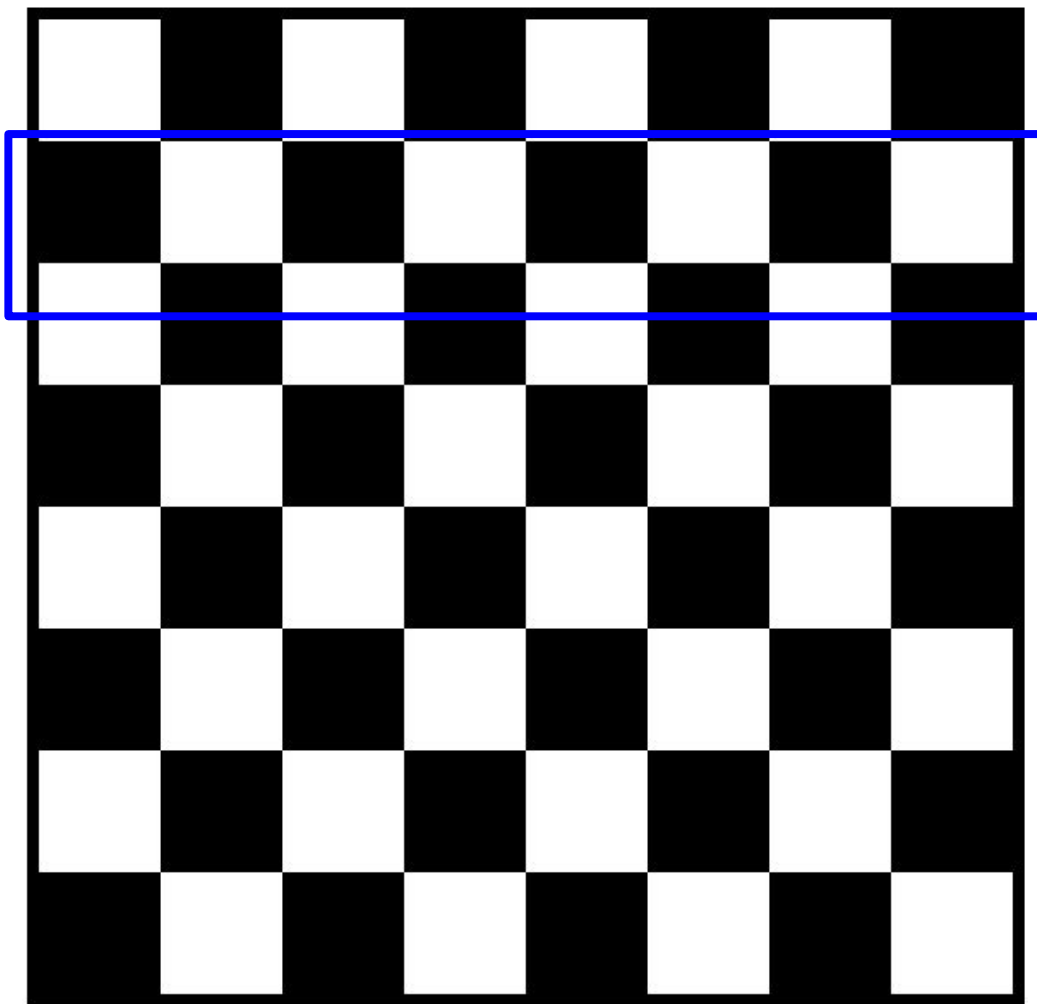
4 Be Beryllium	20 Ca Calcium	92 U Uranium	34 Se Selenium
-----------------------------	----------------------------	---------------------------	-----------------------------

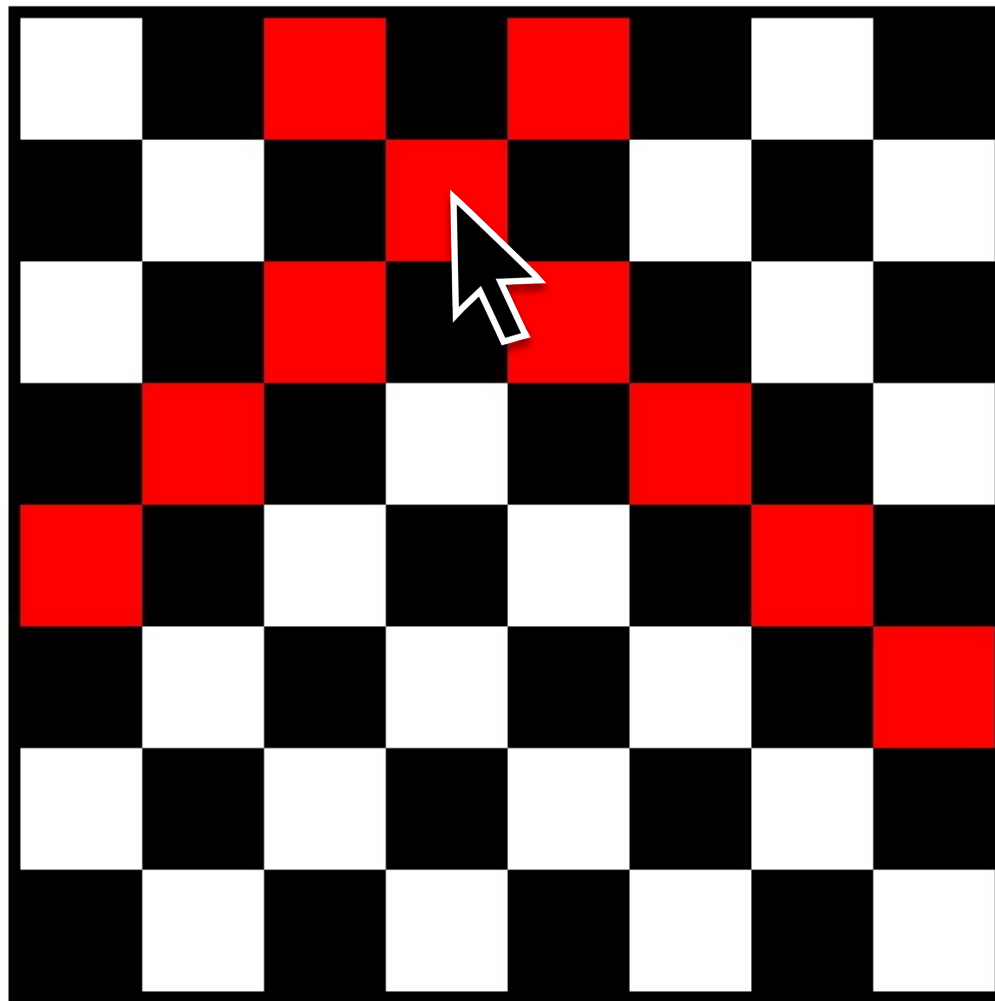
Periodic Table Speller

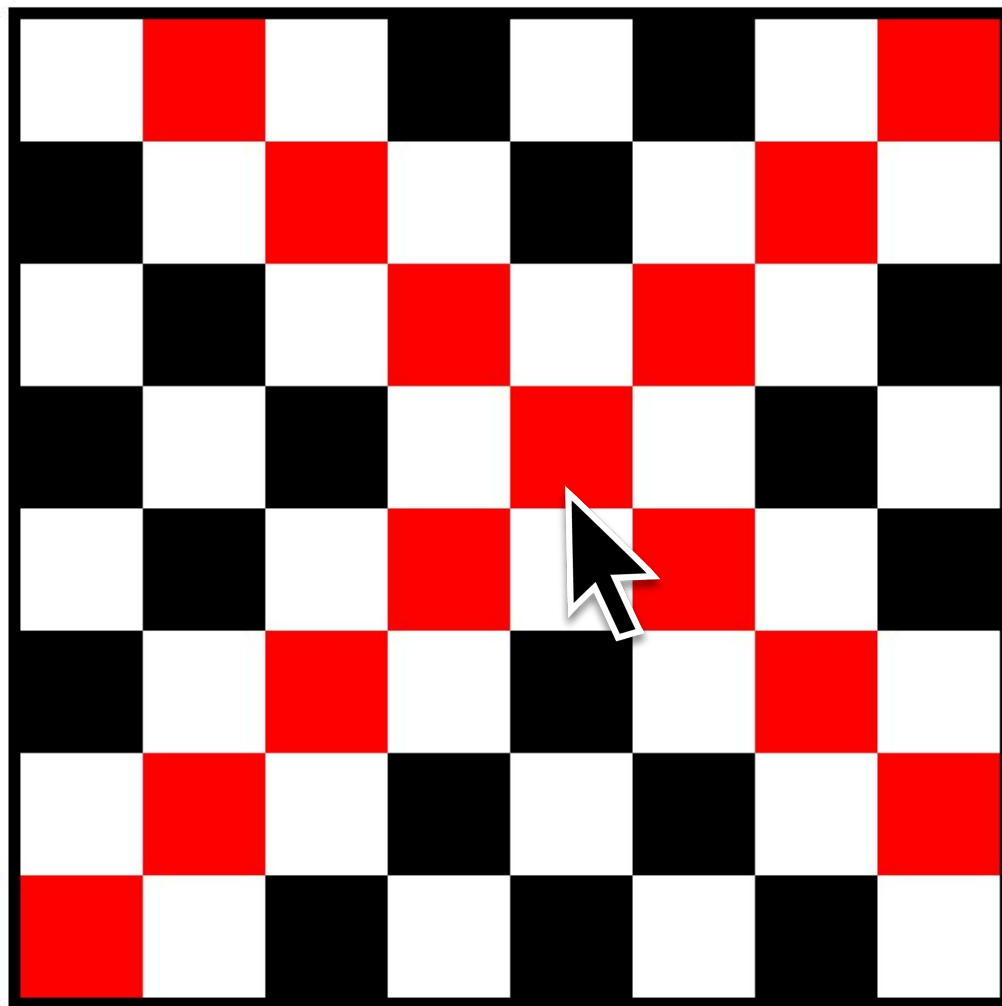
Exercise: Chessboard Diagonals

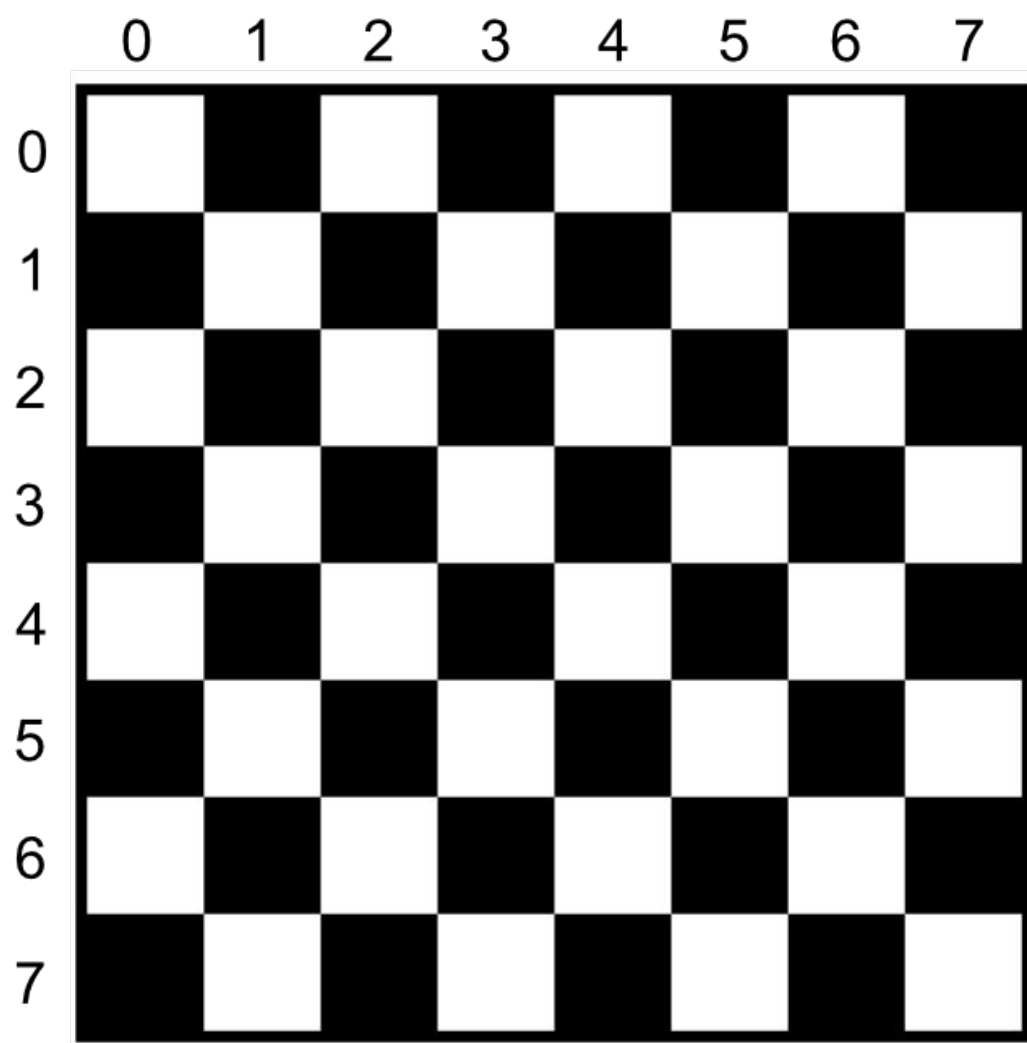
[workshop repo](#)



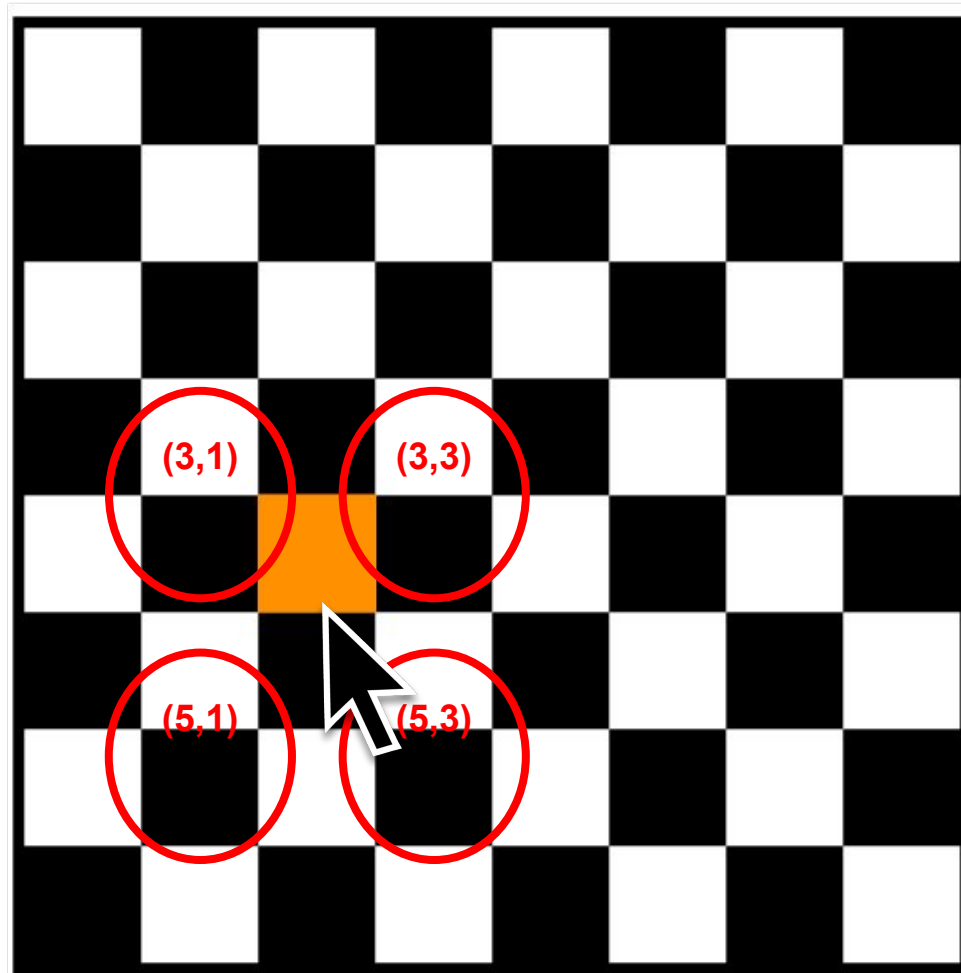








2



4

(4,2)

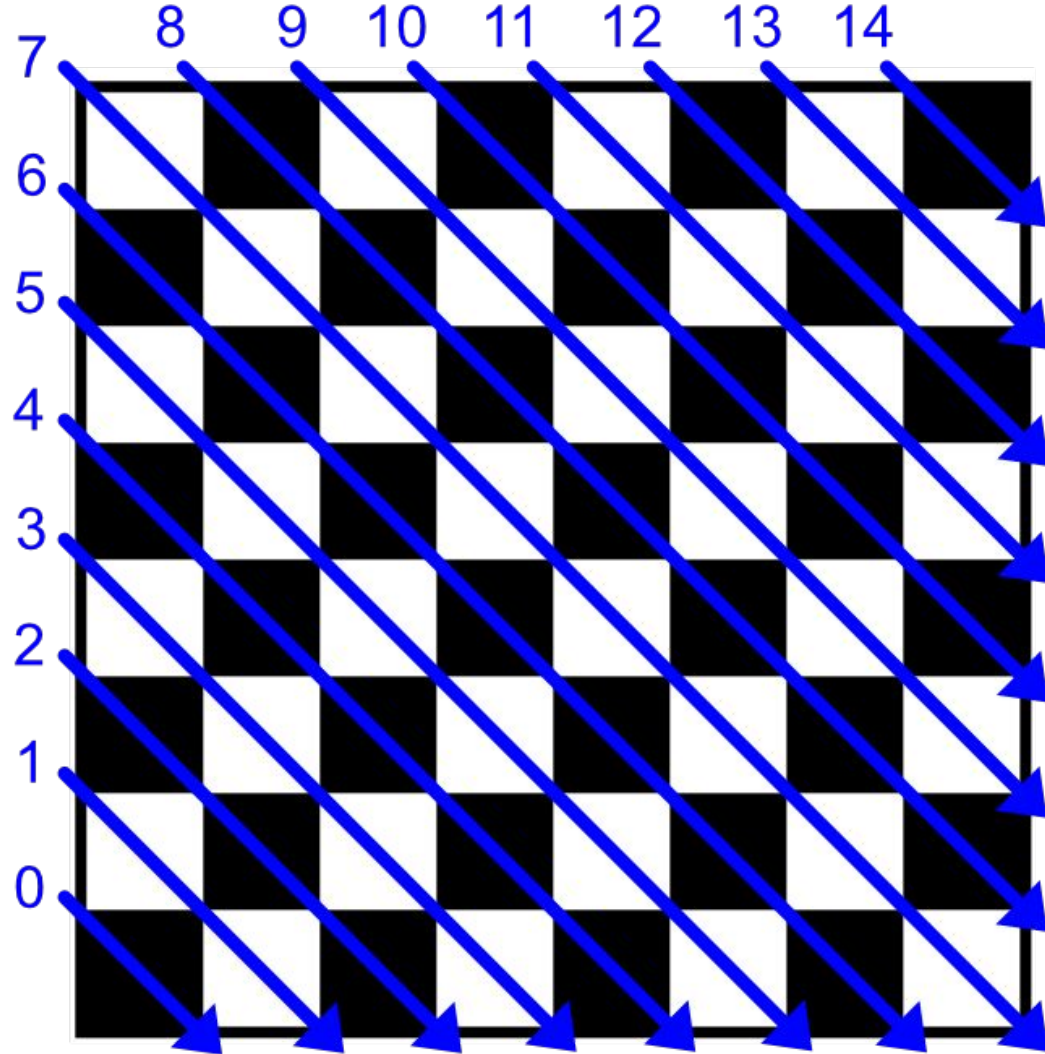
(3,1)

(3,3)

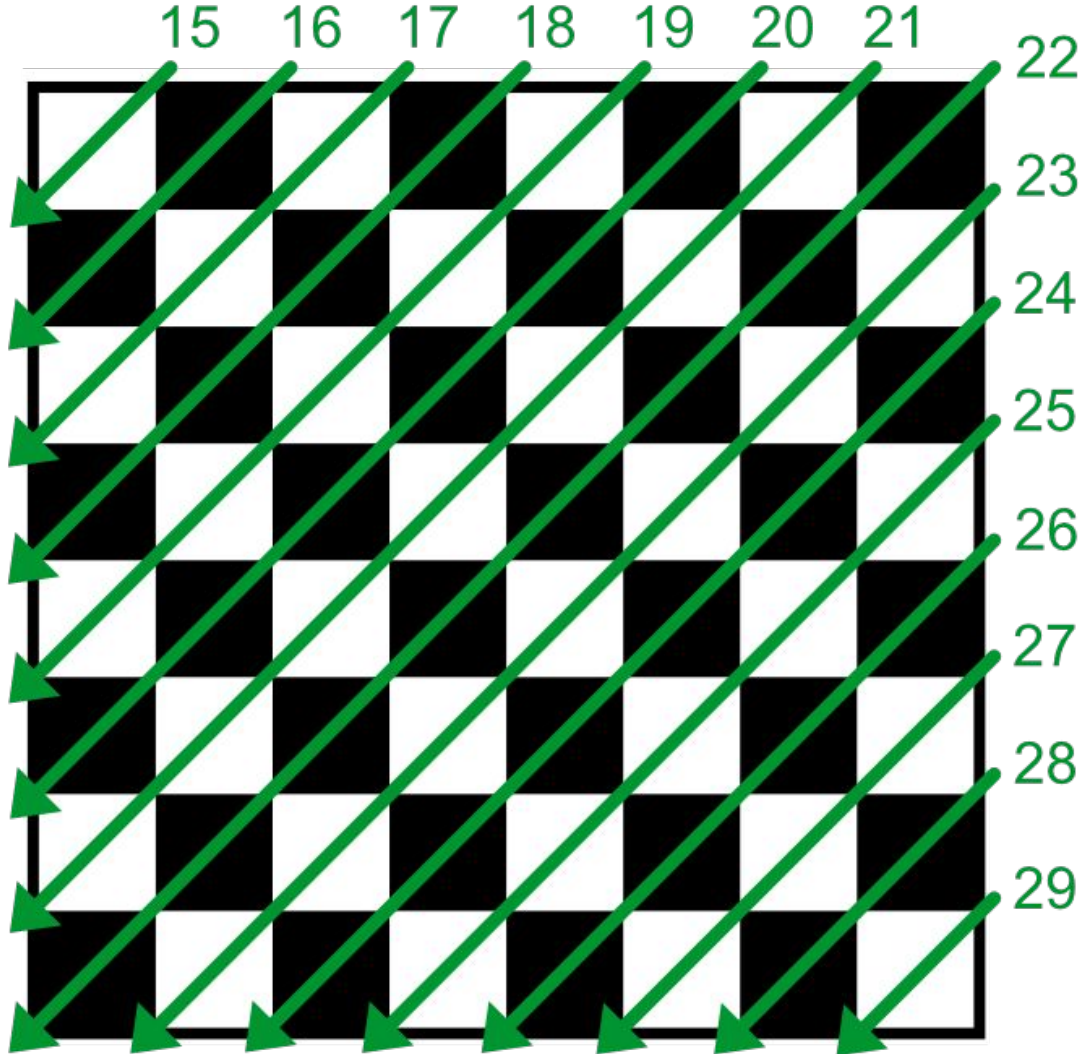
(5,1)

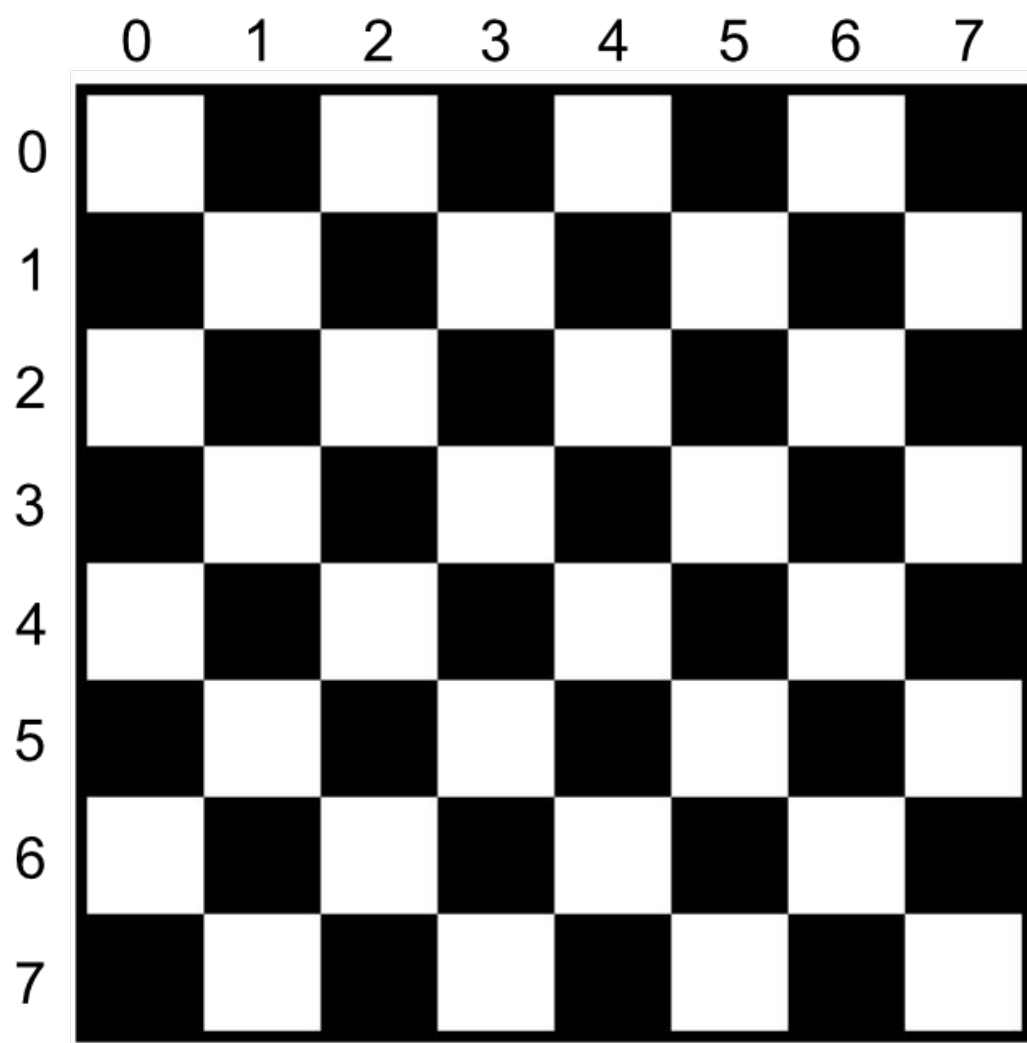
(5,3)

**Major
Diagonals**

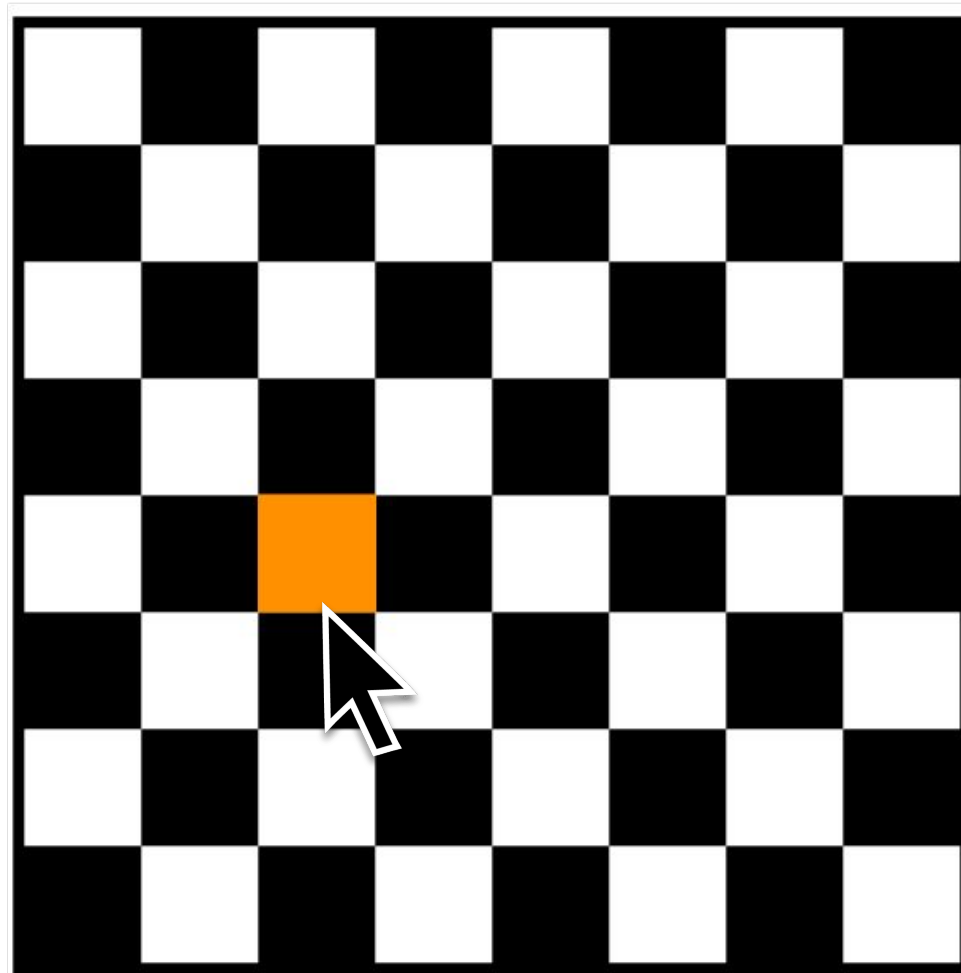


**Minor
Diagonals**





2



4

(4,2)

2

21

$$5 = 7 - (4 - 2)$$

$$21 = 15 + (4 + 2)$$

5

$$5 = 7 - (2 - 0)$$

$$5 = 7 - (3 - 1)$$

$$5 = 7 - (4 - 2)$$

$$5 = 7 - (5 - 3)$$

$$5 = 7 - (6 - 4)$$

$$5 = 7 - (7 - 5)$$

4

$$21 = 15 + (6 + 0)$$

$$21 = 15 + (5 + 1)$$

$$21 = 15 + (4 + 2)$$

$$21 = 15 + (3 + 3)$$

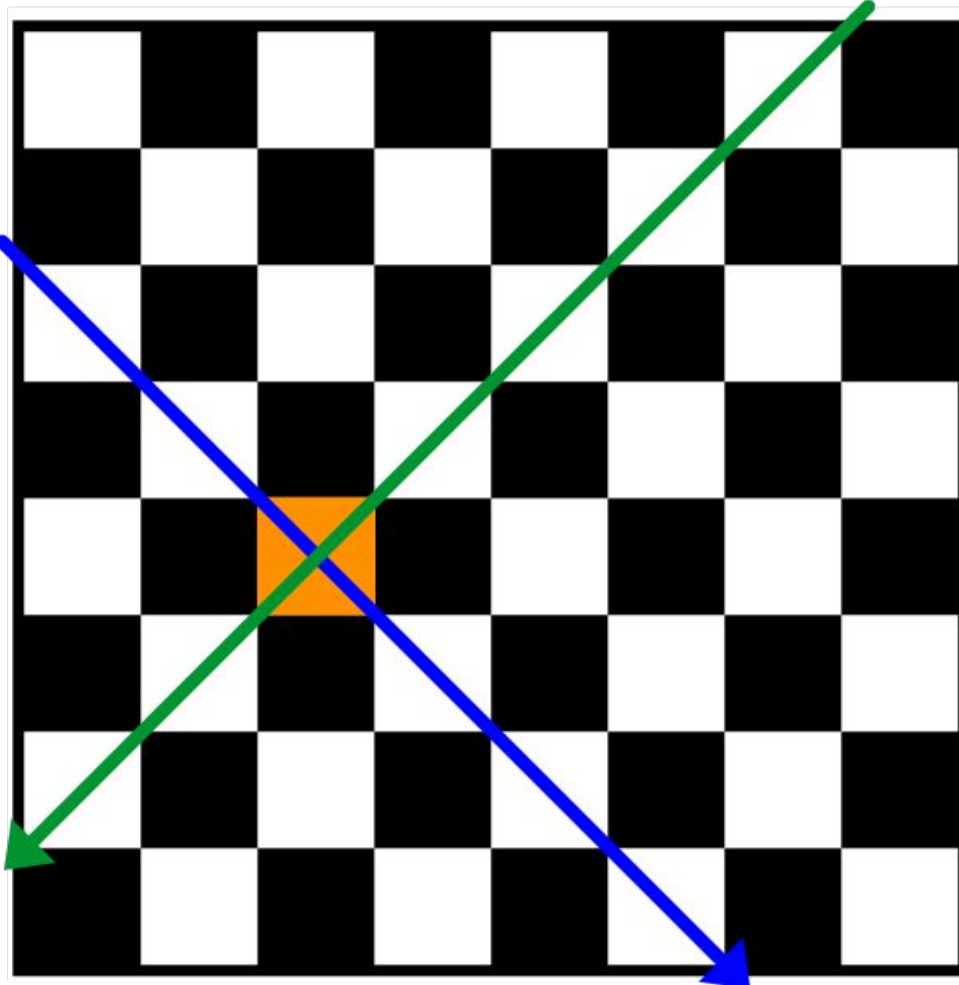
$$21 = 15 + (2 + 4)$$

$$21 = 15 + (1 + 5)$$

$$21 = 15 + (0 + 6)$$

$$7 - (\text{row} - \text{col})$$

$$15 + (\text{row} + \text{col})$$



Exercise: Knight's Dialer

[workshop repo](#)

1

2

3

4

5

6

7

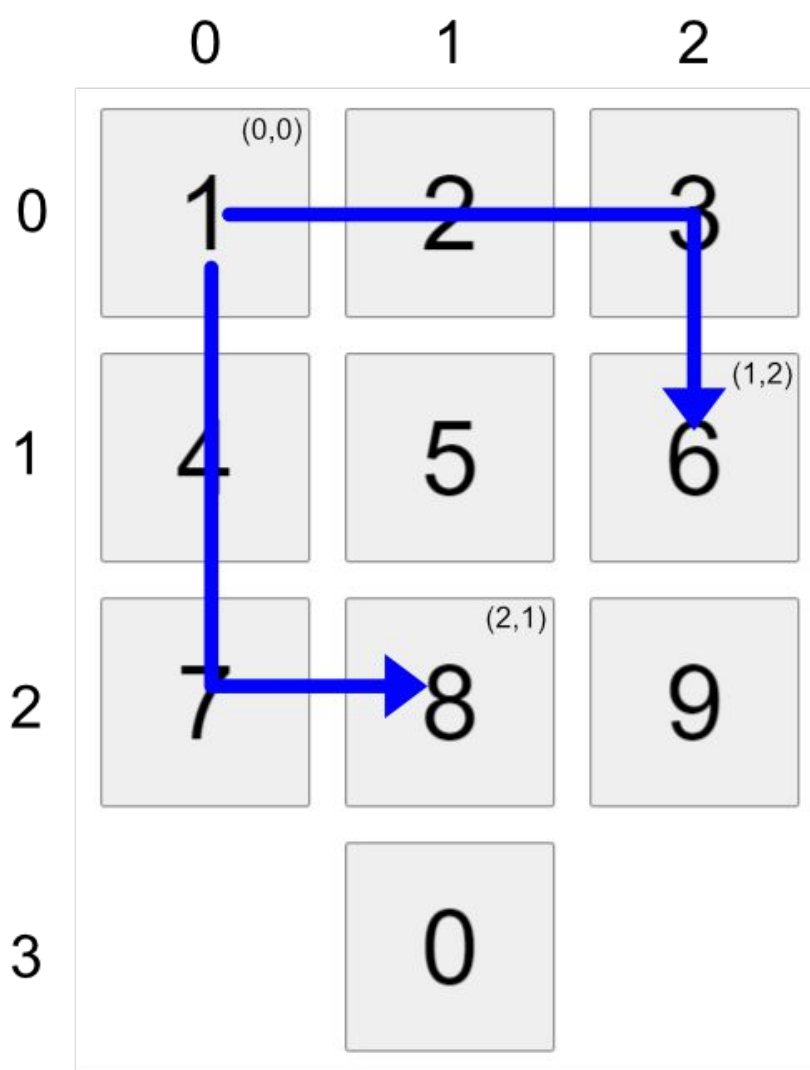
8

9

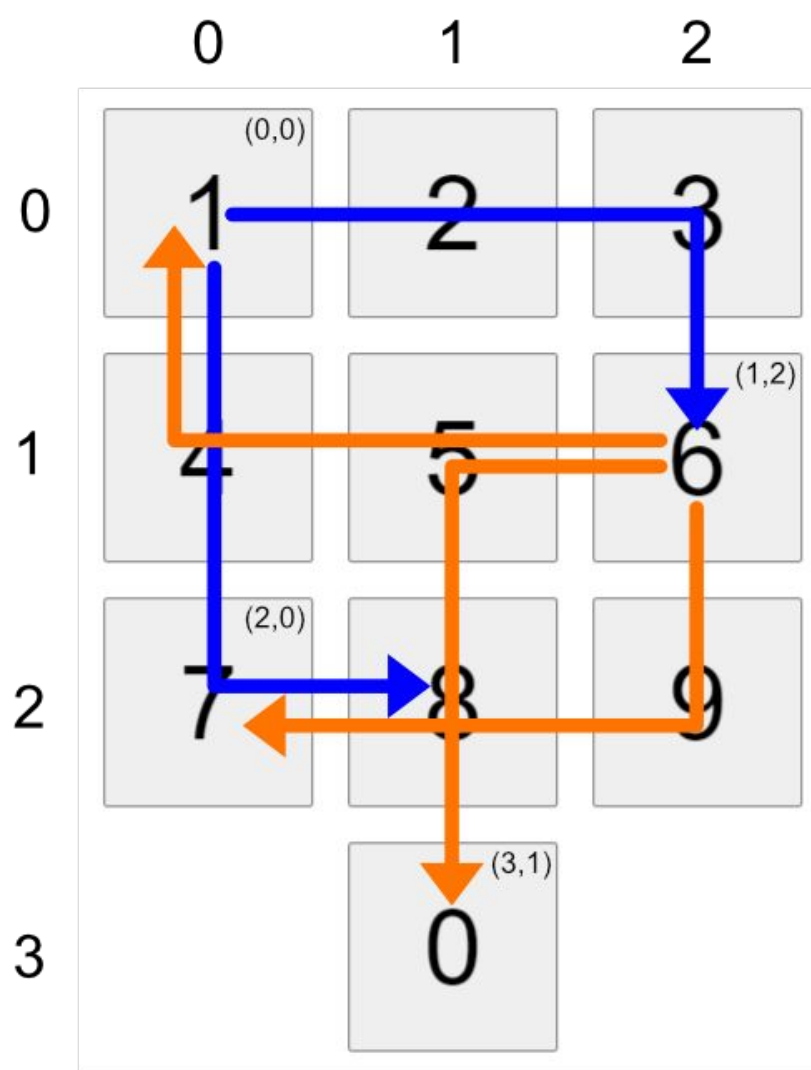
0

	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



$$\begin{aligned} \text{countPaths}(4,6) &= & 168 \\ &\text{countPaths}(3,5) + & 52 \\ &\text{countPaths}(9,5) + & 52 \\ &\text{countPaths}(0,5) & 64 \end{aligned}$$

Recursion

countPaths(3,5) = 52

countPaths(4,4) + 32

countPaths(8,4) 20

Recursion

Breadth-First

Depth-First

Traversal

$$f(4,6)$$

Breadth-First Traversal Iterative+Queue

$$f(4,6)$$

Depth-First Traversal

Recursion

$$O(n): \sim 2.222^n$$

$O(6): \sim 120 \text{ ops}$ $O(7): \sim 268 \text{ ops}$ $O(8): \sim 595 \text{ ops}$

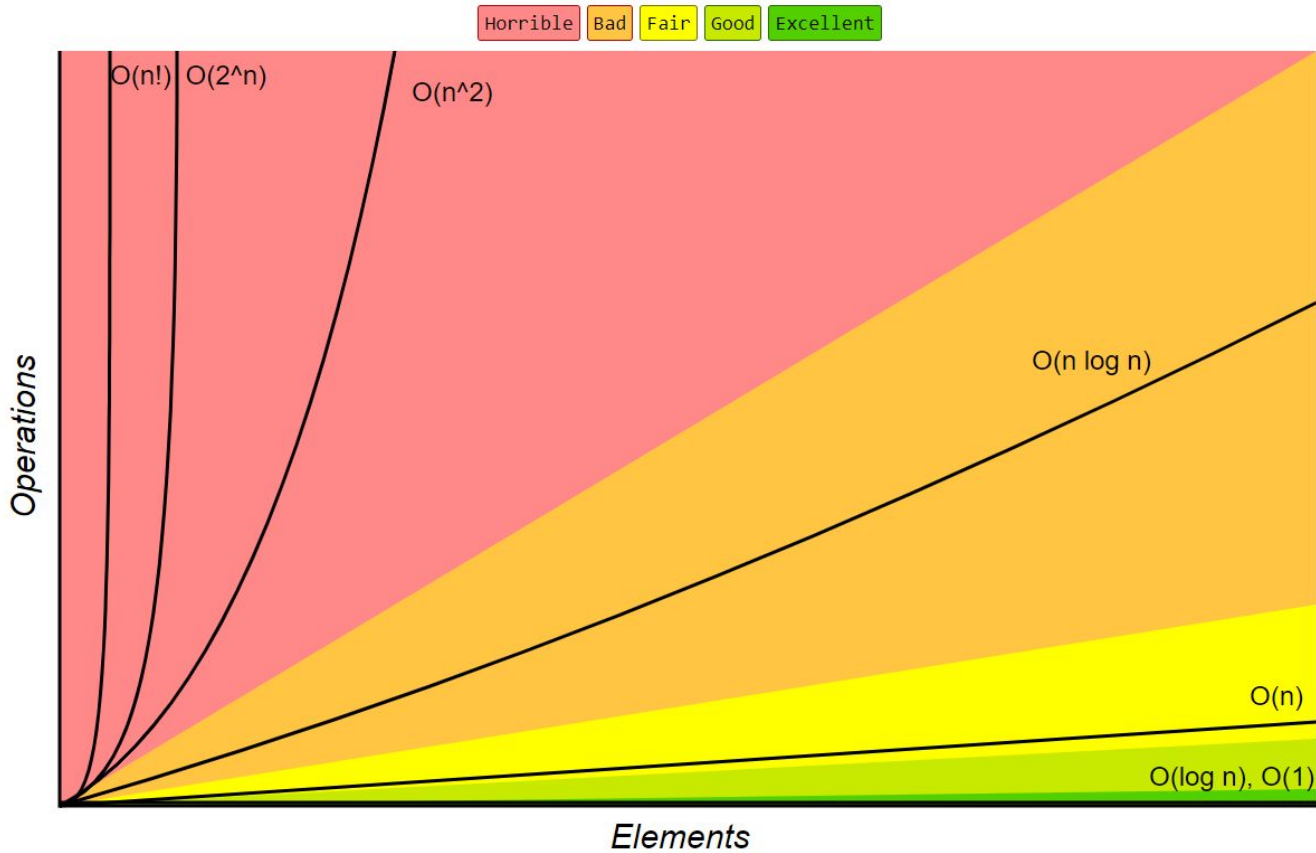
Exponential

Recursion

Factorial
Exponential
Polynomic
Linear * Logarithmic
Linear
Logarithmic
Constant

Big-O Complexity Chart

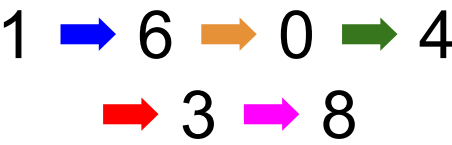
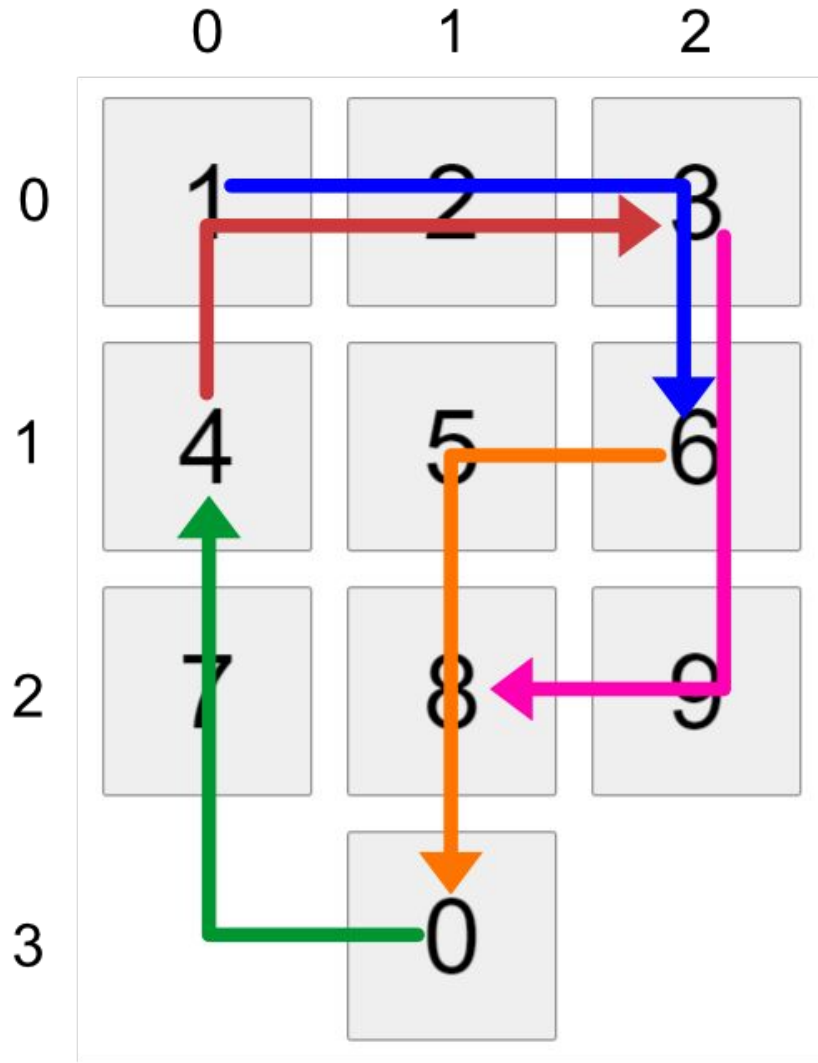
bigocheatsheet.com

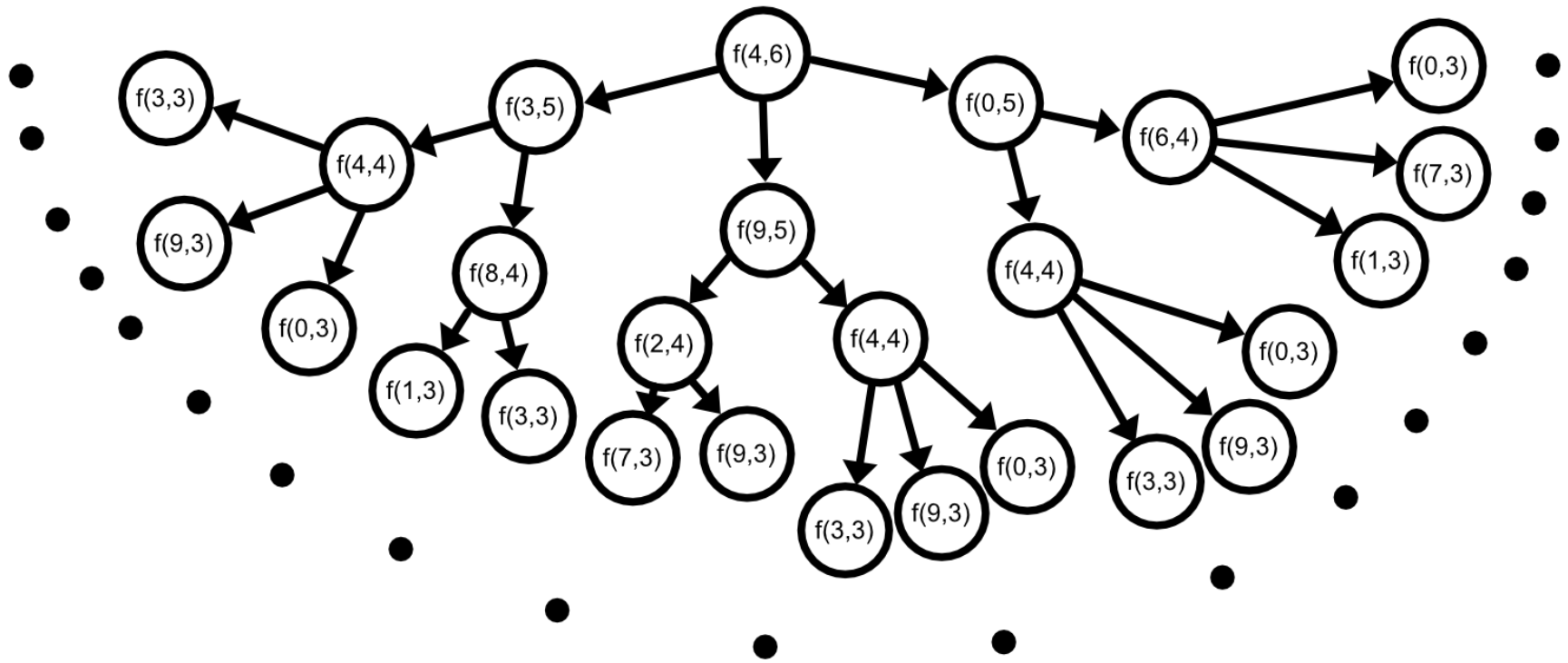


Acyclic
Paths

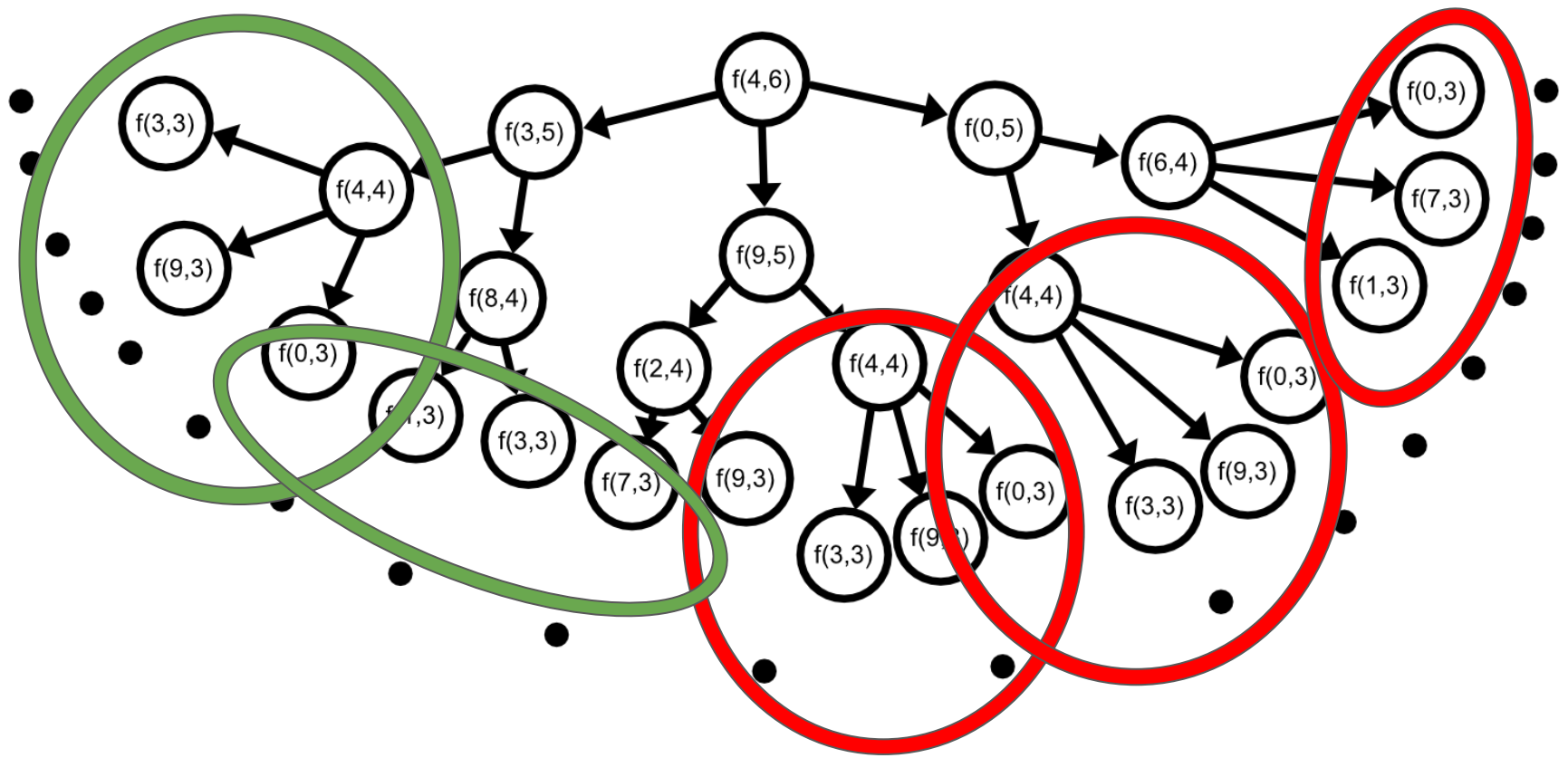
	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3		0	

Acyclic Paths

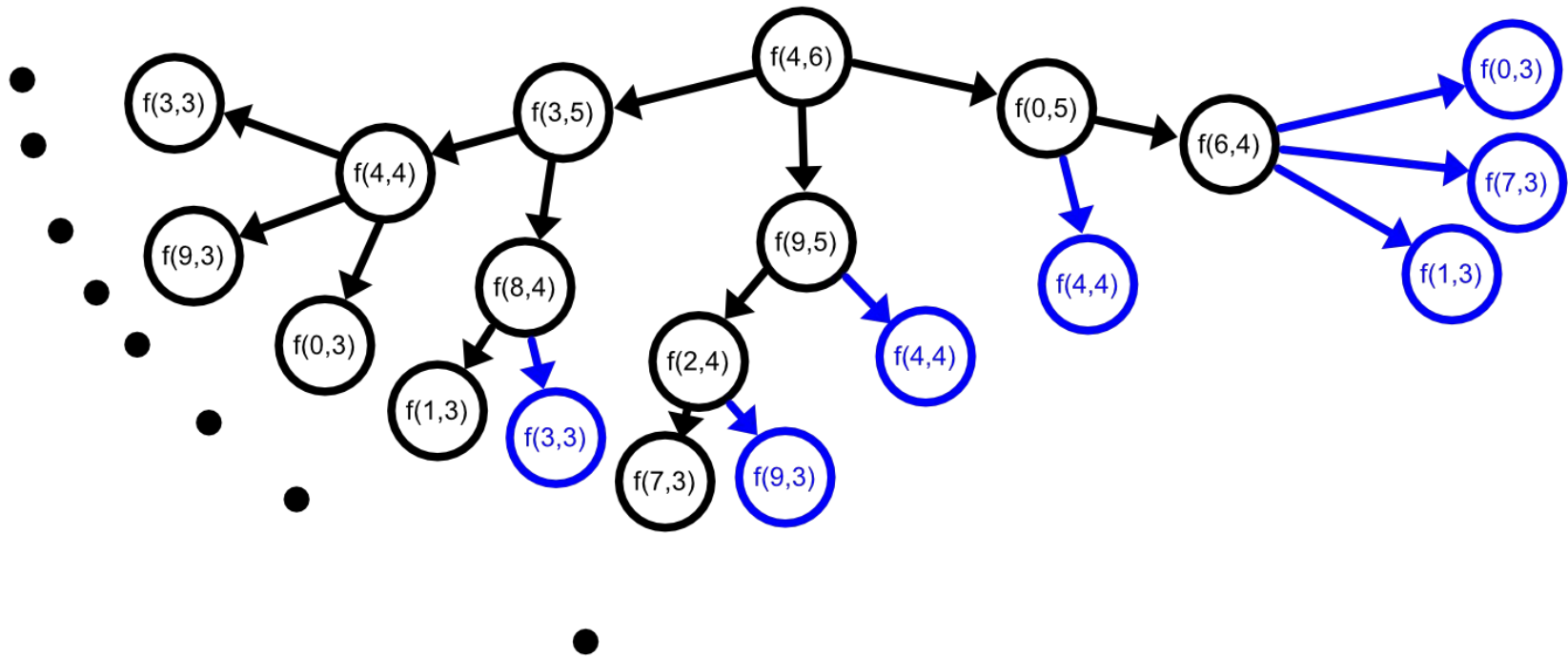




(Optimizing) Recursion



Recursion + Memoization



Recursion + Memoization

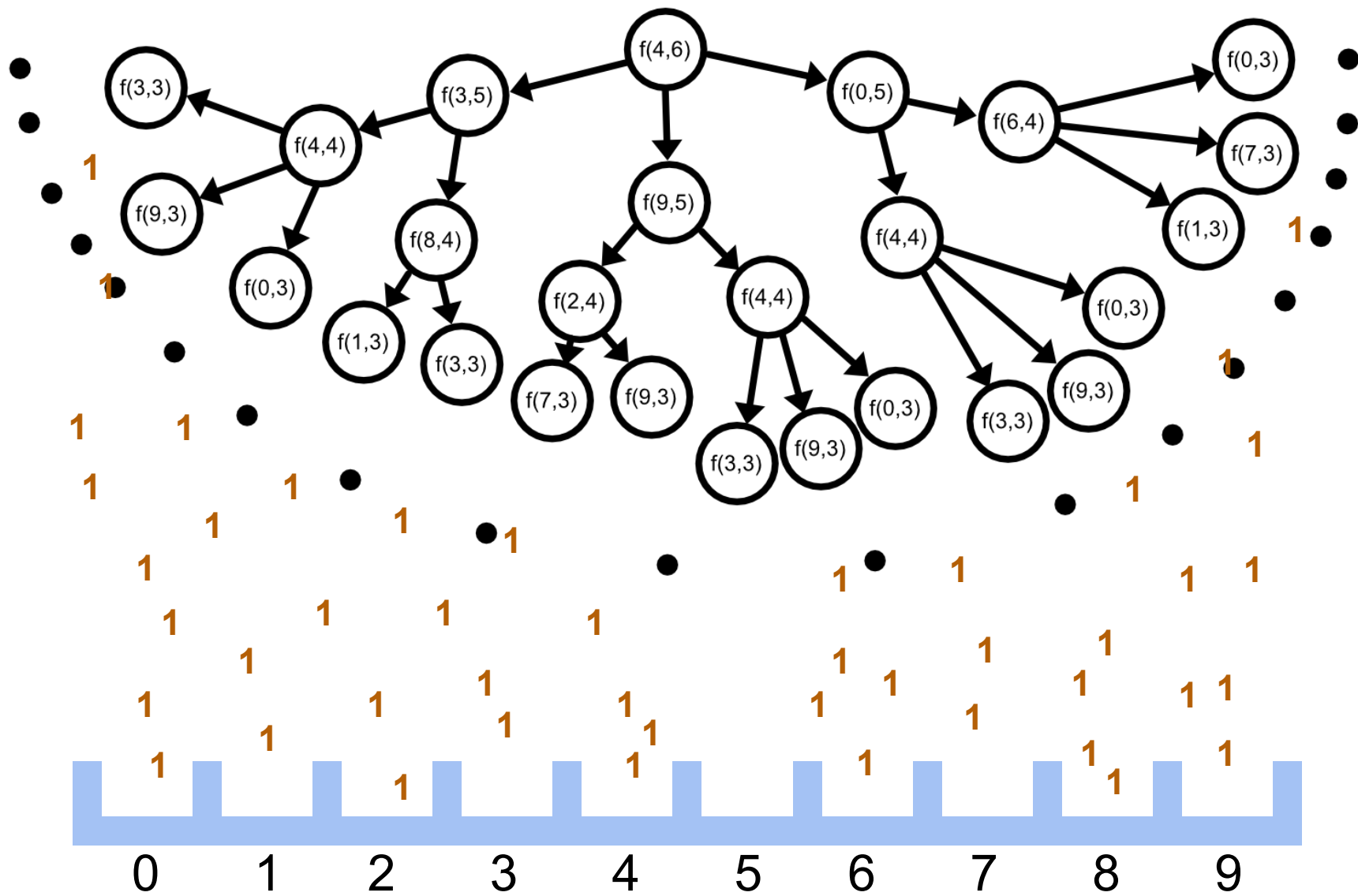
Top-Down

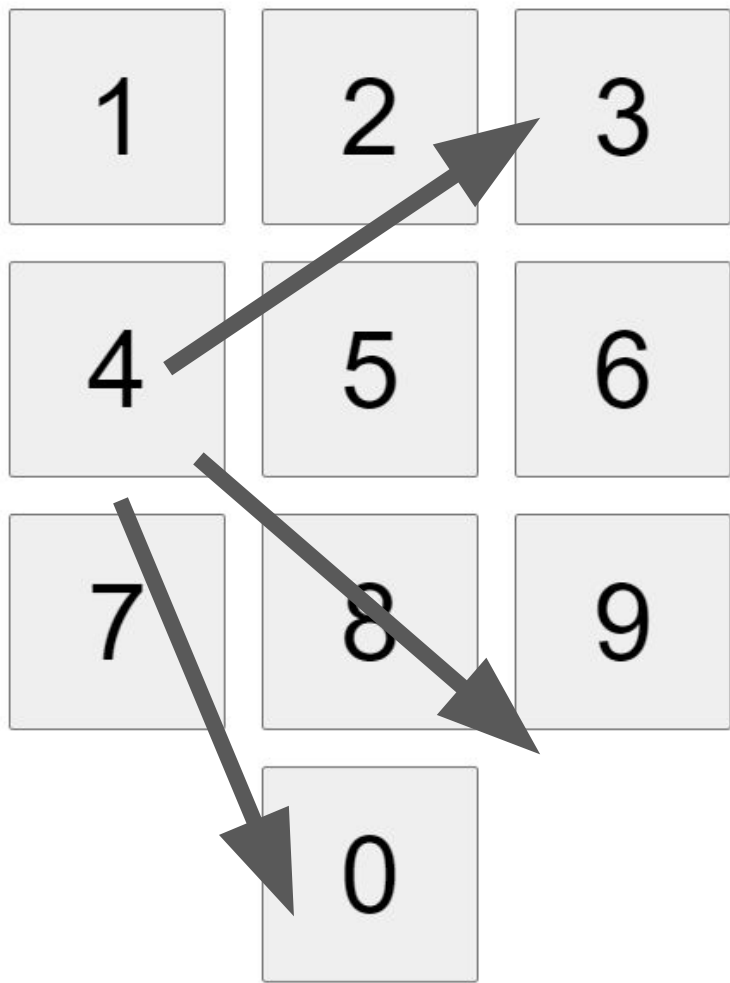
(memoization)

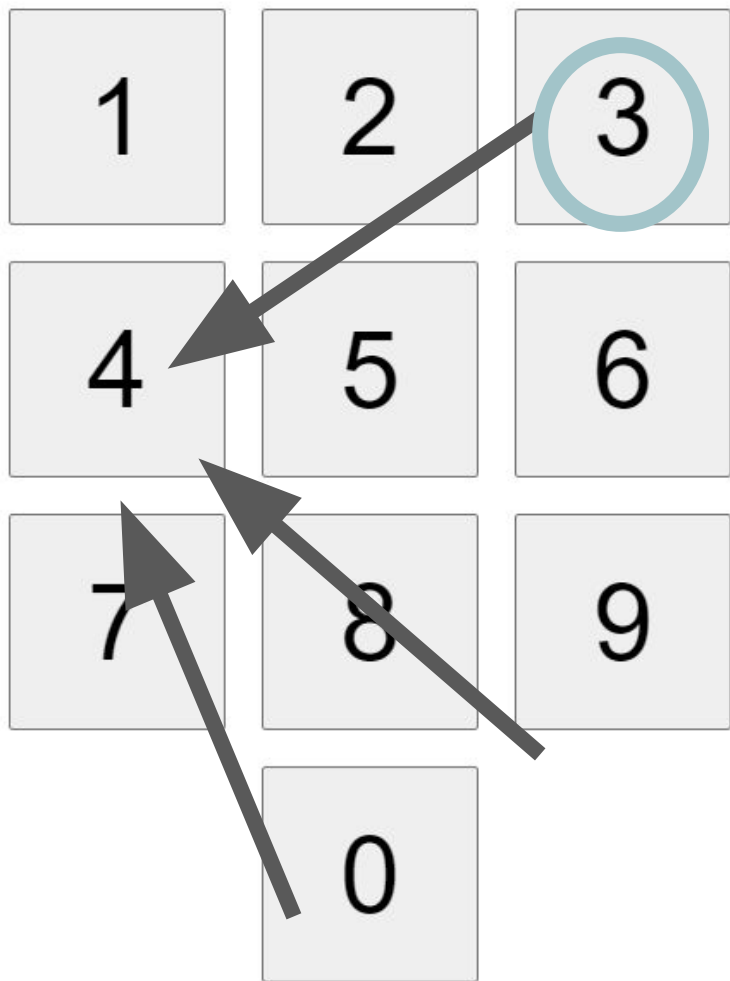
Bottom-Up

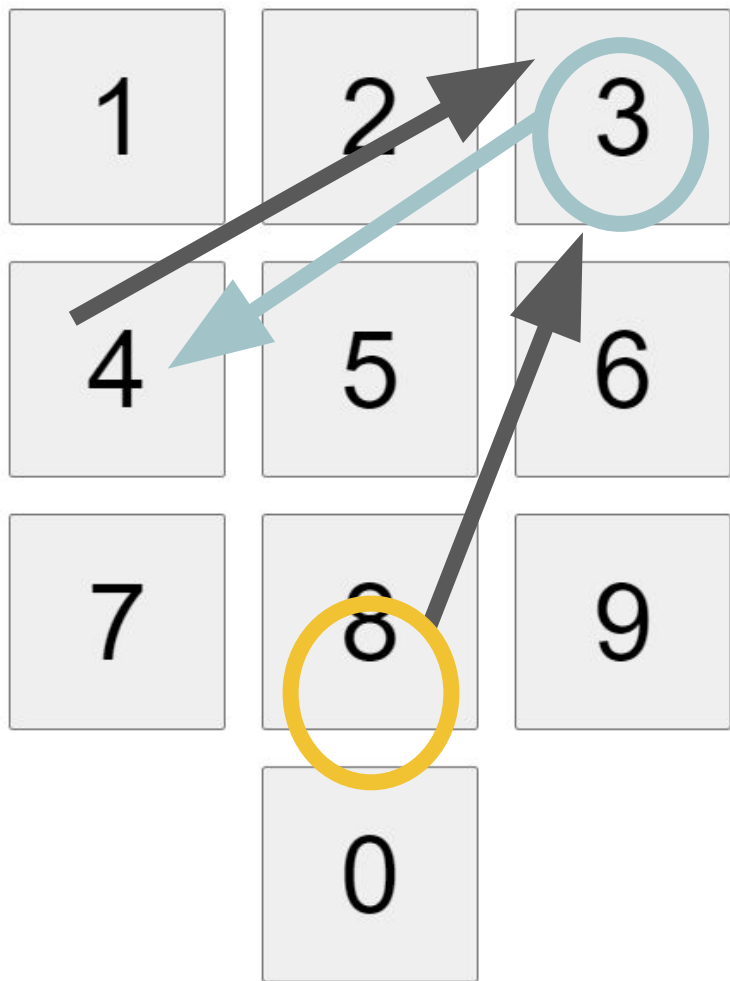
(tabulation)

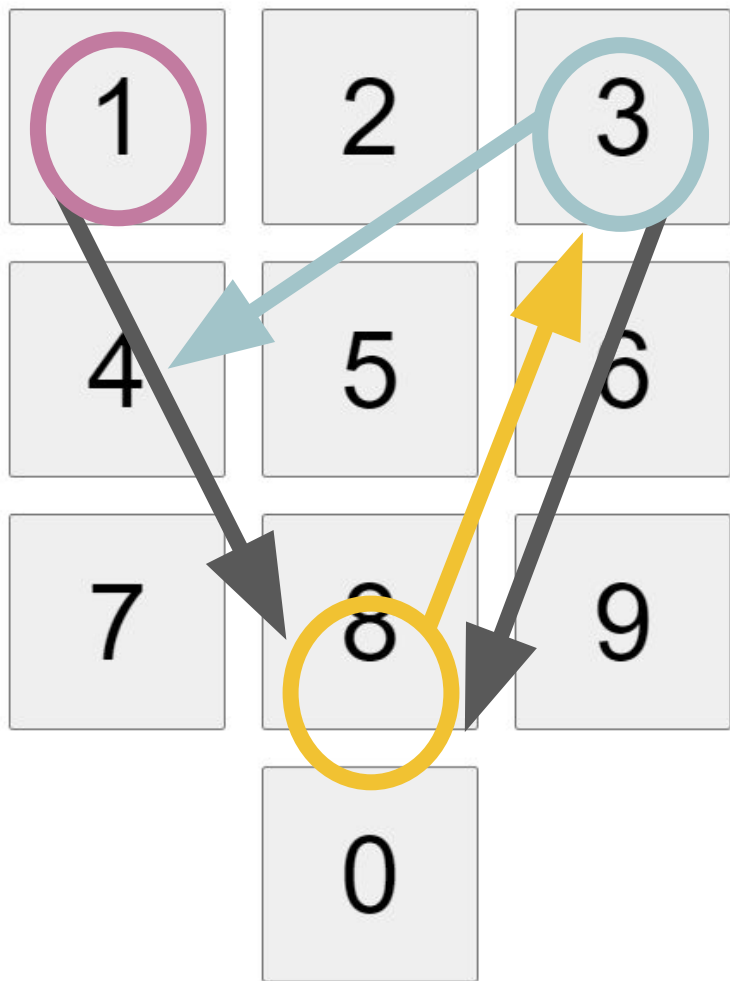
Dynamic Programming



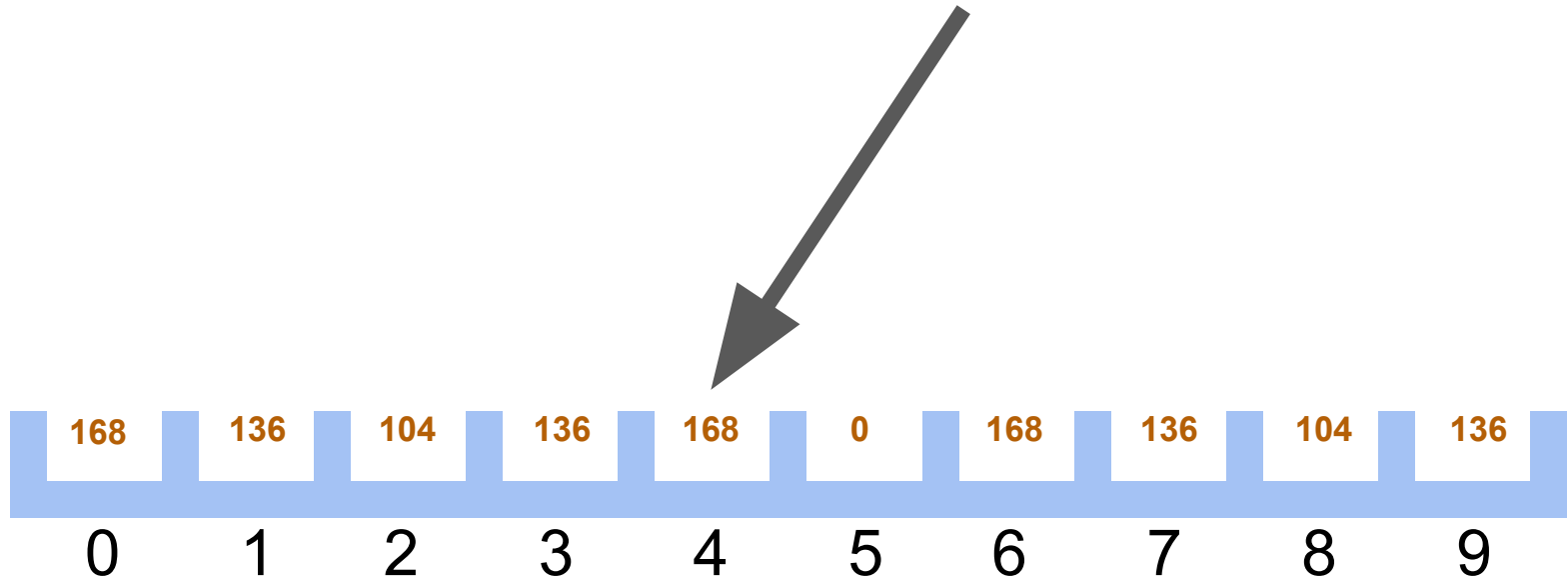




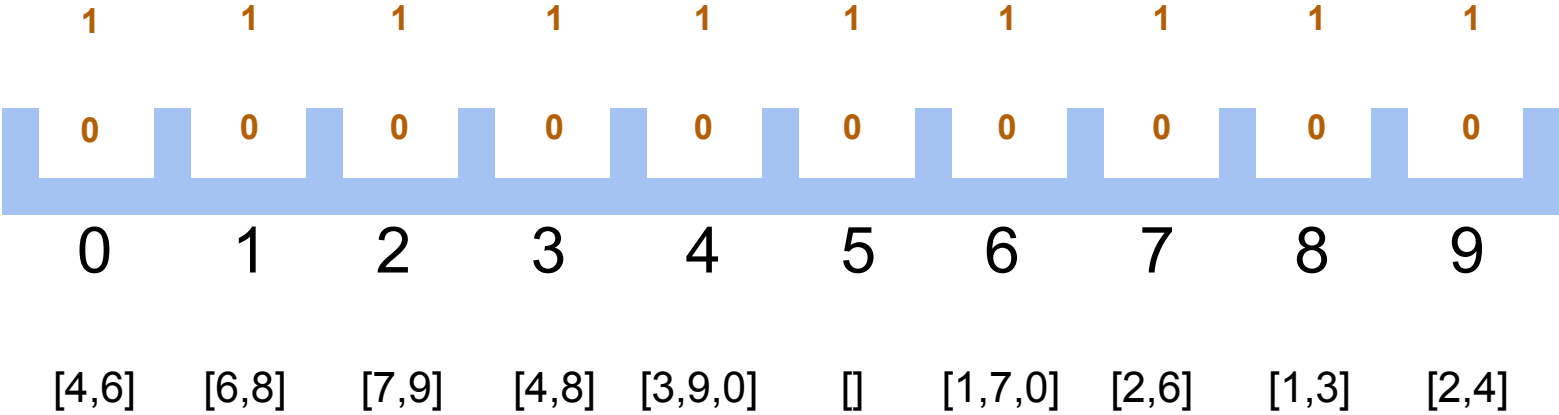




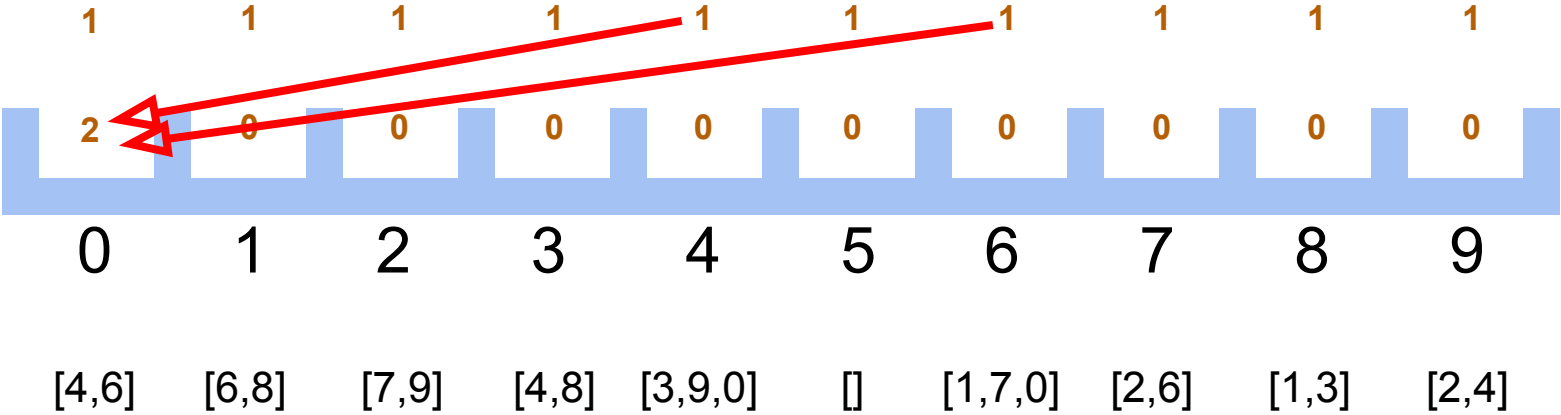
countPaths(4,6)



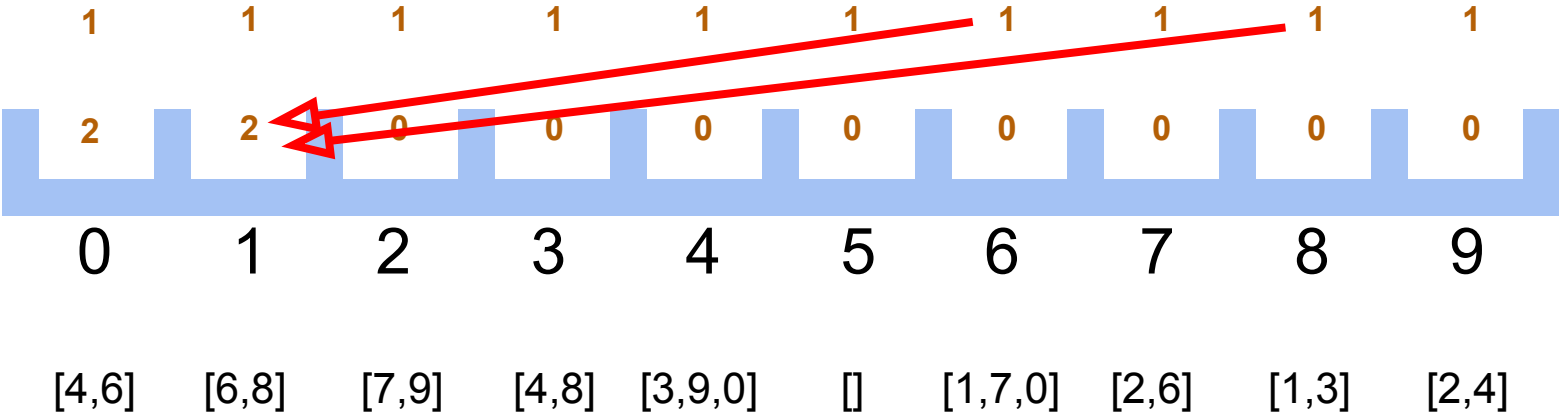
Iteration: 1
hopCount: 6



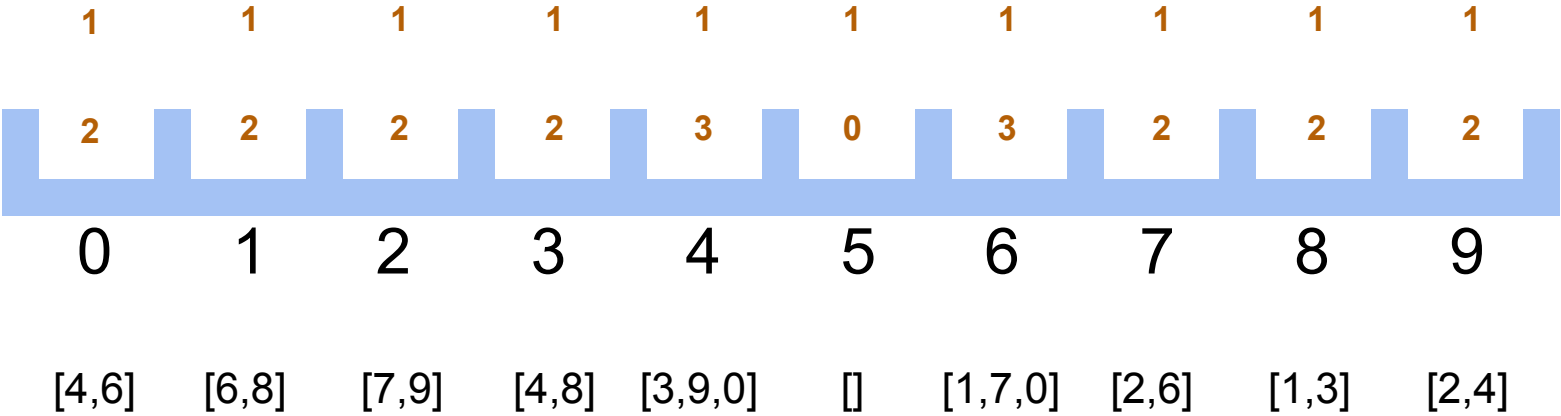
Iteration: 1
hopCount: 6



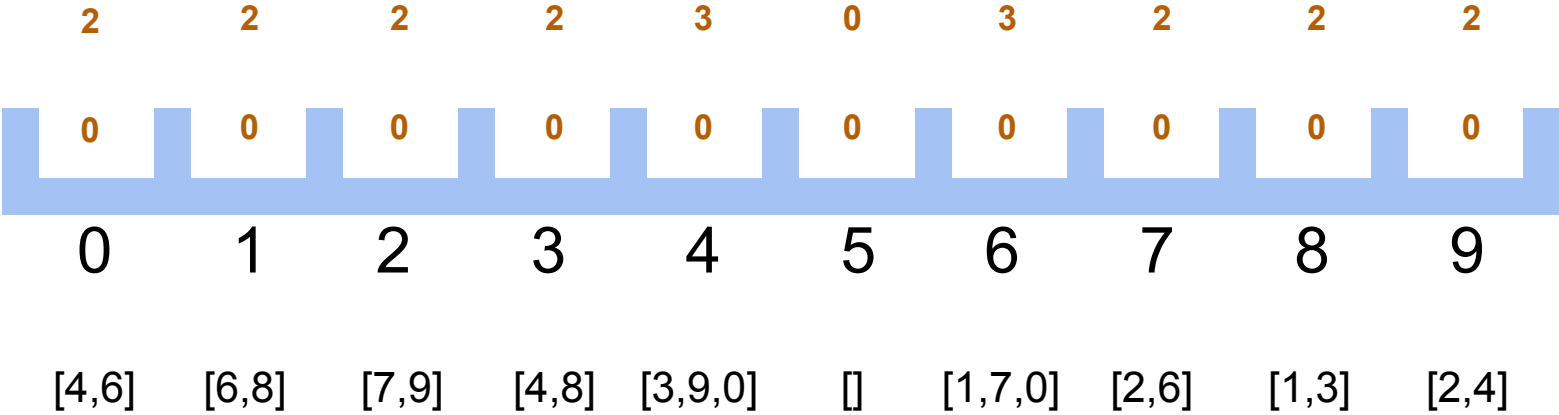
Iteration: 1
hopCount: 6



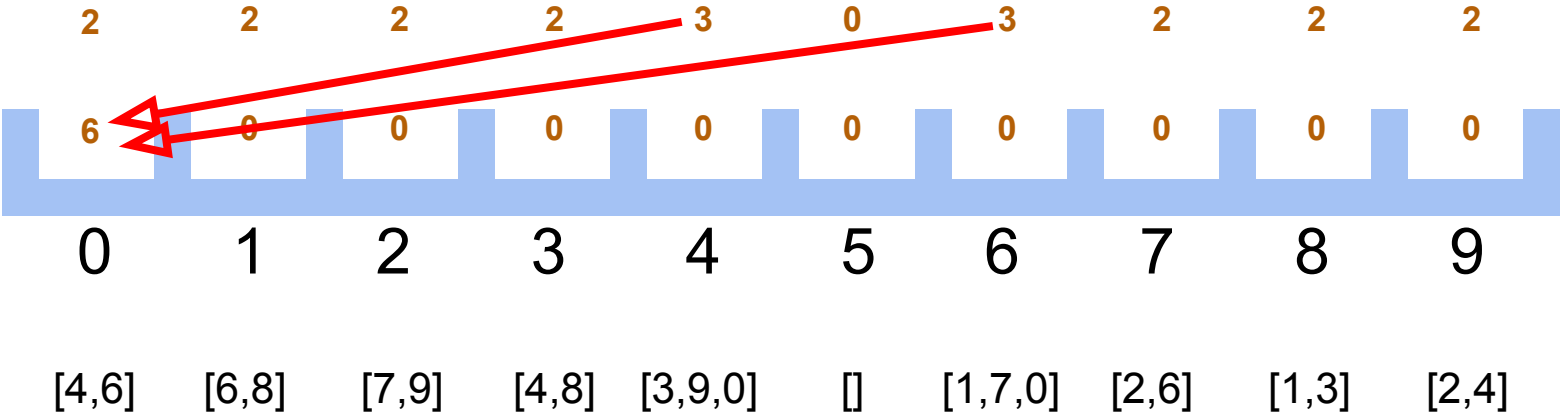
Iteration: 1
hopCount: 6



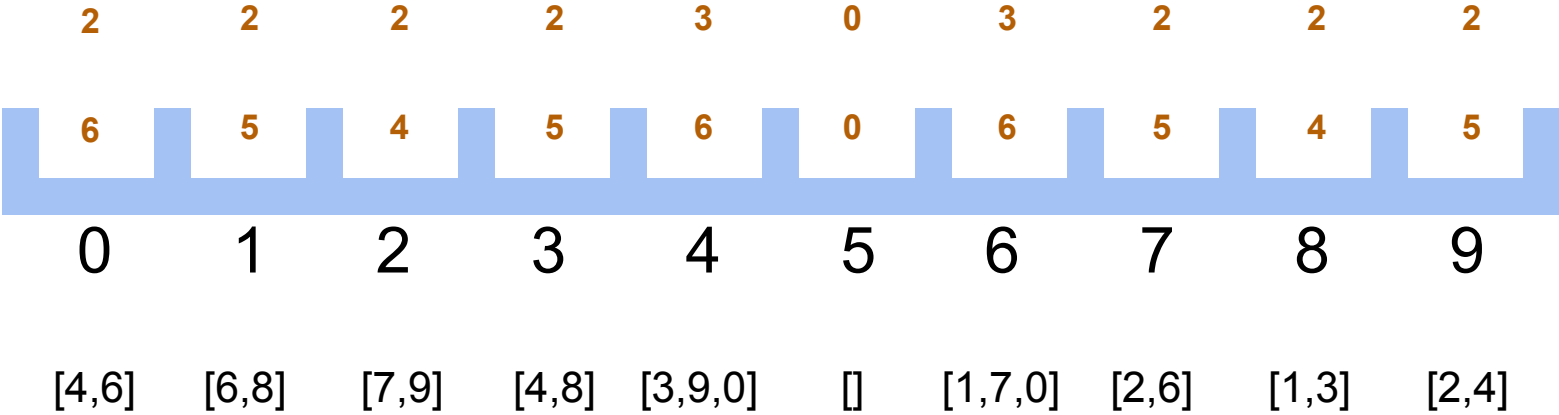
Iteration: 2
hopCount: 5



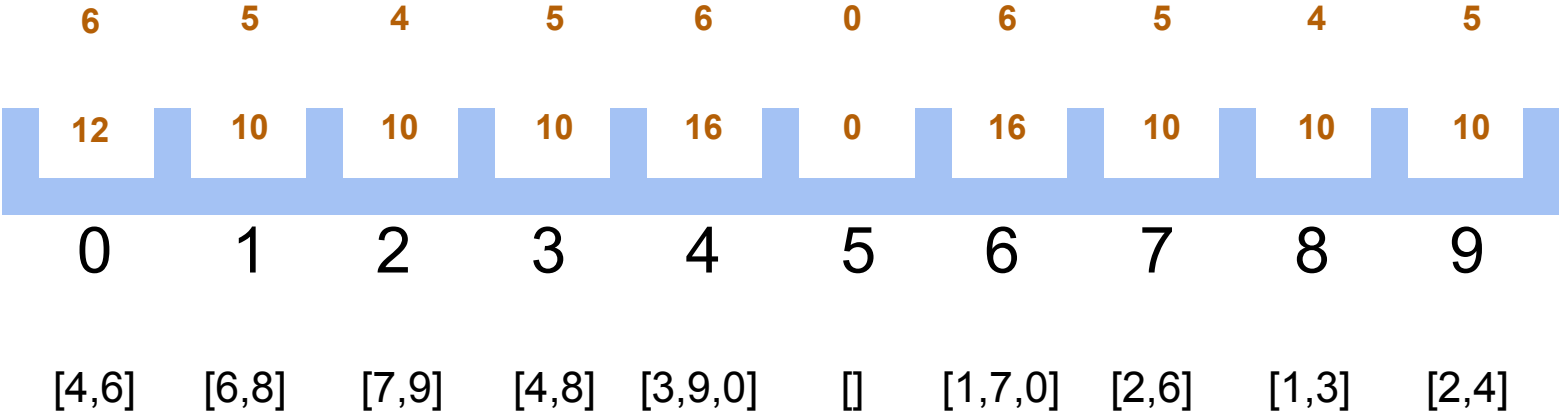
Iteration: 2
hopCount: 5



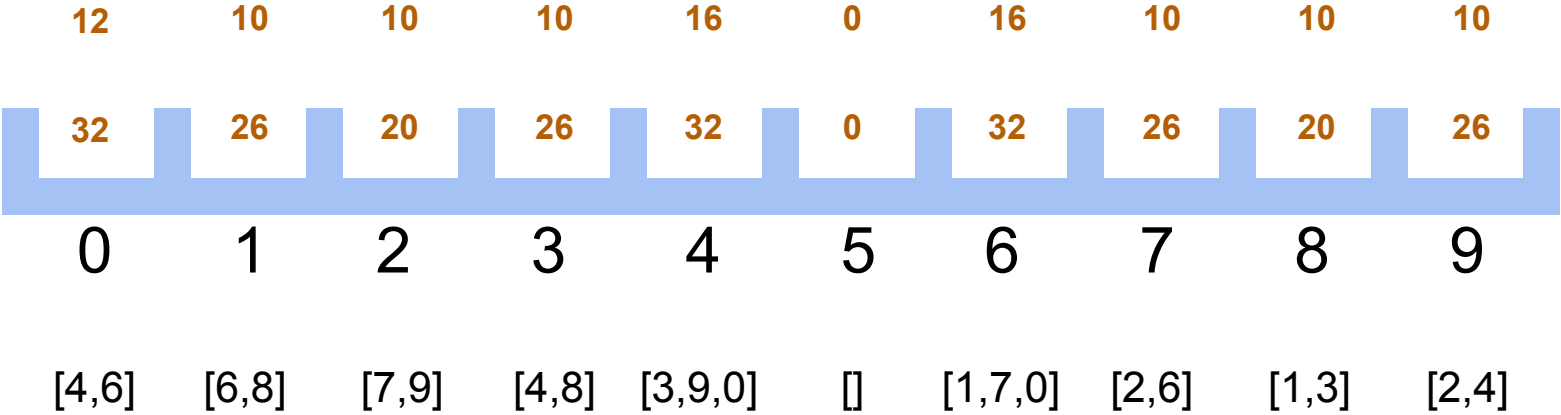
Iteration: 2
hopCount: 5



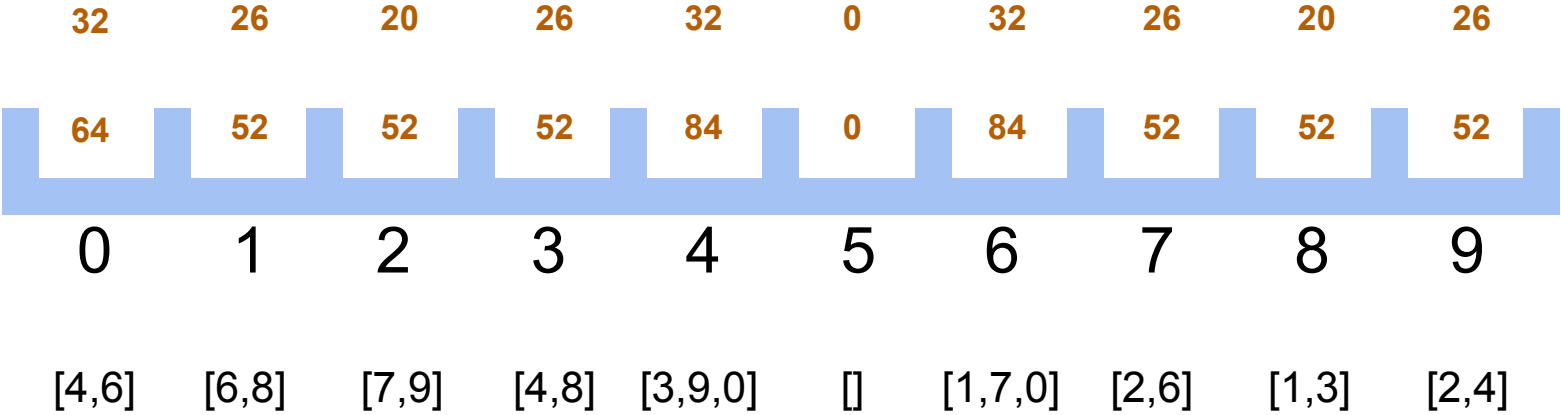
Iteration: 3
hopCount: 4



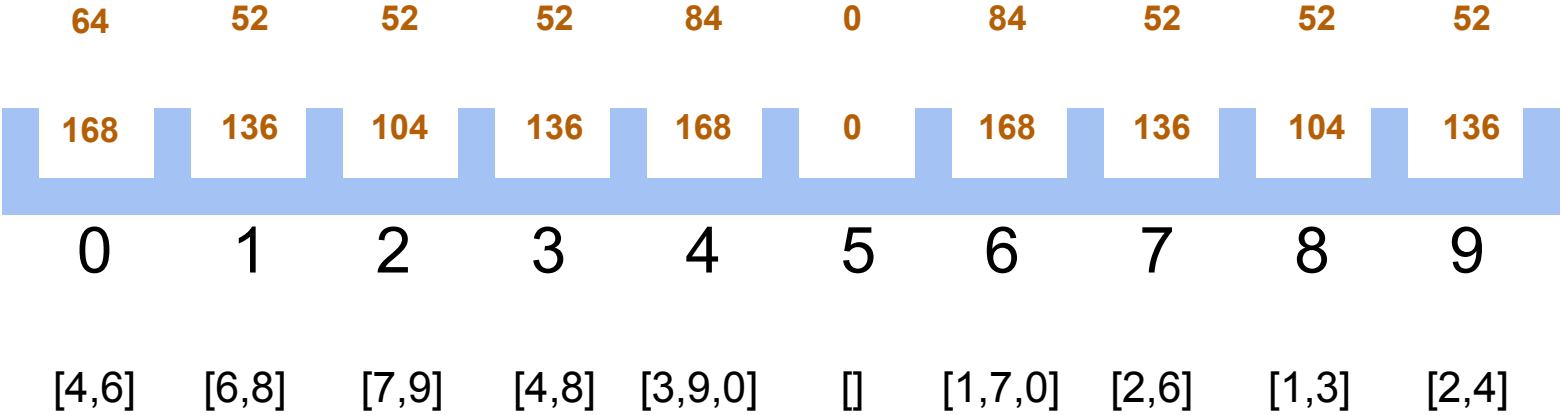
Iteration: 4
hopCount: 3



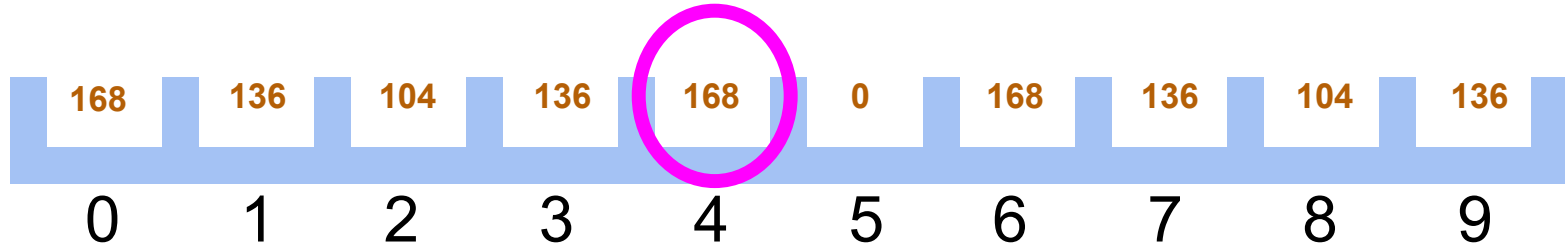
Iteration: 5
hopCount: 2



Iteration: 6
hopCount: 1



countPaths(4,6)



$O(n): n * 20 \Rightarrow n$

$O(6)$: 120 ops

$O(60)$: 1,200 ops

$O(600)$: 12,000 ops

Linear

Exercise: Wordy Unscrambler

[workshop repo](#)

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 RAECR 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 RAECR 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 **CAP**RE 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 RAECR 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 **CAP**RE 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 RAECR 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 RAECR 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 RAECR 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 RAEP C 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 RAEC P 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 RAECR 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** RAEP C 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 RAECR 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.

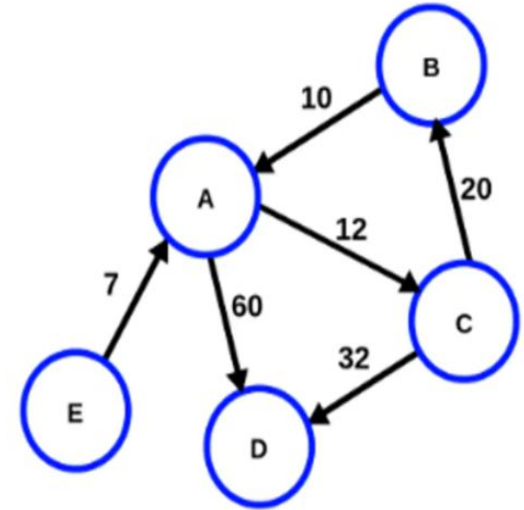
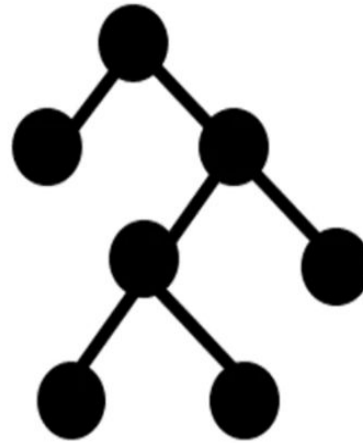
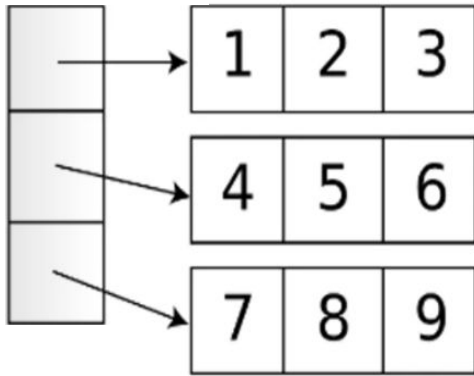
Above 11 or 12 characters, that'd be too many strings to hold in memory all at once.

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.**



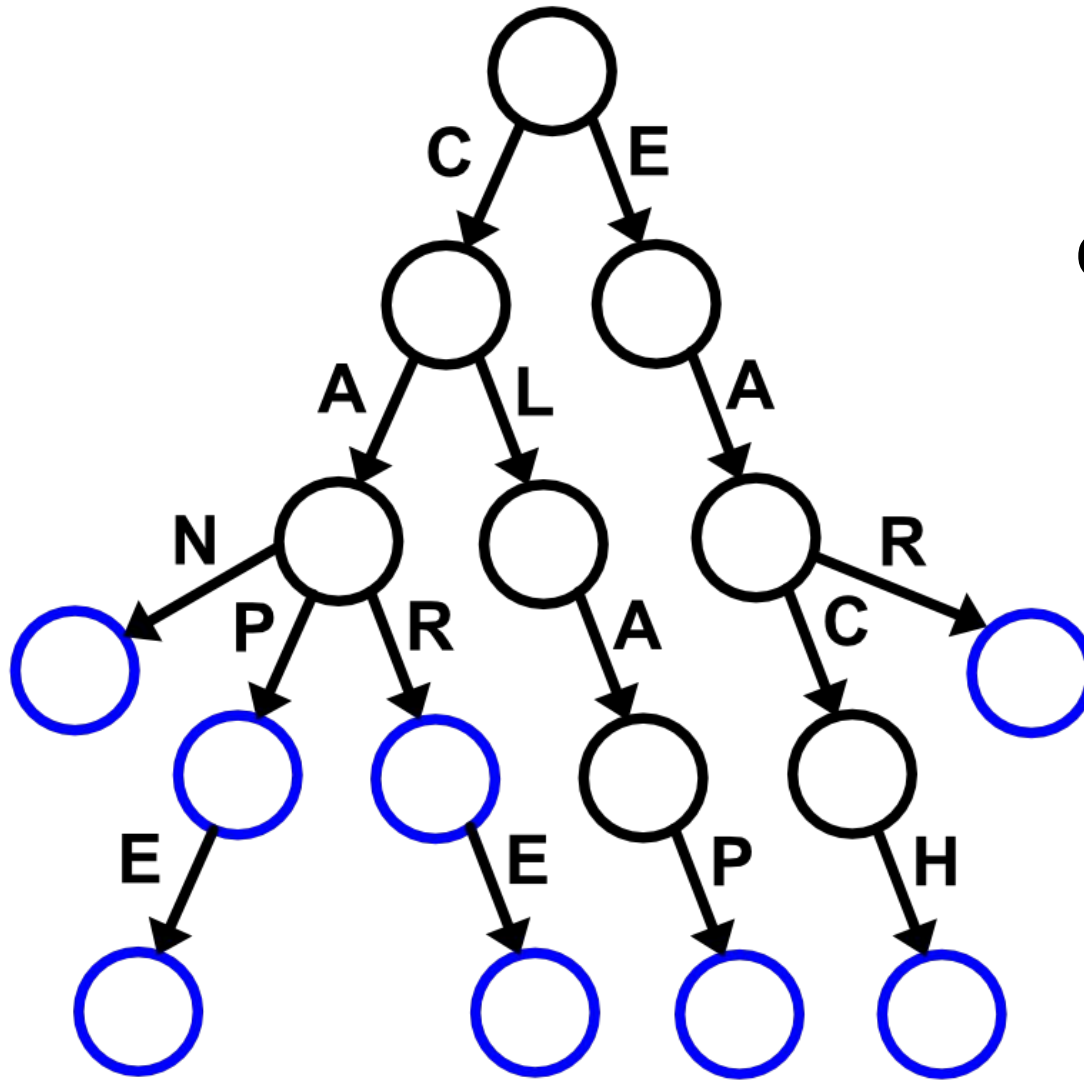
Data Structures

Trie

CAN
CAP
CAPE
CAR
CARE
CLAP
EACH
EAR

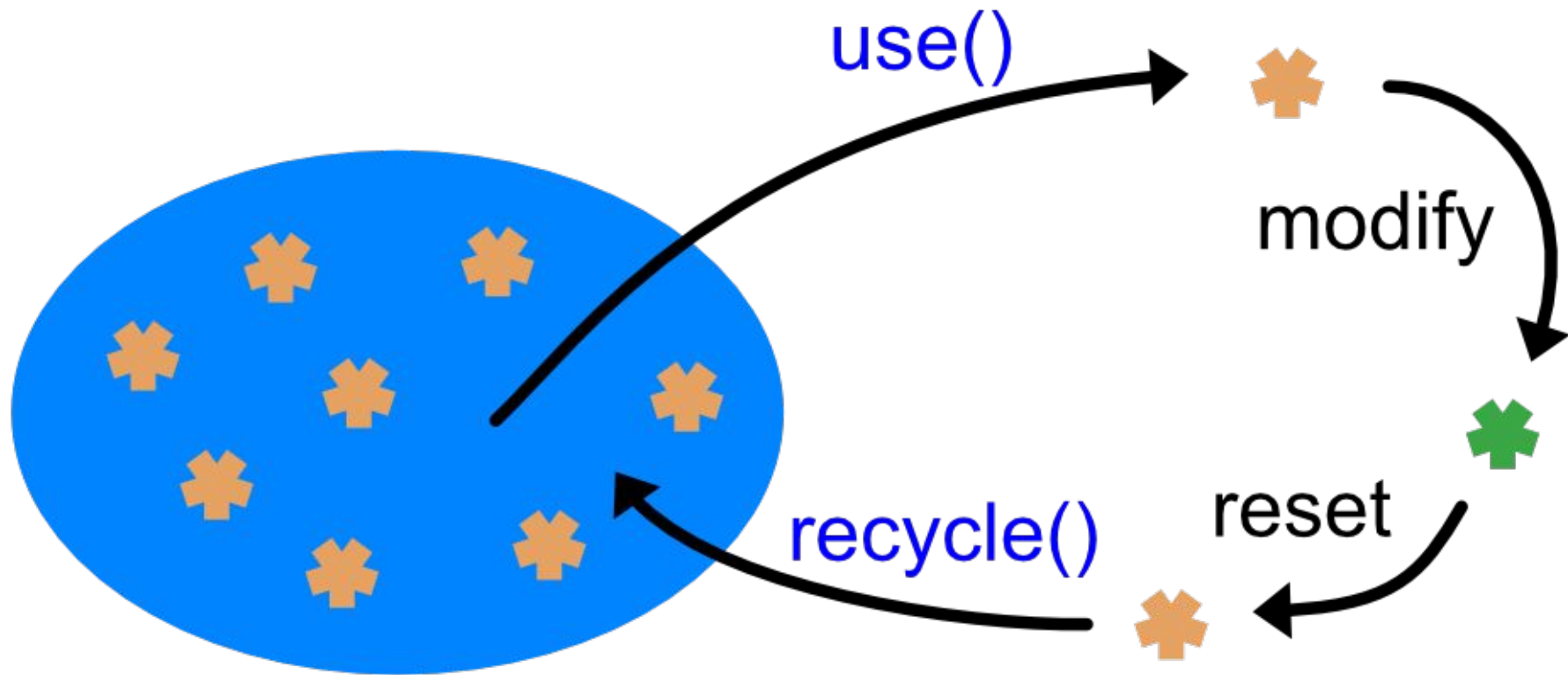
$O(k): k * m$

Linear

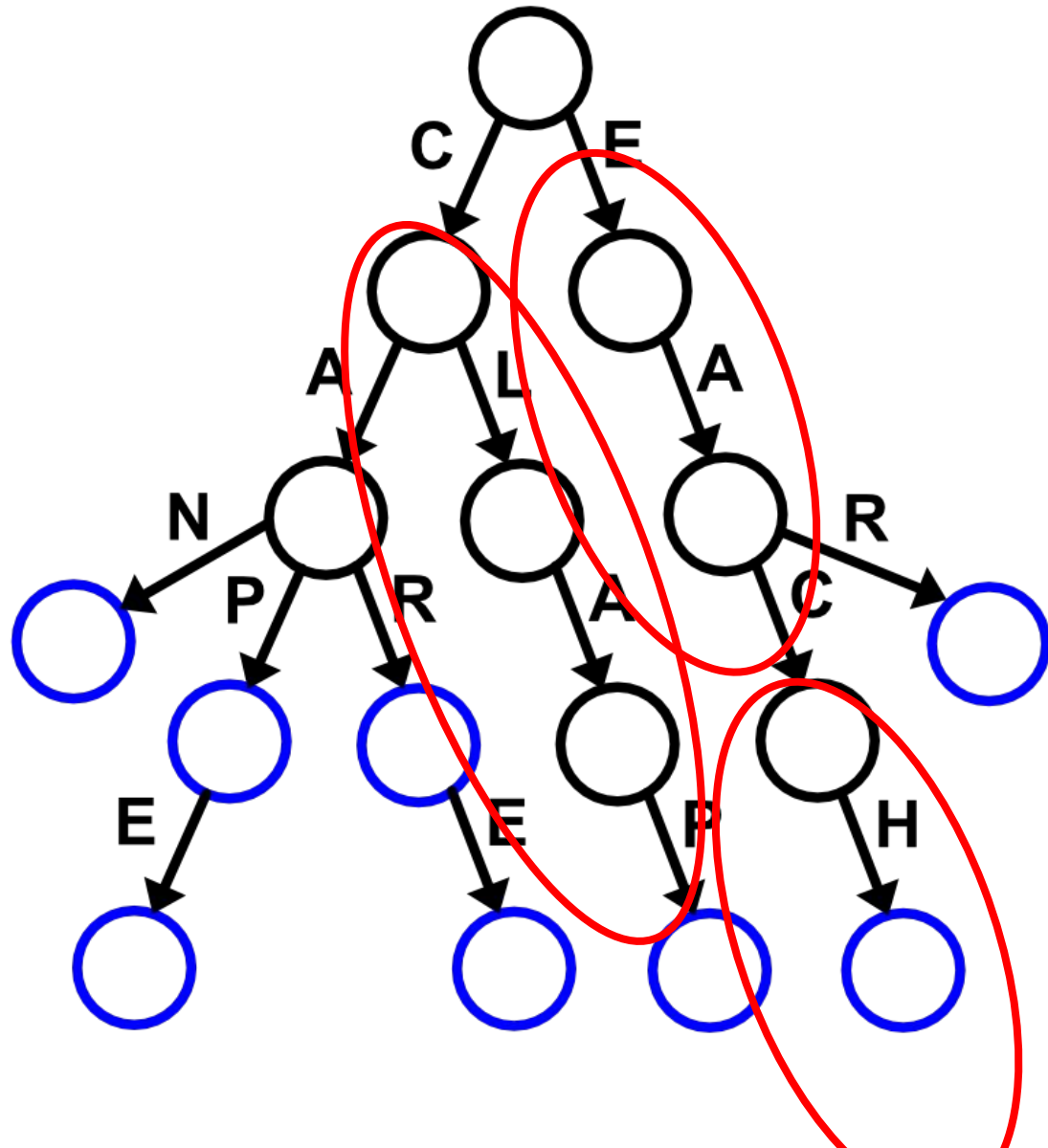




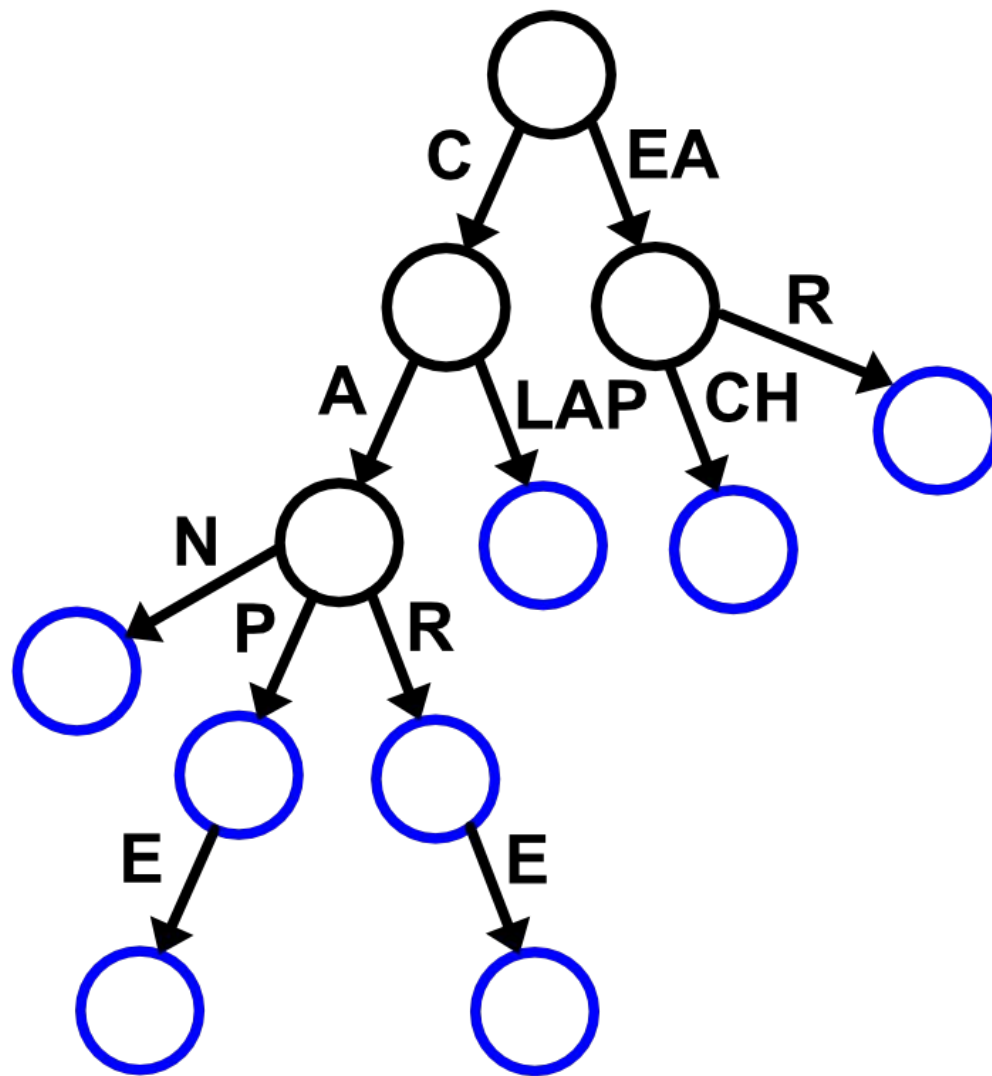
Minimizing Garbage Collection



Object Pool

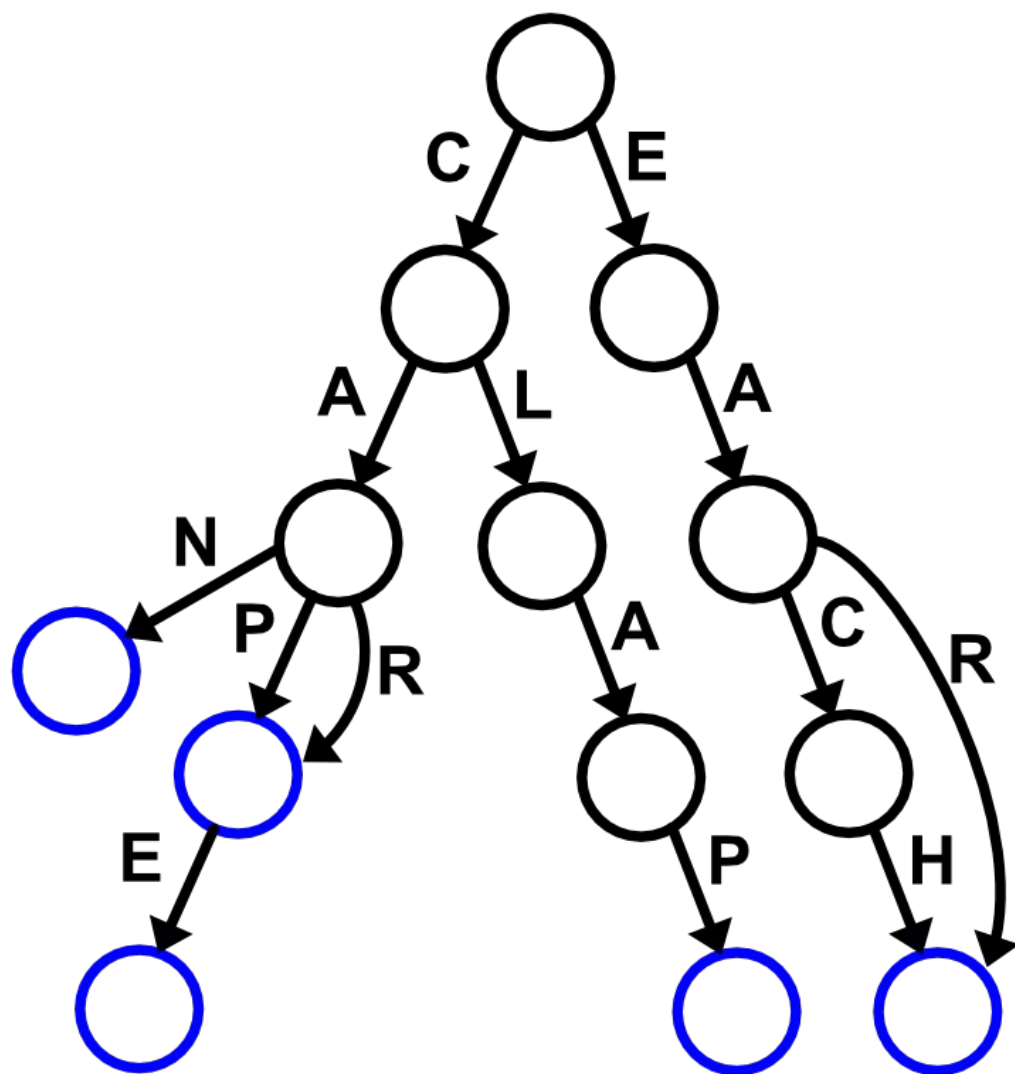


CAN
CAP
CAPE
CAR
CARE
CLAP
EACH
EAR



Radix
Tree

CAN
CAP
CAPE
CAR
CARE
CLAP
EACH
EAR



DAFSA /
DAWG

_ H _ _ _

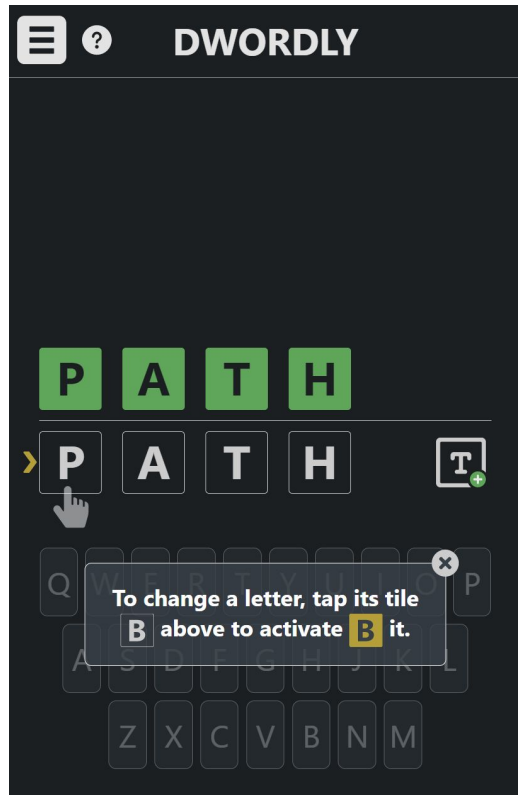


explain

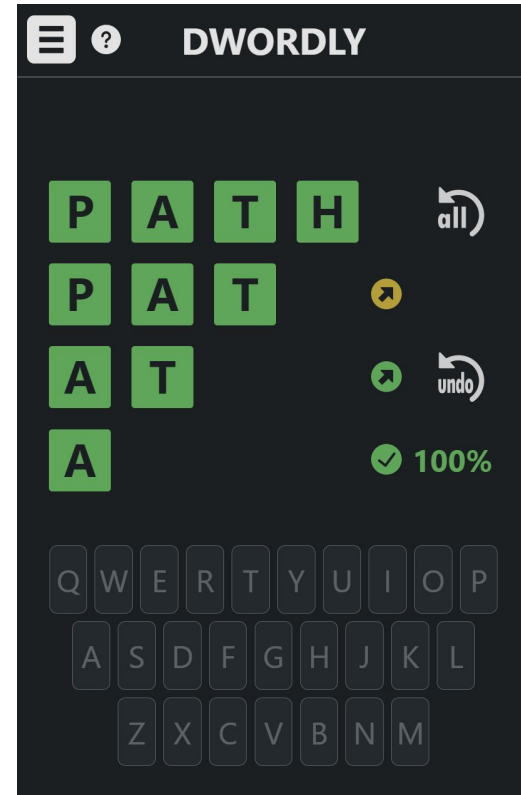
e xplain
xe plain
pxe lain
lpxe ain
alpxe in
ialpxe n
nialpxe

GADDAG

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



Bonus:
Dwordly



[website](#)

The End