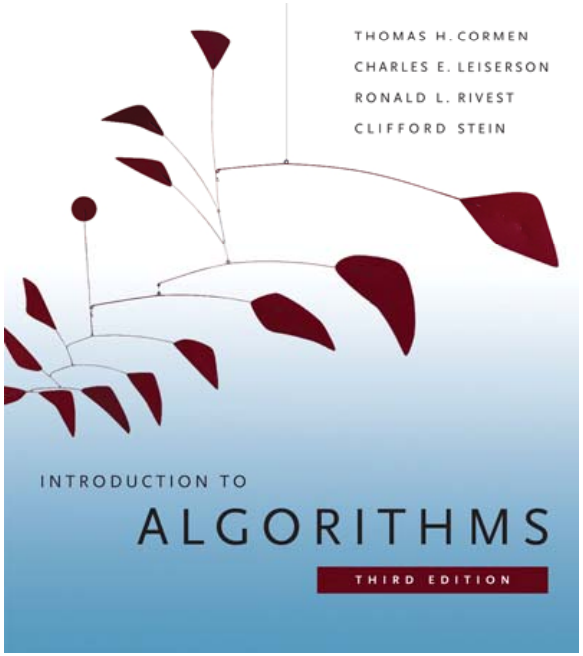
# 6.006- *Introduction to Algorithms*




## *Lecture 11 - Searching I*

**Prof. Manolis Kellis**

CLRS 22.1-22.3, B.4

# Unit #4 – Games, Graphs, Searching, Networks

| Unit | Pset | Week | Date | | Lecture (Tuesdays and Thursdays) | | Recitation (Wed and Fri) |
|---|---|---|---|---|---|---|---|
| Intro | PS1 | 1 | Tue Feb 01 | 1 | Introduction and Document Distance | 1 | Python and Asymptotic Complexity |
| Binary | Out: 2/1 | | Thu Feb 03 | 2 | Peak Finding Problem | 2 | Peak Finding correctness & analysis |
| Search | Due: Mon 2/14 | 2 | Tue Feb 08 | 3 | Scheduling and Binary Search Trees | 3 | Binary Search Tree Operations |
| Trees | HW lab: Sun 2/13 | | Thu Feb 10 | 4 | Balanced Binary Search Trees | 4 | Rotations and AVL tree deletions |
| Hashing | PS2 Out: 2/15 | 3 | Tue Feb 15 | 5 | Hashing I : Chaining, Hash Functions | 5 | Hash recipes, collisions, Python dicts |
| | Due: Mon 2/28 | | Thu Feb 17 | 6 | Hashing II : Table Doubling, Rolling Hash | 6 | Probability review, Pattern matching |
| | HW lab:Sun 2/27 | 4 | Tue Feb 22 | - | President's Day - Monday Schedule - No Class | - | No recitation |
| | | | Thu Feb 24 | 7 | Hashing III : Open Addressing | 7 | Universal Hashing, Perfect Hashing |
| Sorting | PS3. Out: 3/1 | 5 | Tue Mar 01 | 8 | Sorting I : Insertion & Merge Sort, Master Theorem | 8 | Proof of Master Theorem, Examples |
| | Due: Mon 3/7 | | Thu Mar 03 | 9 | Sorting II : Heaps | 9 | Heap Operations |
| | HW lab: Sun 3/6 | 6 | Tue Mar 08 | 10 | Sorting III: Lower Bounds, Counting Sort, Radix Sort | 10 | Models of computation |
| | | | Wed Mar 09 | Q1 | Quiz 1 in class at 7:30pm. Covers L1-R10. Review Session on Tue 3/8 at 7:30pm. | | |
| Graphs | PS4. Out: 3/10 | | Thu Mar 10 | 11 | Searching I: Graph Representation, Depth-1st Search | 11 | Strongly connected components |
| and | Due: Fri 3/18 | 7 | Tue Mar 15 | 12 | Searching II: Breadth-1st Search, Topological Sort | 12 | Rubik's Cube Solving |
| Search | HW lab:W 3/16 | | Thu Mar 17 | 13 | Searching III: Games, Network properties, Motifs | 13 | Subgraph isomorphism |
| Shortest | PS5 | 8 | Tue Mar 29 | 14 | Shortest Paths I: Introduction, Bellman-Ford | 14 | Relaxation algorithms |
| Paths | Out: 3/29 | | Thu Mar 31 | 15 | Shortest Paths II: Bellman-Ford, DAGs | 15 | Shortest Path applications |
| | Due: Mon 4/11 | 9 | Tue Apr 05 | 16 | Shortest Paths III: Dijkstra | 16 | Speeding up Dijkstra's algorithm |
| | HW lab:Sun 4/10 | | Thu Apr 07 | 17 | Graph applications, Genome Assembly | 17 | Euler Tours |
| Dynamic | PS6 | 10 | Tue Apr 12 | 18 | DP I: Memoization, Fibonacci, Crazy Eights | 18 | Limits of dynamic programming |
| Program | Out: Tue 4/12 | | Wed Apr 13 | Q2 | Quiz 2 in class at 7:30pm. Covers L11-R17. Review Session on Tue 4/13 at 7:30pm. | | |
| ming | Due: Fri 4/29 | | Thu Apr 14 | 19 | DP II: Shortest Paths, Genome sequence alignment | 19 | Edit Distance, LCS, cost functions |
| | HW lab:W 4/27 | 11 | Tue Apr 19 | - | Patriot's Day - Monday and Tuesday Off | - | No recitation |
| | | | Thu Apr 21 | 20 | DP III: Text Justification, Knapsack | 20 | Saving Princess Peach |
| | | 12 | Tue Apr 26 | 21 | DP IV: Piano Fingering, Vertex Cover, Structured DP | 21 | Phylogeny |
| Numbers | PS7 out Thu4/28 | | Thu Apr 28 | 22 | Numerics I - Computing on large numbers | 22 | Models of computation return! |
| Pictures | Due: Fri 5/6 | 13 | Tue May 3 | 23 | Numerics II - Iterative algorithms, Newton's method | 23 | Computing the nth digit of π |
| (NP) | HW lab: Wed 5/4 | | Thu May 5 | 24 | Geometry: Line sweep, Convex Hull | 24 | Closest pair |
| | | 14 | Tue May 10 | 25 | Complexity classes, and reductions | 25 | Undecidability of Life |
| Beyond | | | Thu May 12 | 26 | Research Directions (15 mins each) + related classes | | |
| | | 15 | Finals week | Q3 | Final exam is cumulative L1-L26. Emphasis on L18-L26. Review Session on Fri 5/13 at 3pm | | |

# Unit #4 Overview: Searching

**Today: Introduction to Games and Graphs**

- Rubik's cube, Pocket cube, Game space
- Graph definitions, representation, searching

**Tuesday: Graph algorithms and analysis**

- Breadth First Search, Depth First Search
- Queues, Stacks, Augmentation, Topological sort

**Thursday: Networks in biology and real world**

- Network/node properties, metrics, motifs, clusters
- Dynamic processes, epidemics, growth, resilience

# Graph Applications

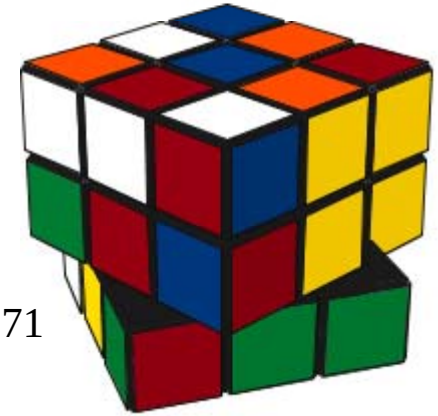
- Web
  - crawling
- Social Network
  - friend finder
- Computer Networks
  - internet routing
  - connectivity
- Game states
  - rubik’s cube, chess

# Today: Solving Rubik's cube…


… and finding God's number

# Cracking the 3x3 Rubik's cube

- Increasingly efficient algorithms exist for solving the cube using a fixed set of moves
  - 1981: 52 moves. Today: <30 moves
- In practice, shortcuts *may* be possible!
  - Human intuition can reveal patterns, not follow fixed algorithm
- How hard is Rubik's cube:
  - Size of game space: count distinct positions, number of edges
  - 43,252,003,274,489,856,000 positions ($4.3*10^{19}$)
- How big is 43 quadrillion?
  - Number of atoms in the universe: $10^{81}$
  - Complexity of chess (Shannon number): ~$10^{47}$
  - 19x19 go: #turns ~$10^{48}$ ; $10^{10\text{^}48}$<#games<~$10^{10\text{^}171}$



# Searching for God's number

| Date | Lower bound | Upper bound | Gap |
|---|---|---|---|
| July, 1981 | 18 | 52 | 34 |
| April, 1992 | 18 | 42 | 24 |
| May, 1992 | 18 | 39 | 21 |
| May, 1992 | 18 | 37 | 19 |
| January, 1995 | 18 | 29 | 11 |
| January, 1995 | 20 | 29 | 9 |
| December, 2005 | 20 | 28 | 8 |
| April, 2006 | 20 | 27 | 7 |
| May, 2007 | 20 | 26 | 6 |
| March, 2008 | 20 | 25 | 5 |
| April, 2008 | 20 | 23 | 3 |
| August, 2008 | 20 | 22 | 2 |
| **July, 2010** | **20** | **20** | **0** |

- ***God's algorithm*** would always use the minimal number of moves
- ***God's number***: maximum number of moves needed by an optimal algorithm
- ***Upper bound*** nearing in by increasingly faster general algorithms
- ***Lower bound*** given by hardest known positions requiring most moves
- The two met just last year!

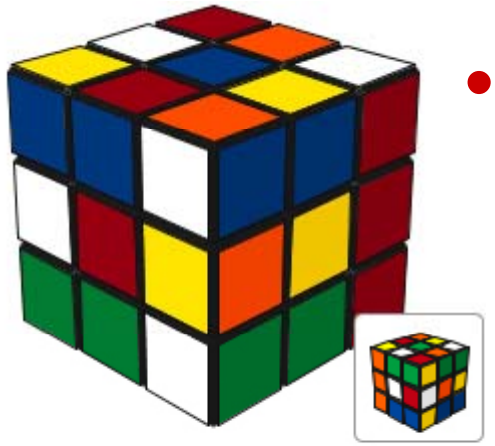
# So, how did they do it? (TM)

- Start with 43,252,003,274,489,856,000 positions
- Partition into 2.2 billion sets, each with 19.5 billion positions, solve each separately
- Reduce 2.2 billion sets to 55.8 million by symmetry & set cover
- Solve each set of 19.5 billion positions in 20 seconds or less each
- With only 20 seconds to solve each set of 19.5 billion positions, solve each of 2.2 billion sets
- Call Google and compute 35 CPU years in a week

cube20.org

# How many ‘hardest’ positions exist?

- 18 is the most frequent min number of required moves
- Relatively few 20-away positions exist
- No position requires 21 moves!



- The ‘hardest’ position for the author’s solver

FU-F2D-BUR-F-LD-R-U-LUB-D2R-FU2D2

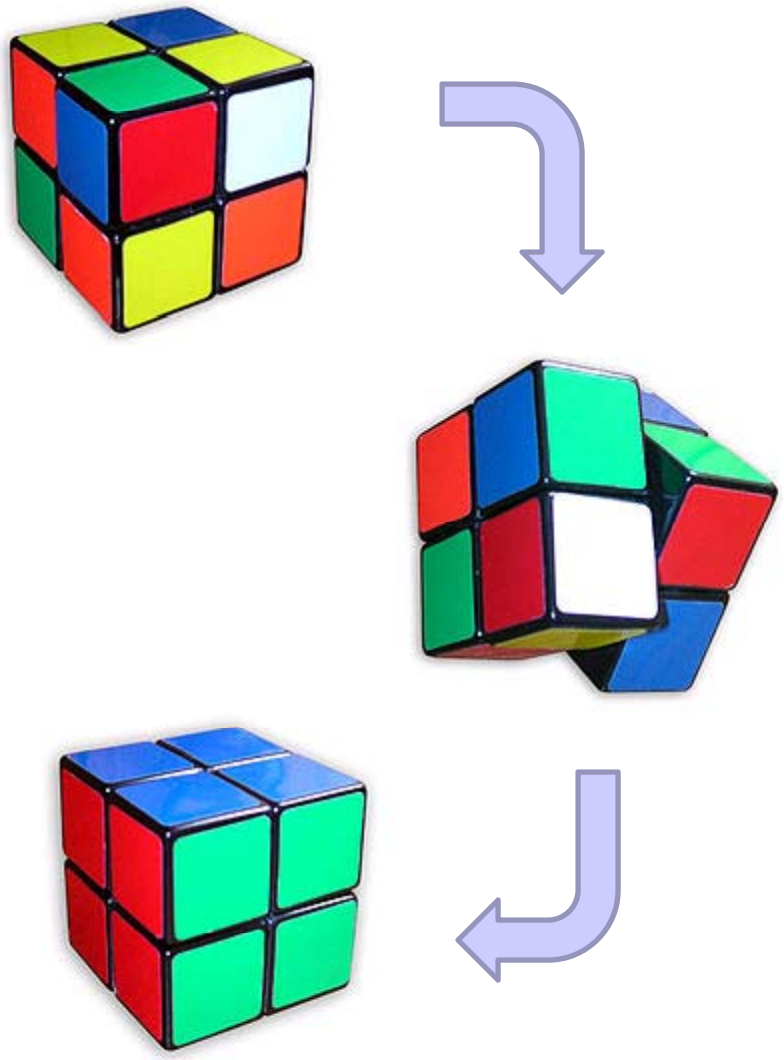
cube20.org

| Distance | Count of Positions |
|---|---|
| 0 | 1 |
| 1 | 18 |
| 2 | 243 |
| 3 | 3,240 |
| 4 | 43,239 |
| 5 | 574,908 |
| 6 | 7,618,438 |
| 7 | 100,803,036 |
| 8 | 1,332,343,288 |
| 9 | 17,596,479,795 |
| 10 | 232,248,063,316 |
| 11 | 3,063,288,809,012 |
| 12 | 40,374,425,656,248 |
| 13 | 531,653,418,284,628 |
| 14 | 6,989,320,578,825,350 |
| 15 | 91,365,146,187,124,300 |
| 16 | ~1,100,000,000,000,000,000 |
| 17 | ~12,000,000,000,000,000,000 |
| 18 | ~29,000,000,000,000,000,000 |
| 19 | ~1,500,000,000,000,000,000 |
| 20 | ~300,000,000 |
| 21 | None |

# Representing space of solutions

2x2 Rubik’s cube

# Pocket Cube



- $2 \times 2 \times 2$ Rubik's cube
- Start with any colors
- Moves are quarter turns of any face
- "Solve" by making each side one color

# Configuration Graph

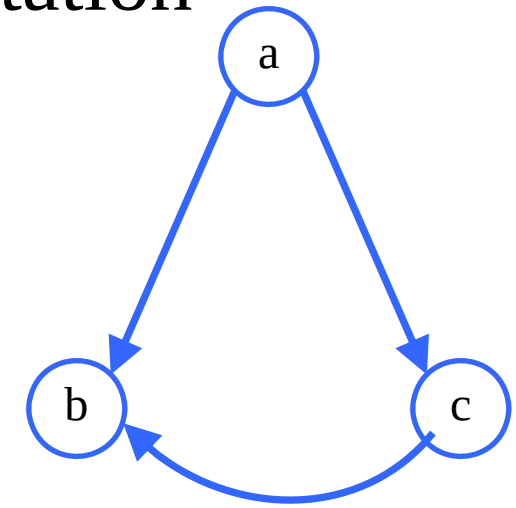
- One vertex for each state
- One edge for each move from a vertex
  - 6 faces to twist
  - 3 nontrivial ways to twist (1/4, 2/4, 3/4)
  - So, 18 edges out of each state
- Solve cube by finding a path (of moves) from initial state (vertex) to “solved” state

# Combinatorics

- State for each arrangement and orientation of 8 cubelets
  - 8 cubelets in each position: 8! Possibilities
  - Each cube has 3 orientations: $3^8$ Possibilities
  - Total: $8!*3^8 = 264{,}539{,}320$ vertices
- But divide out 24 orientations of whole cube
- And there are three separate connected components (twist one cube out of place 3 ways)
- Result: 3,674,160 states to search

# Graph formalization

Definitions and representation

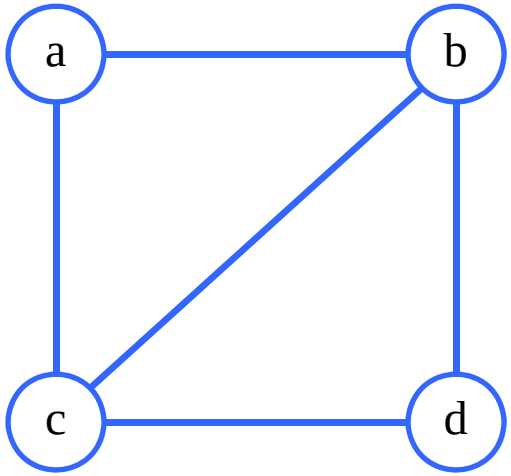


# Graph Definitions

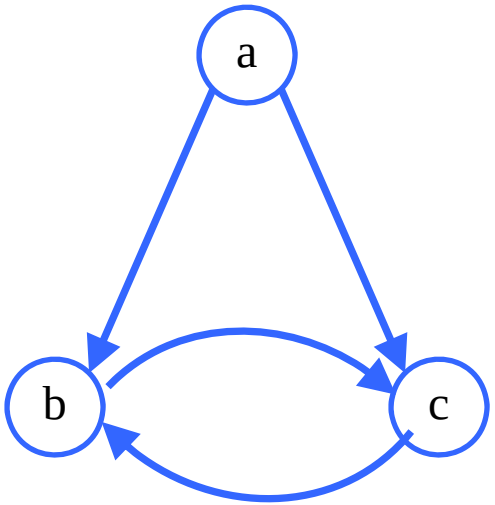
- G=(V,E)
- V a set of vertices
  - usually number denoted by n
- $E \subseteq V \times V$ a set of edges (pairs of vertices)
  - usually number denoted by m
  - note $m < n(n-1) = O(n^2)$
- Flavors:
  - pay attention to order: directed graph
  - ignore order: undirected graph
    - Then only n(n-1)/2 possible edges

# Graph Examples

- Undirected
- V={a,b,c,d}
- E={{a,b}, {a,c}, {b,c}, {b,d}, {c,d}}




- Directed
- V = {a,b,c}
- E = {(a,c), (a,b) (b,c), (c,b)}



# Graph Representation

- To solve graph problems, must examine graph
- So need to represent in computer
- Four representations with pros/cons
  1. Adjacency lists (of neighbors of each vertex)
  2. Incidence lists (of edges from each vertex)
  3. Adjacency matrix (of which pairs are adjacent)
  4. Implicit representation (as neighbor function)

# List vs. matrix representations

Adjacency list

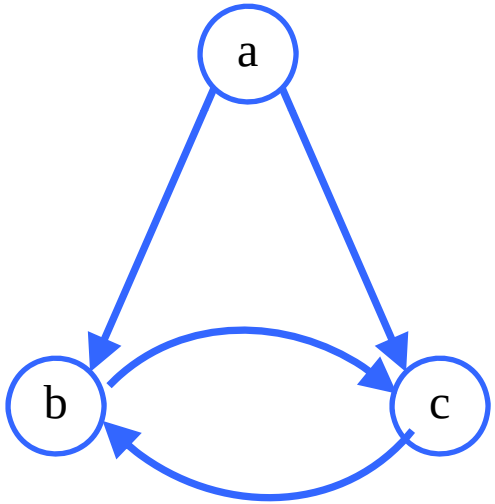
Adjacency matrix

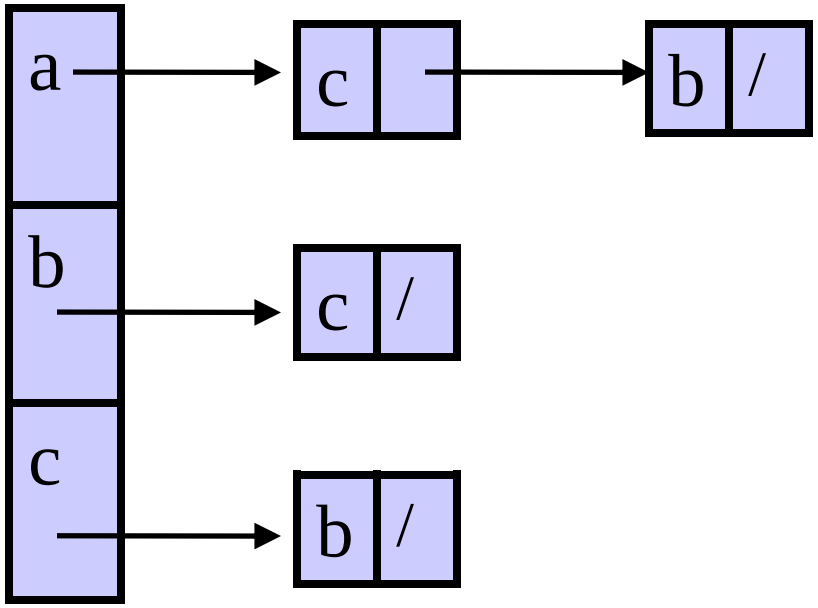
Space/time tradeoffs

# 1. Adjacency Lists

- For each vertex v, list its neighbors (vertices to which it is connected by an edge)
  - Array A of $|V|$ linked lists
  - For $v \in V$, list A[v] stores neighbors $\{u \mid (v,u) \in E\}$
  - Directed graph only stores outgoing neighbors
  - Undirected graph stores edge in two places
- In python, A[v] can be hash table
  - v any hashable object

# Adjacency list example

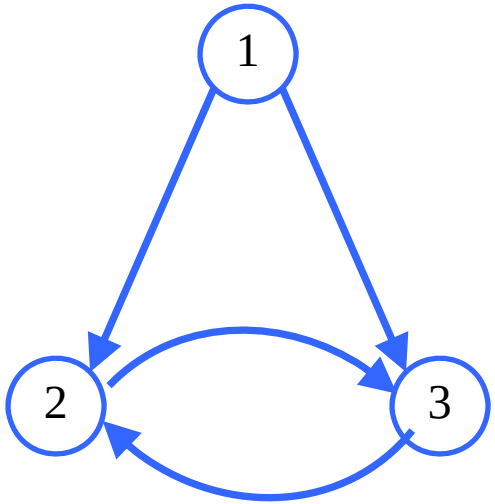


# 2. Adjacency Matrix

- assume $V=\{1, \ldots, n\}$
- matrix $A=(a_{ij})$ is $n \times n$
  - row i, column j
  - $a_{ij} = 1$ if $(i,j) \in E$
  - $a_{ij} = 0$ otherwise
- (store as, e.g., array of arrays)

# Adjacency Matrix Example




| 1 | 2 | 3 | |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 3 |

# Graphs and Matrix Algebra

- can treat adjacency matrix as matrix
- e.g., $A^2$ = length-2 paths between vertices ..
- [note: $A^k$ for large k is related to pagerank of vertices]
- undirected graph → symmetric matrix
- [eigenvalues useful for many graph algorithms, see Lecture 13 for examples]

# Representation Tradeoffs: Space

- Adjacency lists use one list node per edge
  - And two machine words per node
  - So space is $\Theta(mw)$ bits (m=#edges, w=word size)
- Adjacency matrix uses $n^2$ entries
  - But each entry can be just one bit
  - So $\Theta(n^2)$ bits
- Matrix better only for very dense graphs
  - m near $n^2$
  - (Google can't use matrix)

# Representation Tradeoffs: Time

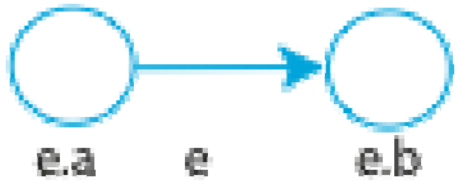
- Add edge
  - both data structures are O(1)
- Check “is there an edge from u to v”?
  - matrix is O(1)
  - adjacency list must be scanned
- Visit all neighbors of v (very common)
  - adjacency list is O(neighbors)
  - matrix is Θ(n)
- Remove edge
  - like find + add

# Other representations

Object-oriented, implicit

# Object Oriented Variants

- object for each vertex u
  - u.neighbors is list of neighbors for u
- incidence list: object for each edge e
  - u.edges = list of outgoing edges from u
  - e object has endpoints e.head and e.tail




- can store additional info per vertex or edge without hashing

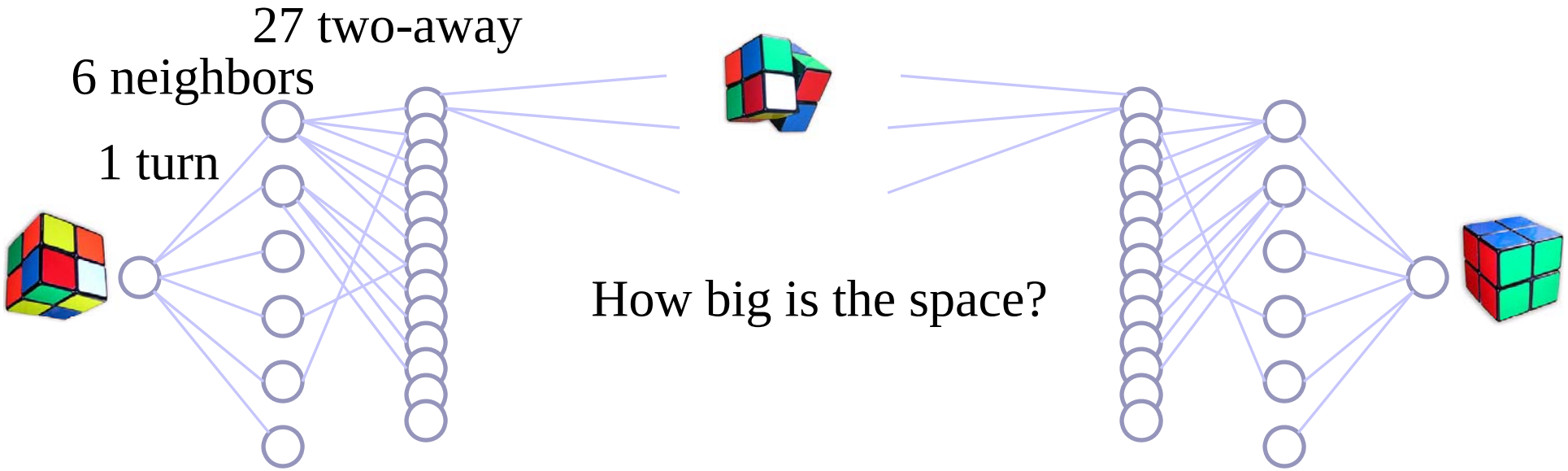
# Implicit representation

- Don’t store graph at all
- Implement function Adj(u) that returns list of neighbors or edges of u
- Requires no space, use it as you need it
- And may be very efficient
- e.g., Rubik’s cube

# Back to the Rubik’s cube game

Searching graphs

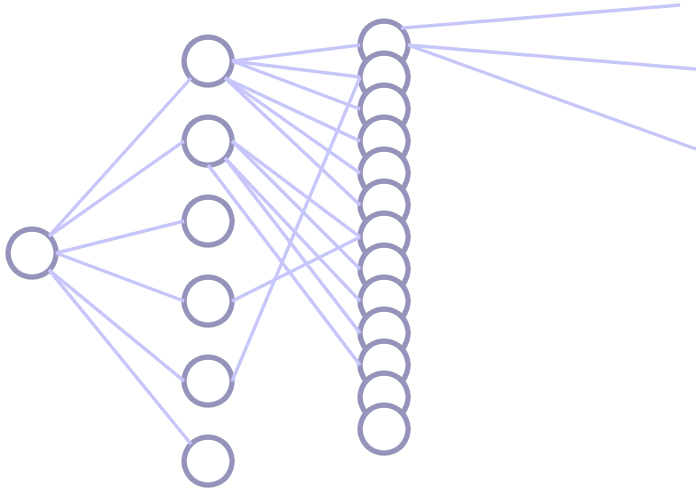
# Searching for a solution path




- Graph algorithms allow us explore space
  - Nodes: configurations
  - Edges: moves between them
  - Paths to ‘solved’ configuration: solutions

# The lay of the land (geography)

- 6 vertices reachable by one 90° turn
- 9 vertices reachable by one 90° or 180° turn
- To reach furthest node, 11 or 14 moves needed



| distance | 90° | 90° and 180° |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 6 | 9 |
| 2 | 27 | 54 |
| 3 | 120 | 321 |
| 4 | 534 | 1847 |
| 5 | 2,256 | 9,992 |
| 6 | 8,969 | 50,136 |
| 7 | 33,058 | 227,526 |
| 8 | 114,149 | 870,072 |
| 9 | 360,508 | 1,887,748 |
| 10 | 930,588 | 623,800 |
| 11 | 1,350,852 | 2,644 |
| 12 | 782,536 | |
| 13 | 90,280 | |
| 14 | 276 | |

diameter

# Unit #4 Overview: Searching

**Today: Introduction to Games and Graphs**

- Rubik's cube, Pocket cube, Game space
- Graph definitions, representation, searching

**Tuesday: Graph algorithms and analysis**

- Breadth First Search, Depth First Search
- Queues, Stacks, Augmentation, Topological sort

**Thursday: Networks in biology and real world**

- Network/node properties, metrics, motifs, clusters
- Dynamic processes, epidemics, growth, resilience

# Unit #4 – Games, Graphs, Searching, Networks

| Unit | Pset | Week | Date | | Lecture (Tuesdays and Thursdays) | | Recitation (Wed and Fri) |
|---|---|---|---|---|---|---|---|
| Intro | PS1 | 1 | Tue Feb 01 | 1 | Introduction and Document Distance | 1 | Python and Asymptotic Complexity |
| Binary | Out: 2/1 | | Thu Feb 03 | 2 | Peak Finding Problem | 2 | Peak Finding correctness & analysis |
| Search | Due: Mon 2/14 | 2 | Tue Feb 08 | 3 | Scheduling and Binary Search Trees | 3 | Binary Search Tree Operations |
| Trees | HW lab: Sun 2/13 | | Thu Feb 10 | 4 | Balanced Binary Search Trees | 4 | Rotations and AVL tree deletions |
| Hashing | PS2 Out: 2/15 | 3 | Tue Feb 15 | 5 | Hashing I : Chaining, Hash Functions | 5 | Hash recipes, collisions, Python dicts |
| | Due: Mon 2/28 | | Thu Feb 17 | 6 | Hashing II : Table Doubling, Rolling Hash | 6 | Probability review, Pattern matching |
| | HW lab:Sun 2/27 | 4 | Tue Feb 22 | - | President's Day - Monday Schedule - No Class | - | No recitation |
| | | | Thu Feb 24 | 7 | Hashing III : Open Addressing | 7 | Universal Hashing, Perfect Hashing |
| Sorting | PS3. Out: 3/1 | 5 | Tue Mar 01 | 8 | Sorting I : Insertion & Merge Sort, Master Theorem | 8 | Proof of Master Theorem, Examples |
| | Due: Mon 3/7 | | Thu Mar 03 | 9 | Sorting II : Heaps | 9 | Heap Operations |
| | HW lab: Sun 3/6 | 6 | Tue Mar 08 | 10 | Sorting III: Lower Bounds, Counting Sort, Radix Sort | 10 | Models of computation |
| | | | Wed Mar 09 | Q1 | Quiz 1 in class at 7:30pm. Covers L1-R10. Review Session on Tue 3/8 at 7:30pm. | | |
| Graphs | PS4. Out: 3/10 | | Thu Mar 10 | 11 | Searching I: Graph Representation, Depth-1st Search | 11 | Strongly connected components |
| and | Due: Fri 3/18 | 7 | Tue Mar 15 | 12 | Searching II: Breadth-1st Search, Topological Sort | 12 | Rubik's Cube Solving |
| Search | HW lab:W 3/16 | | Thu Mar 17 | 13 | Searching III: Games, Network properties, Motifs | 13 | Subgraph isomorphism |
| Shortest | PS5 | 8 | Tue Mar 29 | 14 | Shortest Paths I: Introduction, Bellman-Ford | 14 | Relaxation algorithms |
| Paths | Out: 3/29 | | Thu Mar 31 | 15 | Shortest Paths II: Bellman-Ford, DAGs | 15 | Shortest Path applications |
| | Due: Mon 4/11 | 9 | Tue Apr 05 | 16 | Shortest Paths III: Dijkstra | 16 | Speeding up Dijkstra's algorithm |
| | HW lab:Sun 4/10 | | Thu Apr 07 | 17 | Graph applications, Genome Assembly | 17 | Euler Tours |
| Dynamic | PS6 | 10 | Tue Apr 12 | 18 | DP I: Memoization, Fibonacci, Crazy Eights | 18 | Limits of dynamic programming |
| Program | Out: Tue 4/12 | | Wed Apr 13 | Q2 | Quiz 2 in class at 7:30pm. Covers L11-R17. Review Session on Tue 4/13 at 7:30pm. | | |
| ming | Due: Fri 4/29 | | Thu Apr 14 | 19 | DP II: Shortest Paths, Genome sequence alignment | 19 | Edit Distance, LCS, cost functions |
| | HW lab:W 4/27 | 11 | Tue Apr 19 | - | Patriot's Day - Monday and Tuesday Off | - | No recitation |
| | | | Thu Apr 21 | 20 | DP III: Text Justification, Knapsack | 20 | Saving Princess Peach |
| | | 12 | Tue Apr 26 | 21 | DP IV: Piano Fingering, Vertex Cover, Structured DP | 21 | Phylogeny |
| Numbers | PS7 out Thu4/28 | | Thu Apr 28 | 22 | Numerics I - Computing on large numbers | 22 | Models of computation return! |
| Pictures | Due: Fri 5/6 | 13 | Tue May 3 | 23 | Numerics II - Iterative algorithms, Newton's method | 23 | Computing the nth digit of π |
| (NP) | HW lab: Wed 5/4 | | Thu May 5 | 24 | Geometry: Line sweep, Convex Hull | 24 | Closest pair |
| | | 14 | Tue May 10 | 25 | Complexity classes, and reductions | 25 | Undecidability of Life |
| Beyond | | | Thu May 12 | 26 | Research Directions (15 mins each) + related classes | | |
| | | 15 | Finals week | Q3 | Final exam is cumulative L1-L26. Emphasis on L18-L26. Review Session on Fri 5/13 at 3pm | | |

# Conclude

- Graphs: fundamental data structure
  - Directed and undirected
- 4 possible representations
- Basic methods of graph search

- Next time:
  - Formalize BFS and DFS
  - Runtime analysis
  - Applications

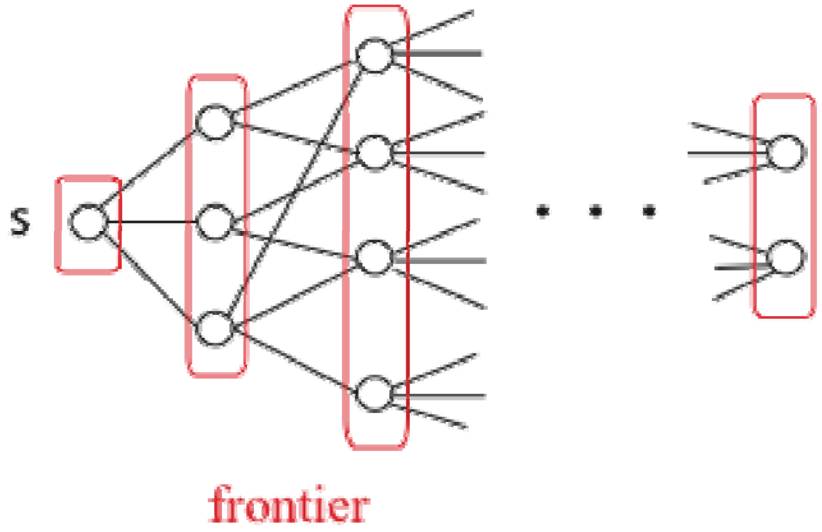
# Graph Searching Algorithms

We want to get from current Rubik state to “solved” state

How do we explore?

# Breadth First Search

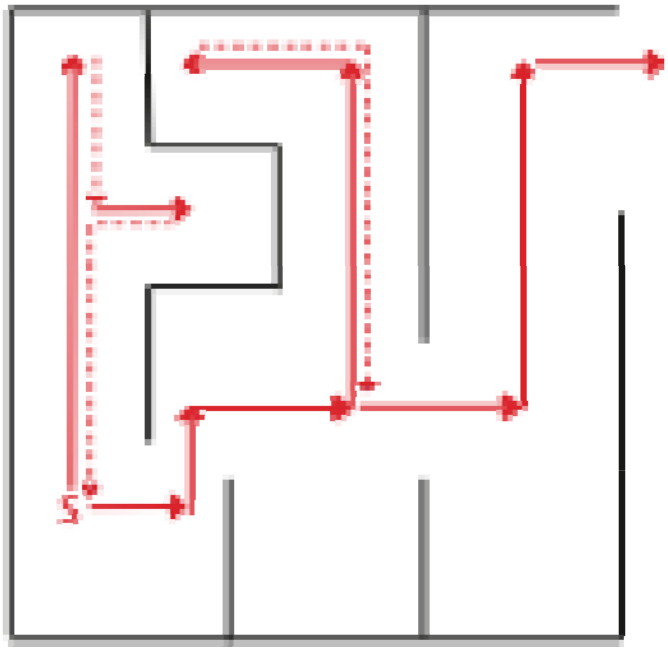
- start with vertex v
- list all its neighbors (distance 1)
- then all their neighbors (distance 2)
- etc.




- algorithm starting at s:
  - define frontier F
  - initially F={s}
  - repeat F=all neighbors of vertices in F
  - until all vertices found

# Depth First Search

- Like exploring a maze
- From current vertex, move to another
- Until you get stuck
- Then backtrack till you find a new place to explore
- e.g “left-hand” rule



# Problem: Cycles

- What happens if unknowingly revisit a vertex?
- BFS: get wrong notion of distance
- DFS: go in circles
- Solution: mark vertices
  - BFS: if you've seen it before, ignore
  - DFS: if you've seen it before, back up