Problem Solving Techniques 문제해결

Jinkyu Lee

Dept. of Computer Science and Engineering,
Sungkyunkwan University (SKKU)

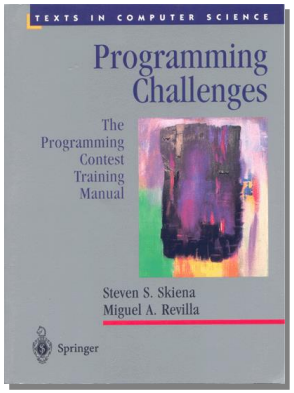
Instructor information

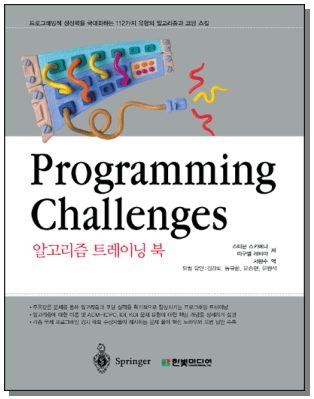
- Jinkyu Lee 이진규
  - Associate Professor in Department of Computer Science and Engineering (March 2014~)
  - Research areas
    - Real-time scheduling and systems
    - Software-defined battery management
    - Mobile computing and systems
- Contact information
  - jinkyu.lee@skku.edu
  - https://rtclskku.github.io/website/jinkyulee.html
  - https://rtclskku.github.io/website
  - #27322B, Engineering 2
  - Office hour: by appointment

Course information

- Tuesday 09:00 – 10:15, Thursday 10:30 – 11:45
- 3 credits
- Reference: Programming Challenges (by S. Skiena, M. Revilla)
  - You don't need to buy this book.
- Language: **English**
- Online recorded lecture (No Tuesday 09:00 real-time class)
- Offline lecture and discussion (Thursday 10:30 – 11:45): in case of severe spread of COVID-19 (Stage 3 or 4), offline lecture will be replaced by online real-time streaming class through Zoom
- Prerequisite: **C programming language**
  - You SHOULD know how to program using C (e.g., pointer)
- For the second-grade students
- Course website
  - http://icampus.skku.edu
- Questions
  - jinkyu.lee@skku.edu
    - Title: [SWE2026] Student id, name

http://lib.skku.edu





What is this course?

- Study problem solving techniques for computer science and engineering
  - Deal with the problems creatively and effectively
  - Develop thinking skills in new and unfamiliar situations
  - Enhance mathematical concepts and skills
  - Implement own ideas using C programming language
- Designed for those who have not taken the algorithm course
  - Rather than solving problems using stereotypical algorithm theories, this course encourages students to create their own solutions.
  - Do not take this course if you already took the algorithm class.
- Problem-based course
  - **The course seems unorganized!**
- Non-technical goals
  - Know your departmental students and make friends
  - Present your idea to your classmates

Weakly Flipped Learning Class

- Online recorded lecture (No Tuesday 09:00 real-time class)
  - Study of problem-solving strategies

- Offline lecture (Thursday 10:30 – 11:45)
  - Explanation of Exercise and Homework
  - Discussion of students' solutions for Exercise and Homework
  - Discussion of students' solutions for in-class problems
  - Further explanation of recorded lecture

Grading

- Attendance + attitude + participation: 5%
  - You will be given F if you are absent ten times or more.
  - For each absence, you will lose 0.5% (out of 5%)
  - Alternative attendance approval: to follow univ. rule
- Presentation: 10% (+ more)
  - Presentation of your solutions (Thursday offline class)
  - 7% + 2% + 1% + α% + α% + … (your solution is not necessarily perfect!)
- Assignment: 55%
  - 3~4 individual homework (to be submitted and evaluated in detail)
  - Some exercises (to be submitted and evaluated as Pass/Fail)
- No mid-term exam
- Final exam: 30%
  - 6/1 10:15-12:00 Offline, closed-book
- Cheating will lead you to fail this course with “F” grade.

Presentation

- Presentation: 10% + more
  - Presentation of your solutions (Thursday offline class)
  - 7% + 2% + 1% + α% + α% + … (your solution is not necessarily perfect!)
  - α=0 initially; if there are not many volunteer, α will be increased.

- Presentation priority
  - Those who did not present his/her solution

    >
  - Those who presented his/her solution once

    >
  - Those who presented his/her solution twice

    >
  - Those who presented his/her solution three times or more

Tentative Topics

- C-programming overview
- Tips for programming
- Sorting
- String
- Backtracking
- Grid
- Graph traversal
- Dynamic programming
- Combinatorics

Tentative schedule

Date and Contents	Announcement
2/28, 3/2 Course introduction, Strategies for problem solving	Exercise A
3/7, 3/9 C-programming overview, Discussion: Exercise A	Exercise B Homework 1
3/14, 3/16 Tips for programming, Discussion: Exercise B	
3/21, 3/23 Sorting, Discussion: Homework 1	Exercise C
3/28, 3/30 Strings, Discussion: Exercise C	Homework 2
4/4, 4/6 Backtracking	
4/11, 4/13 Grid, Discussion: Homework 2	Exercise D
4/18, 4/20 Reserved	Homework 3

Tentative schedule

Date and Contents	Announcement
4/25, 4/27 Graph traversal, Discussion: Exercise D	
5/2, 5/4 Graph traversal, Discussion: Homework 3	Exercise E
5/9, 5/11 Dynamic programming, Discussion: Exercise E	Homework 4
5/16, 5/18 Dynamic programming	
5/23, 5/25 Combinatorics, Discussion: Homework 4	
5/30, 6/1 Reserved 6/1 Final Exam: 6/1 10:15 – 12:00 (Offline, closed-book)	
6/6, 6/8 Reserved	

Cheating examples

```
void minimize(int num_task, int day[], int fine[], int order[]) {
        int i,j;
        int temp = 0;
        float temp1,temp2 = 0;
        for (i = 0; i < num_task; i++){
                for (j = 0; j < num_task; j++){
                        temp1 = (float)fine[j] / (float)(day[j]);
                        temp2 = (float)(fine[j+1]) / (float)(day[j+1]);
                        if (temp1 < temp2){
                                temp = day[j];
                                day[j] = day[j + 1];
                                day[j + 1] = temp;
                                temp = fine[j];
                                fine[j] = fine[j + 1];
                                fine[j + 1] = temp;
                                temp = order[j];
                                order[j] = order[j+1];
                                order[j + 1] = temp;
                        }
                }
        }

}
```

```
void minimize(int num_task, int day[], int fine[], int order[]) {
        int i, j;
        int swap = 0;
        float priority1, priority2 = 0;
        for (i = 0; i < num_task; i++){
                for (j = 0; j < num_task; j++){
                        priority1 = (float)fine[j] / (float)(day[j]);
                        priority2 = (float)(fine[j + 1]) / (float)(day[j + 1]);
                        if (priority1 < priority2){
                                swap = day[j];
                                day[j] = day[j + 1];
                                day[j + 1] = swap;
                                swap = fine[j];
                                fine[j] = fine[j + 1];
                                fine[j + 1] = swap;
                                swap = order[j];
                                order[j] = order[j + 1];
                                order[j + 1] = swap;
                        }
                }
        }

}
```

C program examples

```
#define NCARDS  52      /* number of cards */
#define NSUITS  4       /* number of suits */

char values[] = "23456789TJQKA";
char suits[] = "cdhs";

int rank_card(char value, char suit)
{
        int i,j;        /* counters */

        for (i=0; i<(NCARDS/NSUITS); i++)
                if (values[i]==value)
                        for (j=0; j<NSUITS; j++)
                                if (suits[j]==suit)
                                        return( i*NSUITS + j );

        printf("Warning: bad input value=%d, suit=%d\n",value,suit);
}
```

```
init_queue(queue *q)
{
        q->first = 0;
        q->last = QUEUESIZE-1;
        q->count = 0;
}
```

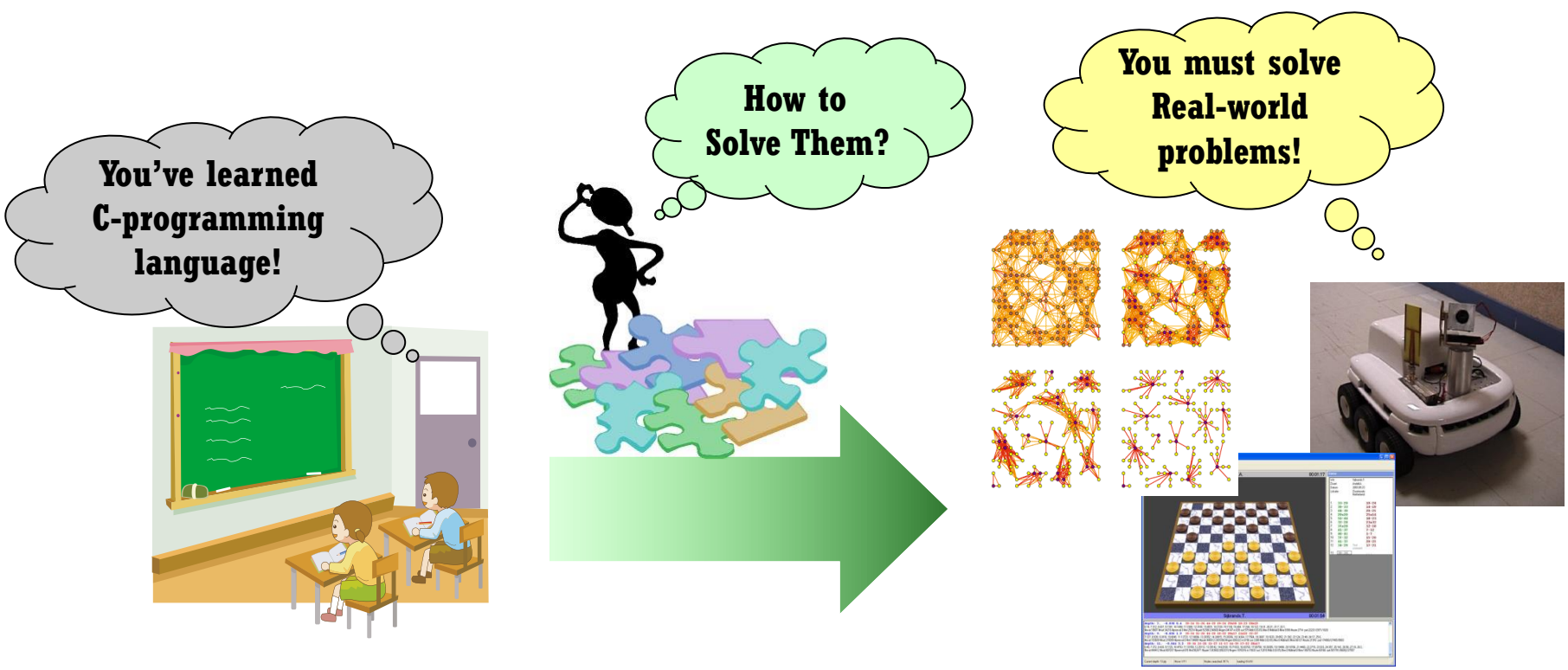
```
int empty(queue *q)
{
        if (q->count <= 0) return (TRUE);
        else return (FALSE);
}
```

Contents

- **Problem solving overview**
- Strategies for problem solving

Some slides are adapted from Prof. Chang Wook Ahn's slides.

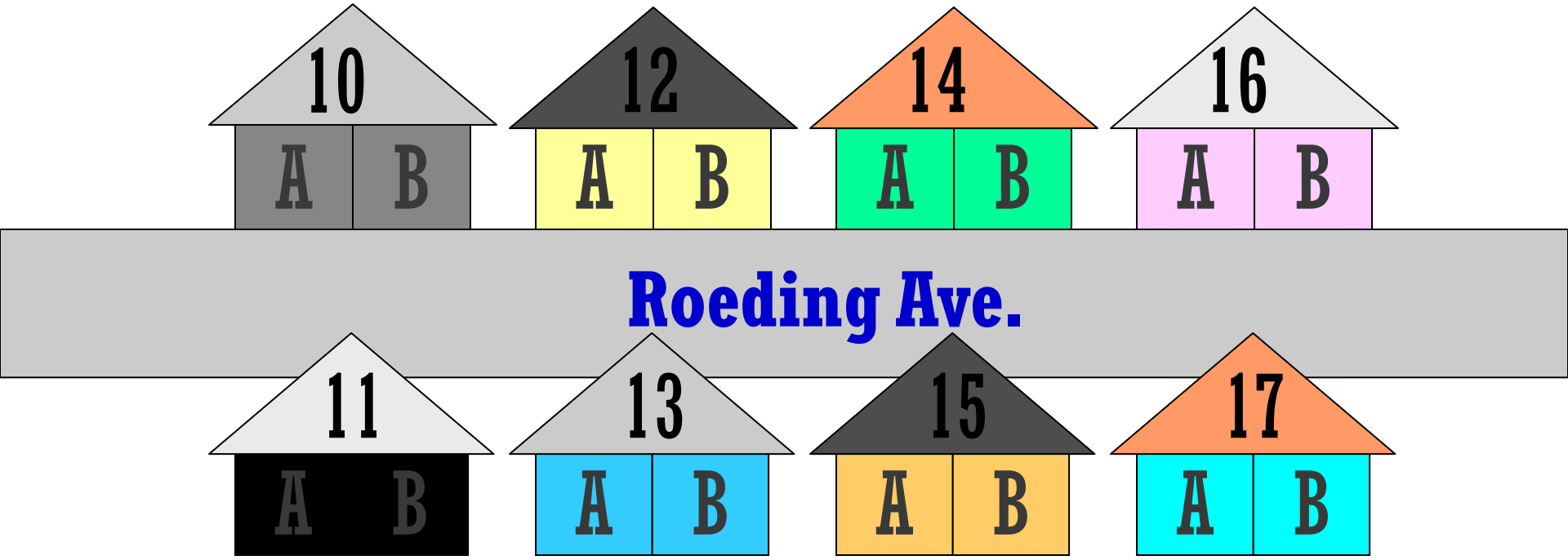
Why Are You Taking this Course?
What Are You Expecting…?



Problem Solving: Overview (1)

How to Solve the Problem?

- Jose's friend Juan lives in a duplex at 17A Roeding Ave. Jose is going to visit Juan. Here is a map of Juan's block. Find where Juan lives.



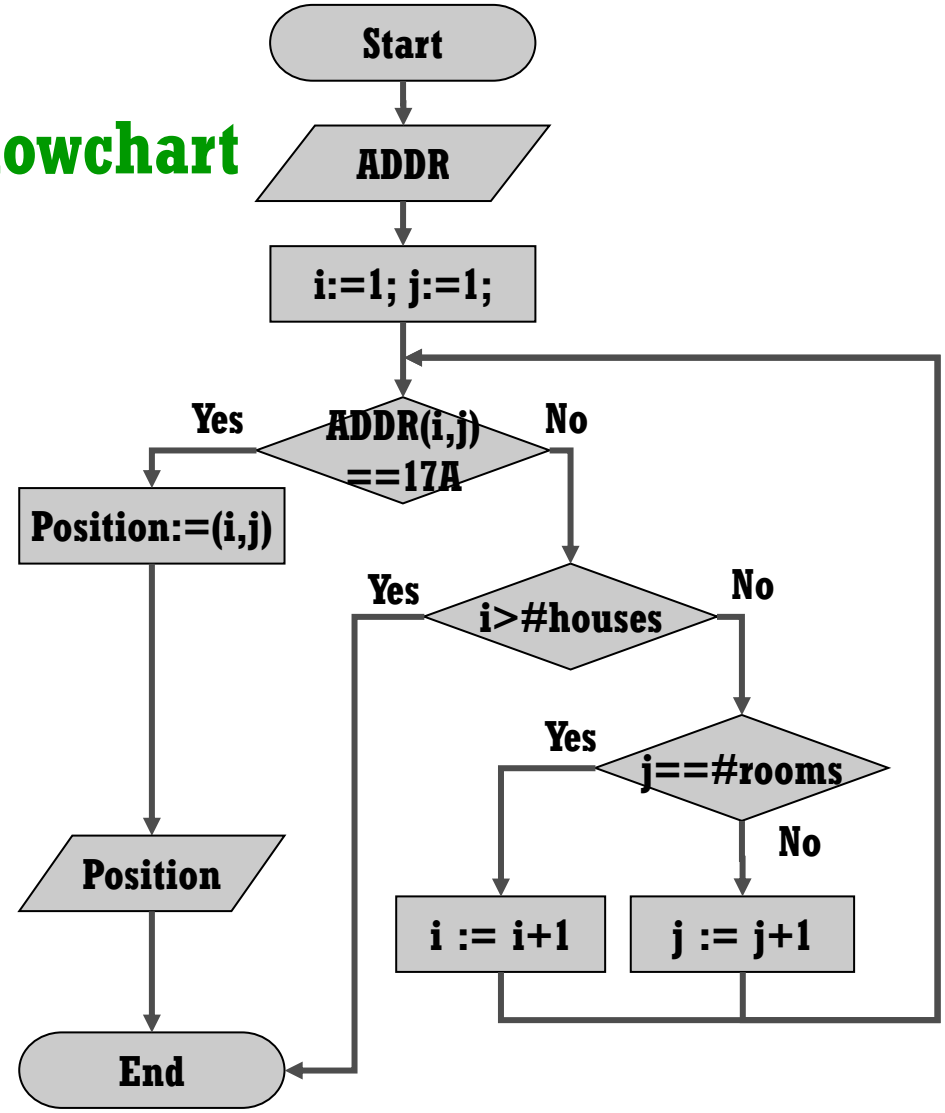
Problem Solving: Overview (2)

- **Brute-Forth Method**

Pseudocode

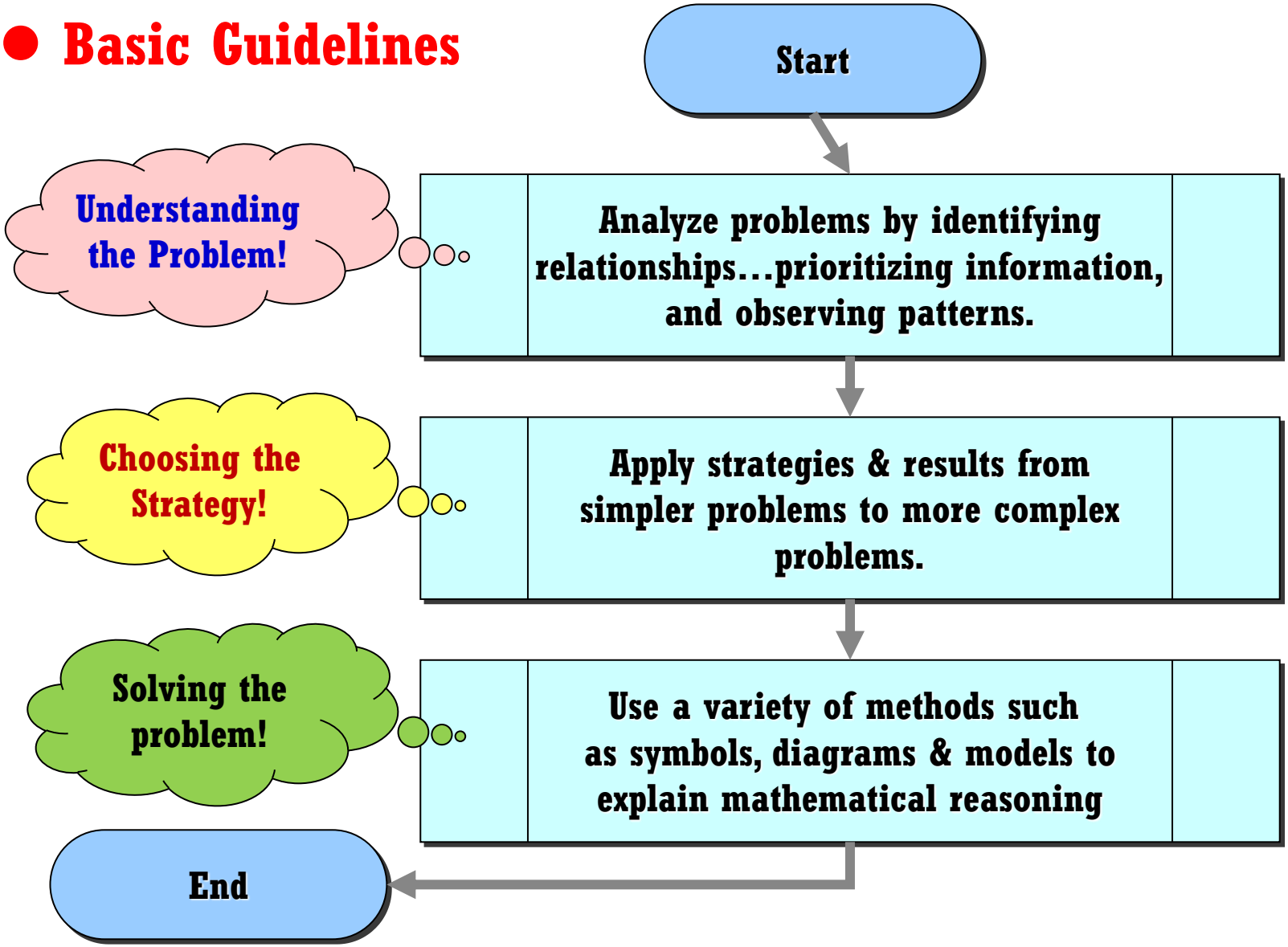
```
Input: ADDR (i.e., Data on the house addresses)
Output: Position of Juan's house (17A)

FOR i := 1 to #of houses
    FOR j := 1 to #of rooms (in each house)
        IF ADDR(i,j) == 17A
            Position := (i,j);
        END
    END
END

return Position;
```

Flowchart



Problem Solving: Overview (3)

- **Basic Guidelines**



Problem Solving: Overview (4)

- Problem Solving is easy if the following steps are used.

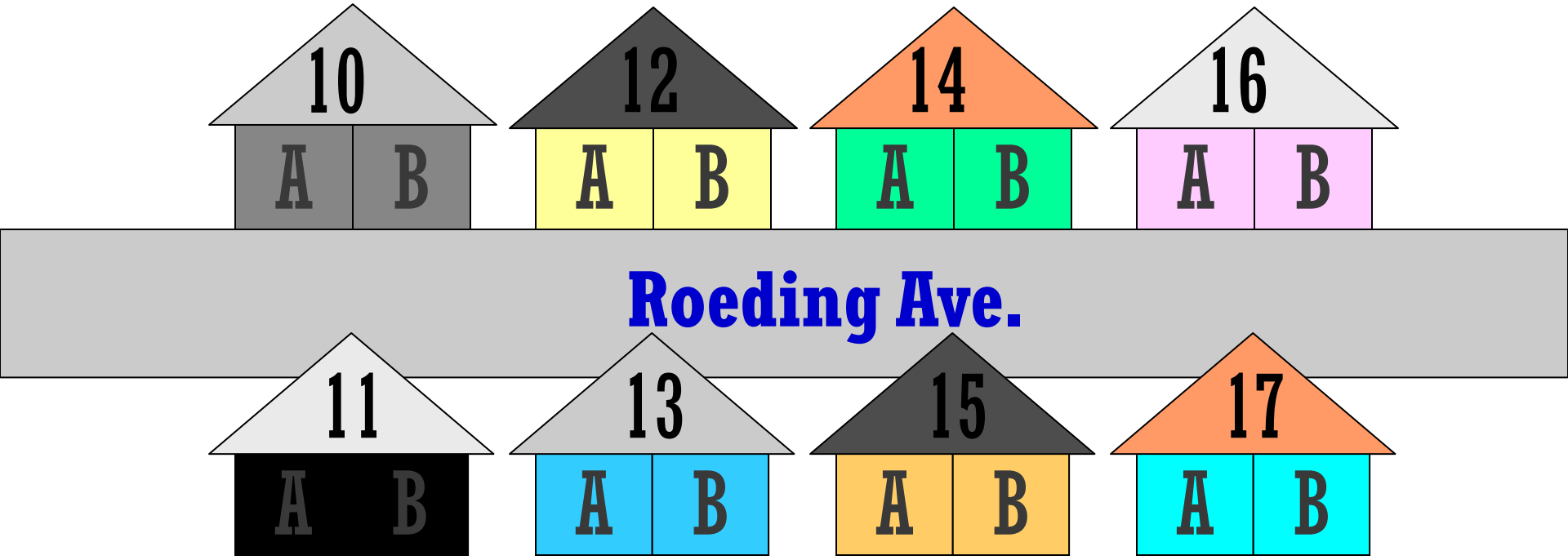
Problem Solving: Overview (5)

- **Step 1: Understand the Problem**
  - Read the problem carefully.
  - Find the important information.
  - Look for patterns.
  - Identify what the problem wants you to solve.

Problem Solving: Overview (6)

Read the Problem Carefully.

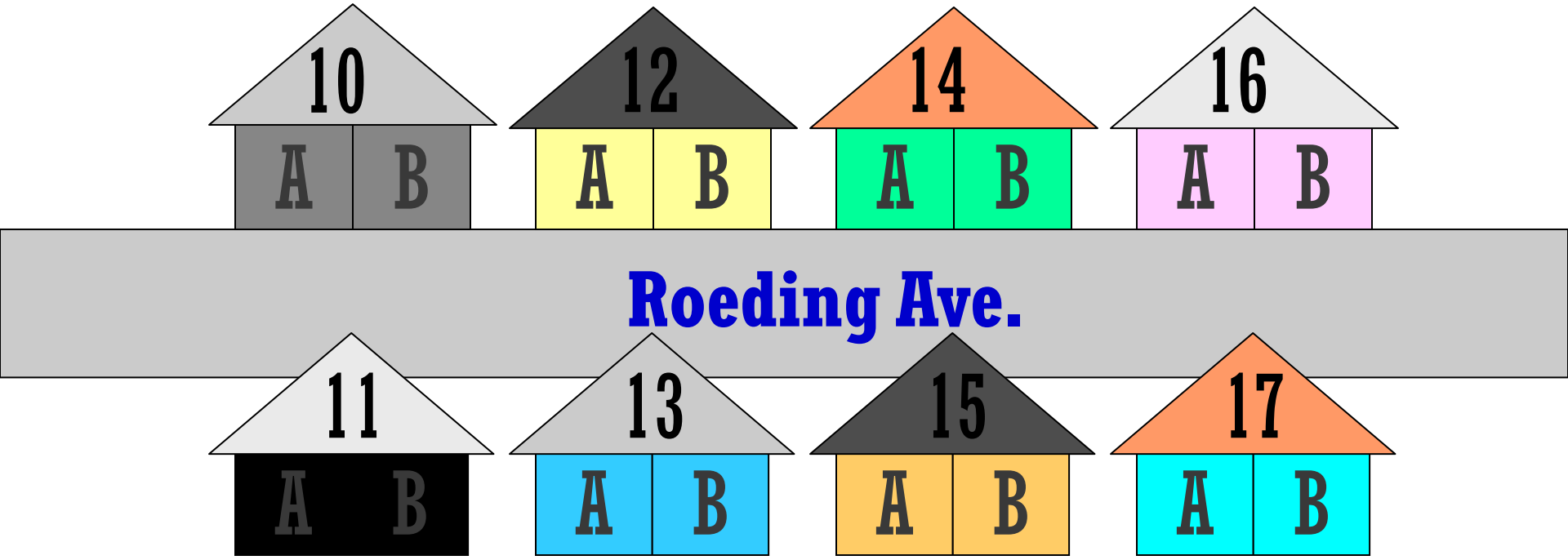
- **Jose's friend Juan lives in a duplex at 17A Roeding Ave. Jose is going to visit Juan. Here is a map of Juan's block. Find where Juan lives.**



Problem Solving: Overview (7)

Find the Important Information.

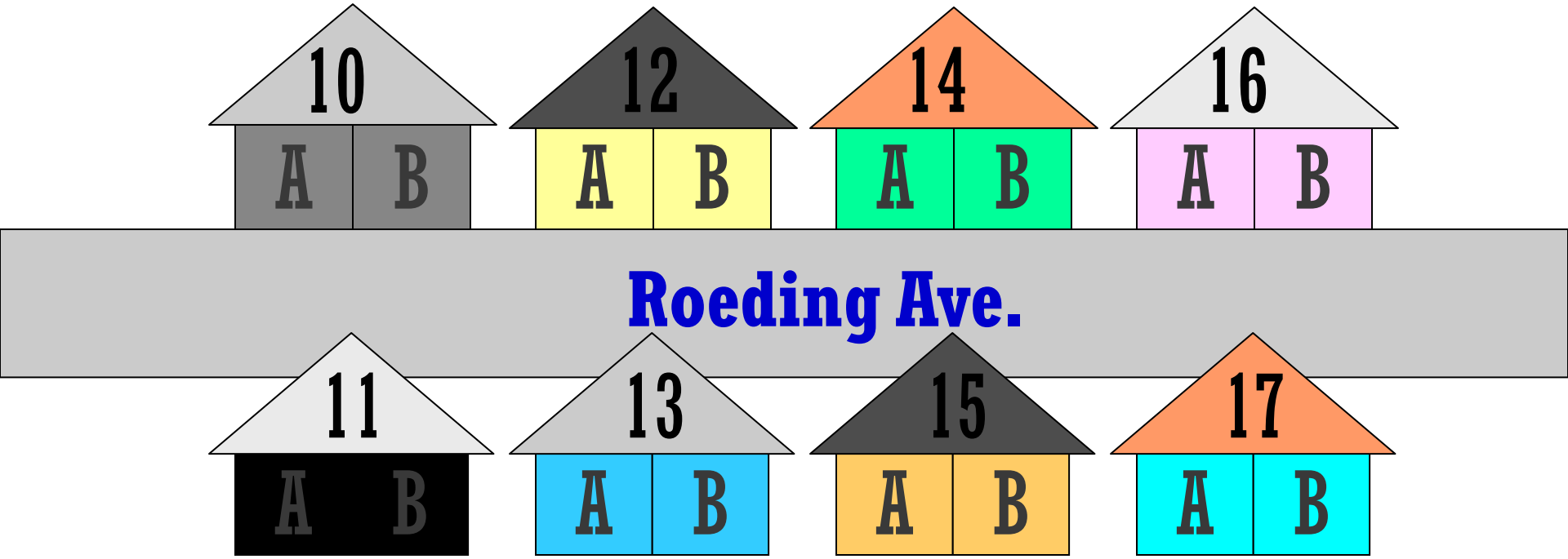
- Jose's friend Juan lives in a duplex at **17A Roeding Ave.** Jose is going to visit Juan. Here is a map of Juan's block. Find where Juan lives.



Problem Solving: Overview (8)

Write down the Key Information.

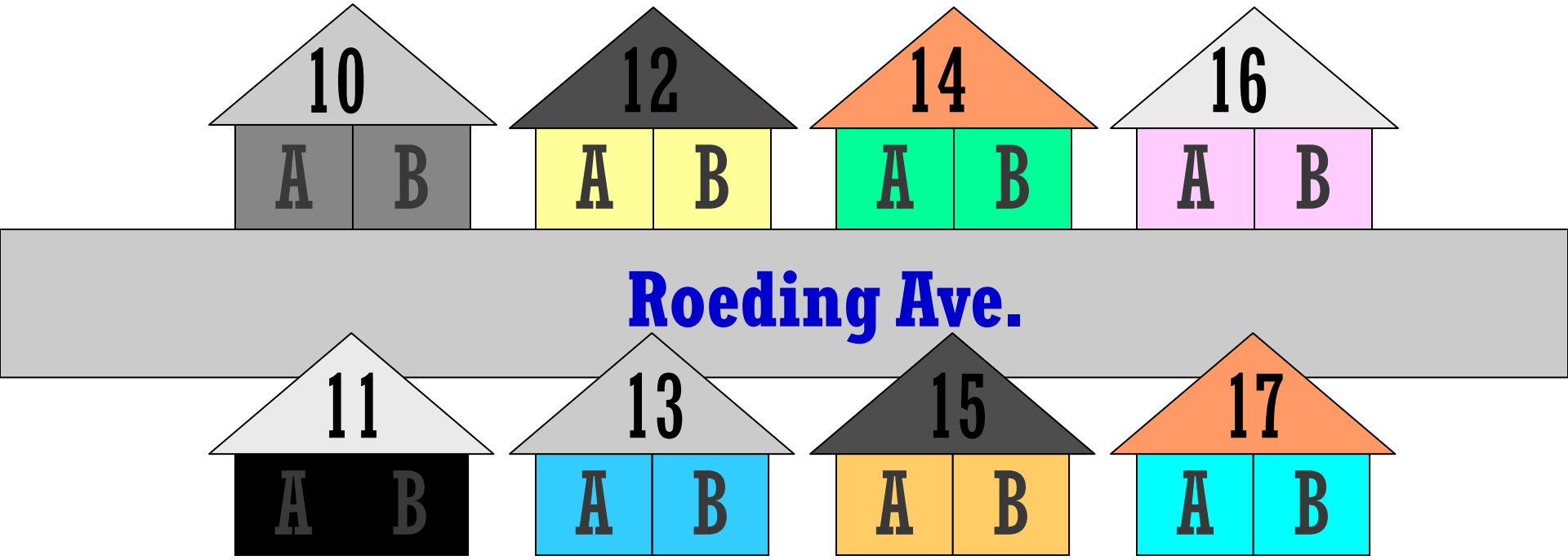
17A Roeding Ave.



Problem Solving: Overview (9)

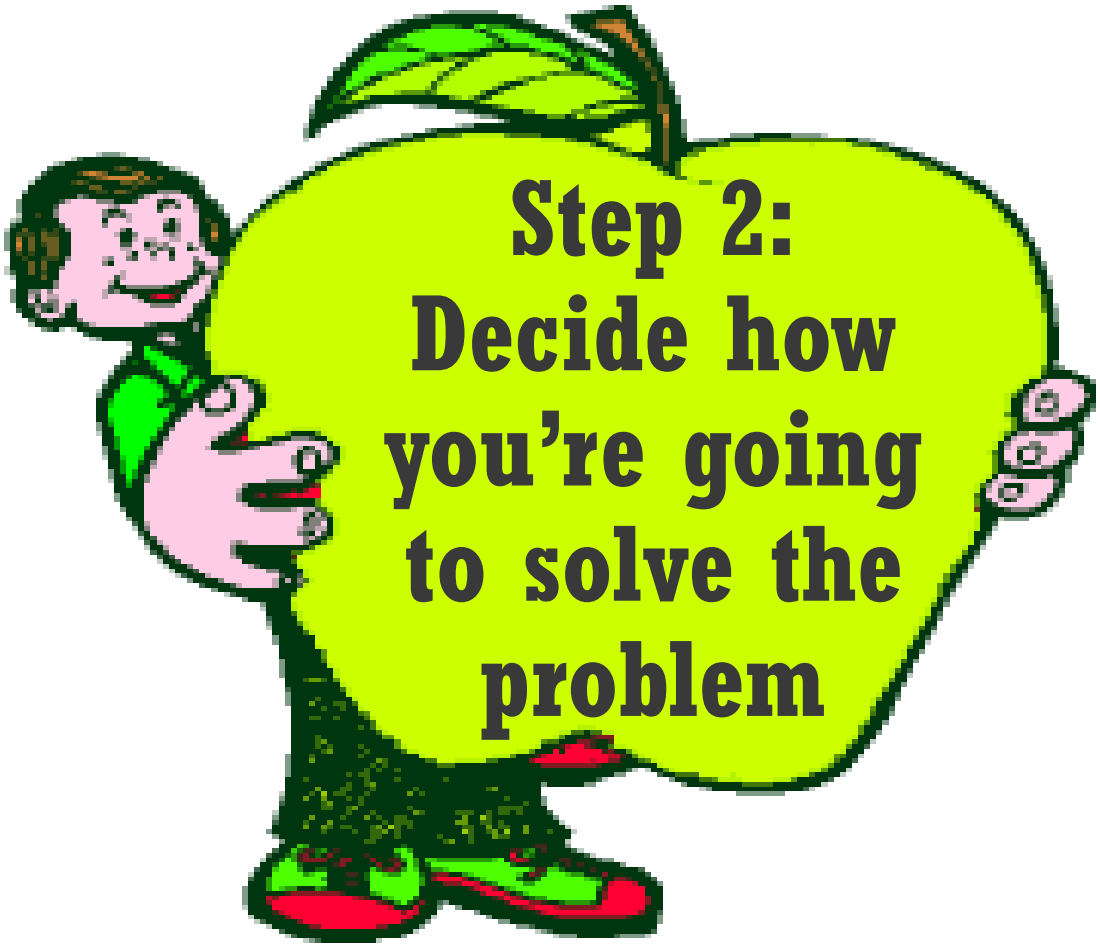
Identify What the Problem wants you to Solve.

- Jose's friend Juan lives in a duplex at **17A Roeding Ave.** Jose is going to visit Juan. Here is a map of Juan's block. **Find where Juan lives.**



Problem Solving: Overview (10)

- Problem Solving is easy if the following steps are used.



Problem Solving: Overview (11)

- **Step 2: Decide the Method to Solve the Problem**
  - ☞ Use a graph
  - ☞ Write an equation
  - ☞ **Find a pattern**
  - ☞ Use reasoning
  - ☞ Make a table
  - ☞ Use formulas
  - ☞ Make a list
  - ☞ Work backwards
  - ☞ Draw a picture
  - ☞ Act it out

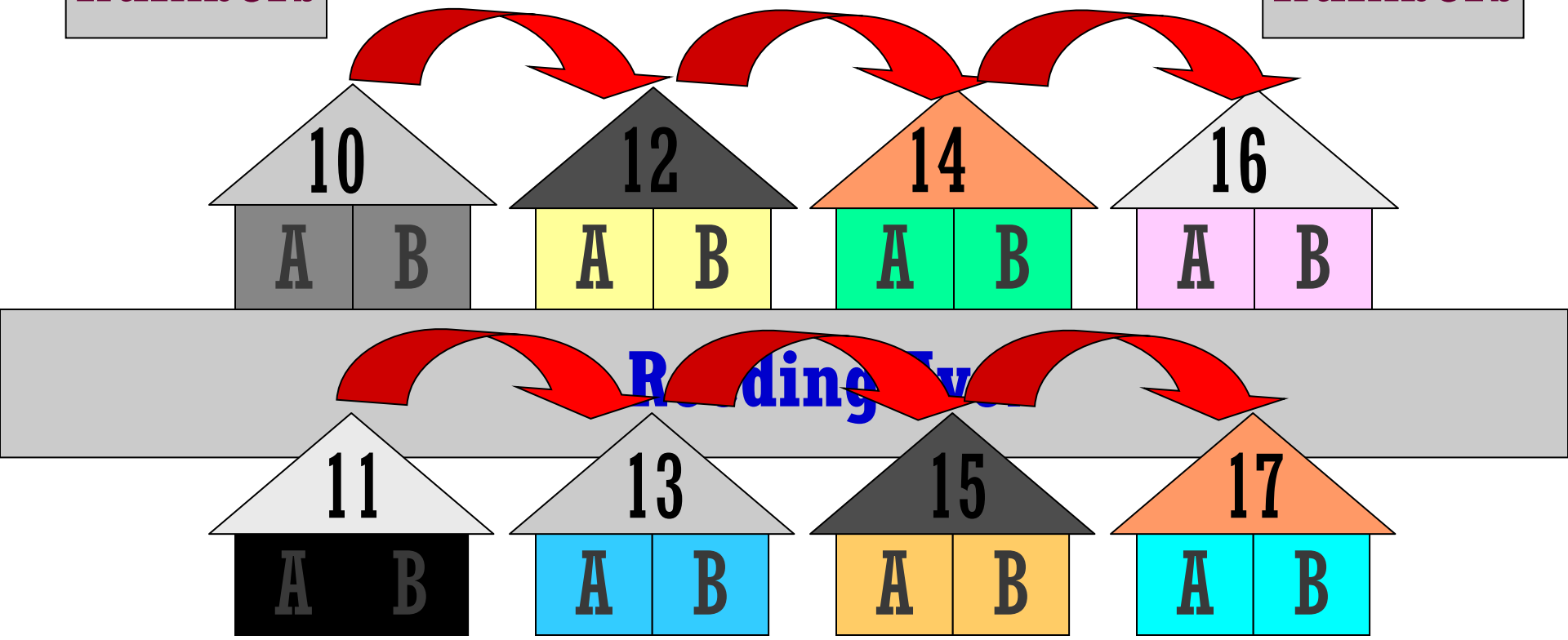
Problem Solving: Overview (12)

Look for the Pattern.

By twos even numbers

17A Roeding Ave.

By twos odd numbers



Problem Solving: Overview (13)

- Problem Solving is easy if the following steps are used.

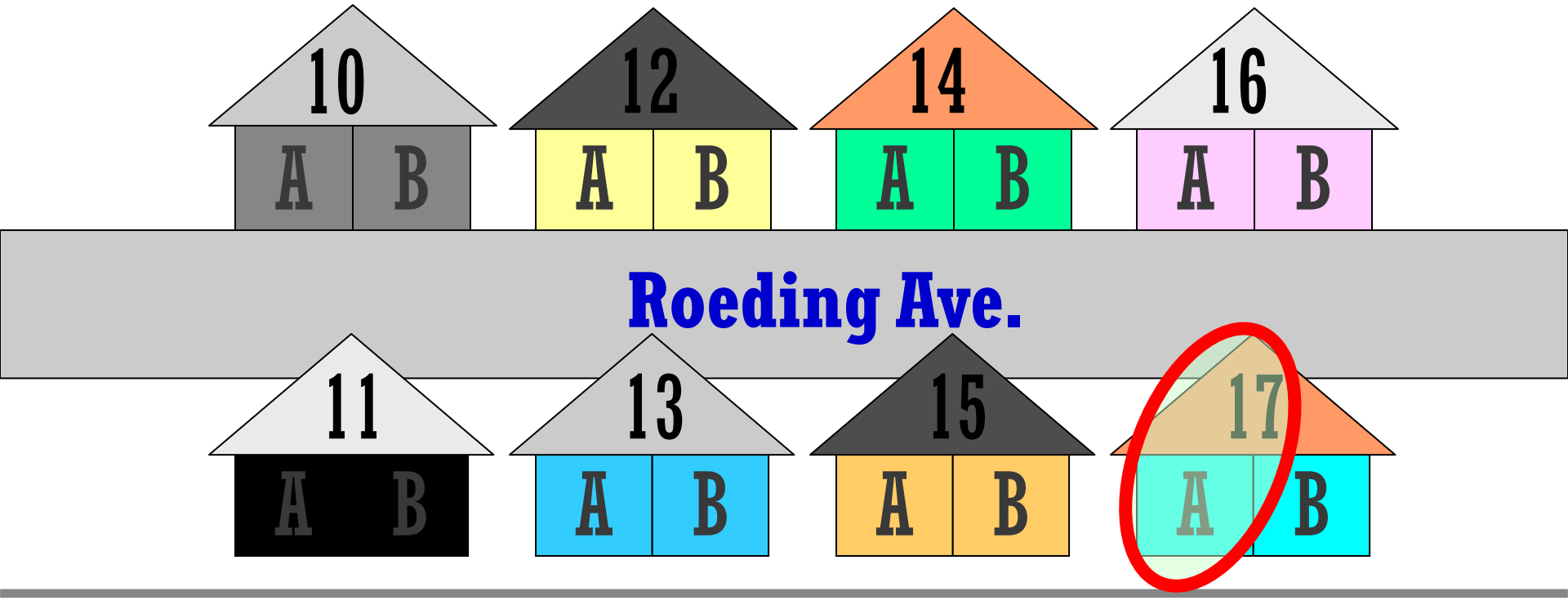
Problem Solving: Overview (14)

Find the Juan's House.

You know 17 is odd.

17A Roeding Ave.

You know he lives in A.



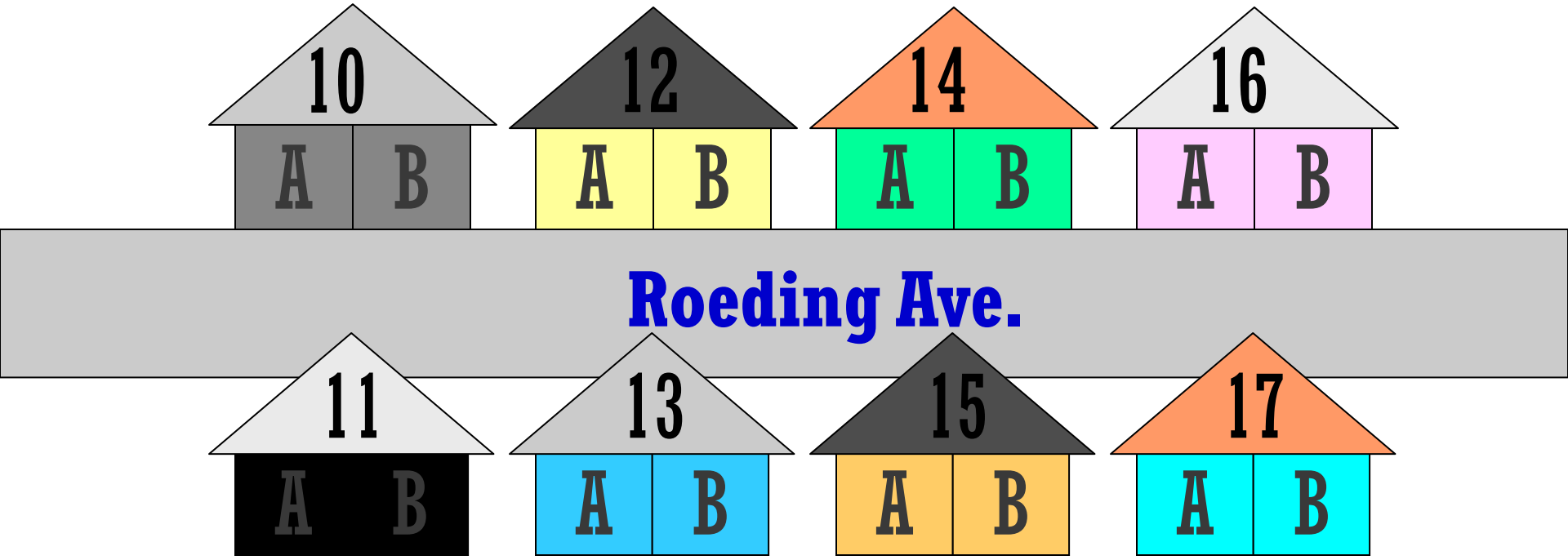
Problem Solving: Overview (15)

❖ Problem Solving is easy if the following steps are used.

Problem Solving: Overview (16)

Read the Problem Again.

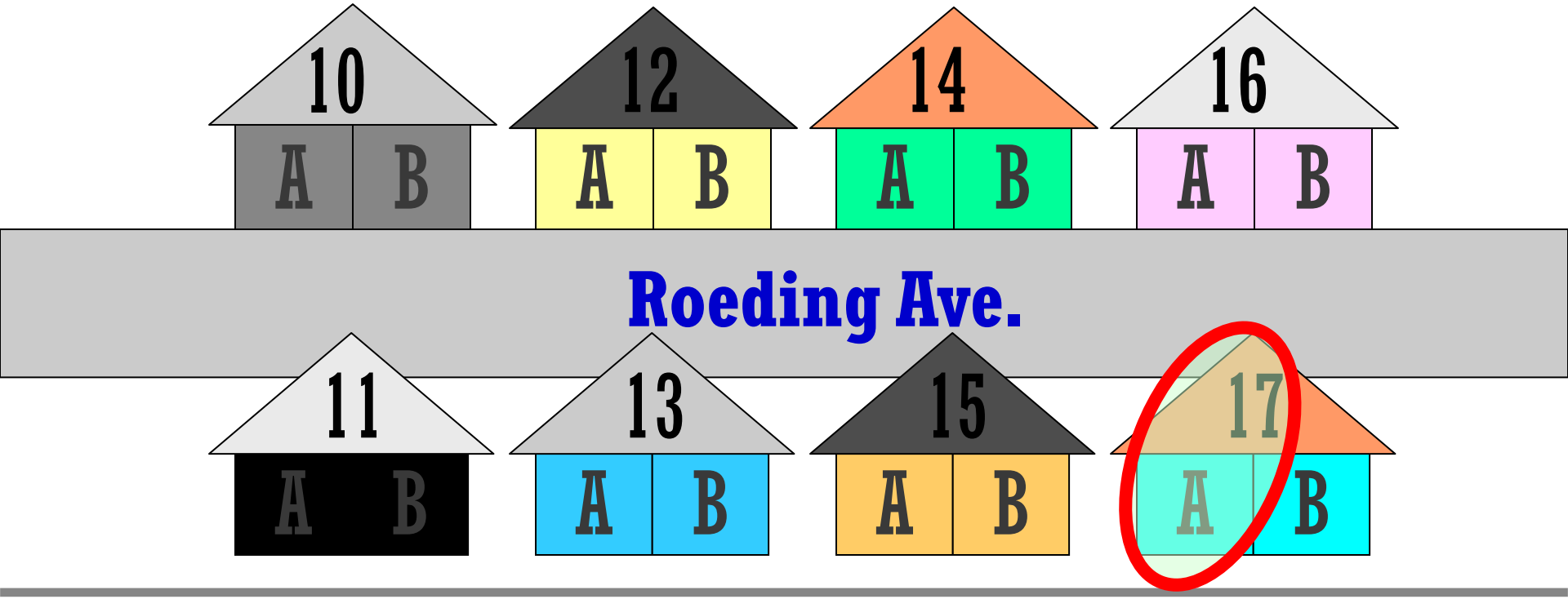
- **Jose's friend Juan lives in a duplex at 17A Roeding Ave. Jose is going to visit Juan. Here is a map of Juan's block. Find where Juan lives.**



Problem Solving: Overview (17)

Compare your Answer to the Problem.

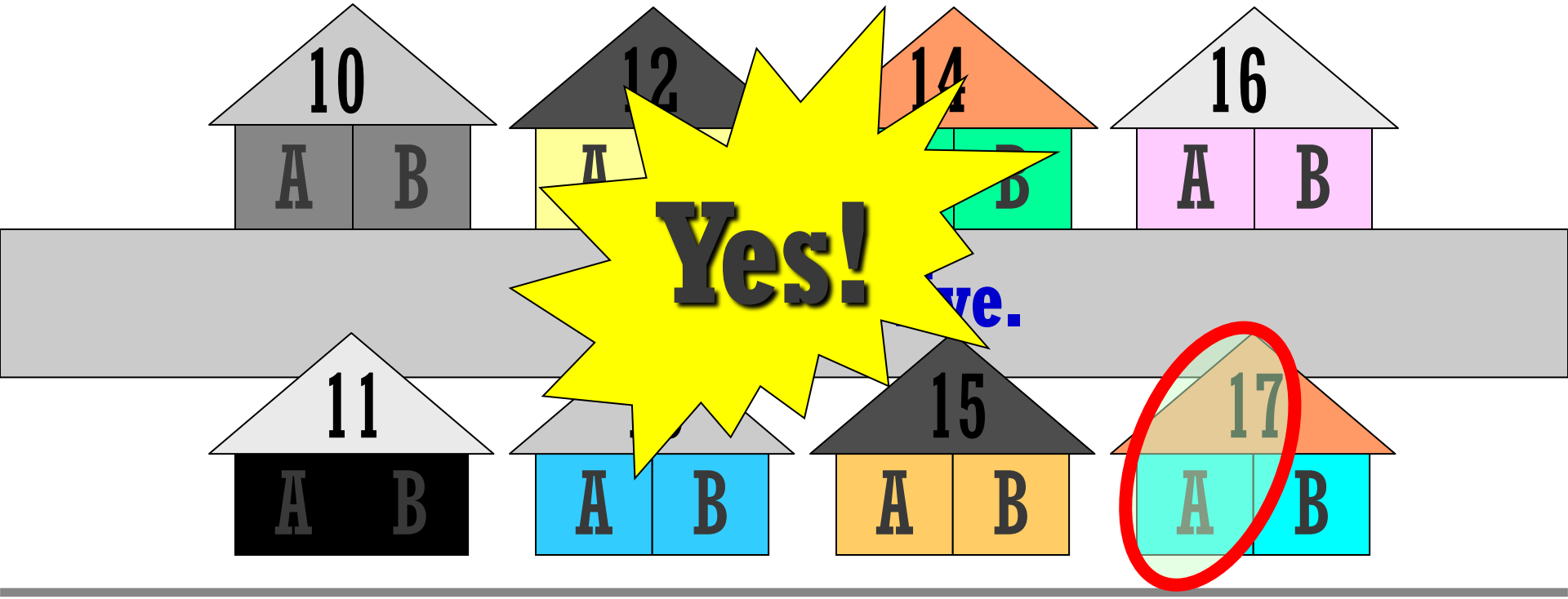
- Jose's friend Juan lives in a duplex at **17A Roeding Ave.** Jose is going to visit Juan. Here is a map of Juan's block. Find where Juan lives.



Problem Solving: Overview (18)

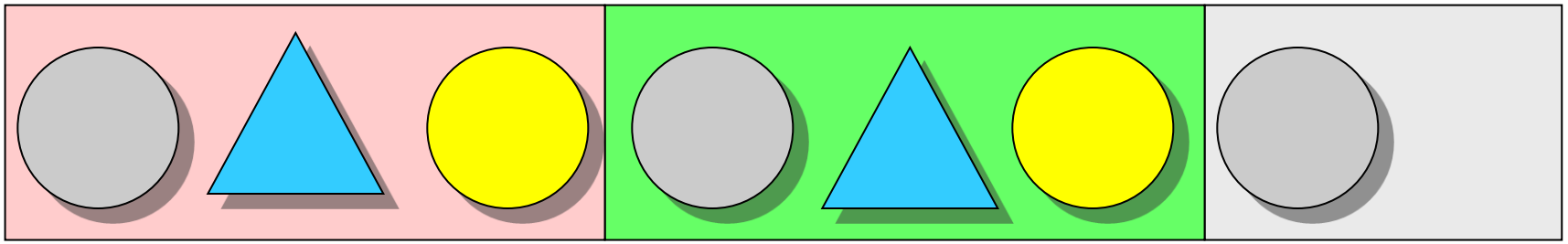
Did You Solve What the Problem Asked?

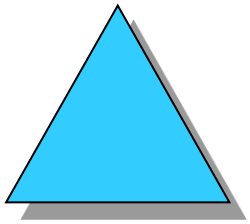
- Jose's friend Juan lives in a duplex at 17A Roeding Ave. Jose is going to visit Juan. Here is a map of Juan's block. Find where Juan lives.



Problem Solving: Overview (19)

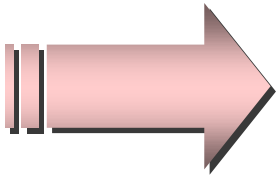
- **What Shape Continues the Pattern?**





Problem Solving: Overview (20)

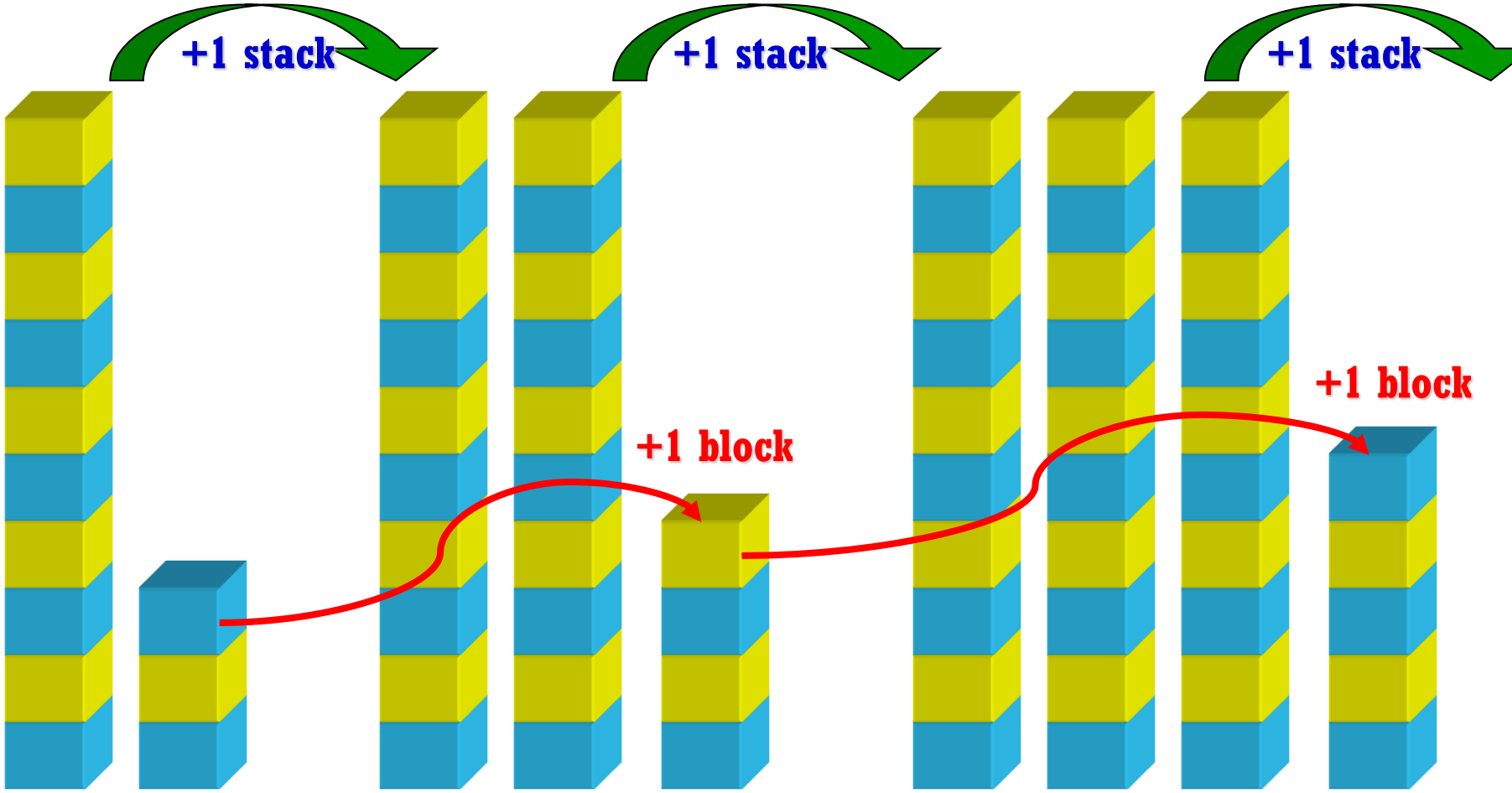
- **What are the Next Two Numbers?**



$5.25, $5.50

Problem Solving: Overview (21)

- What will the Next Set of Blocks Look Like?



Problem Solving: Overview (22)

- To divide a problem helps you to solve it!

Problem Solving: Overview (23)

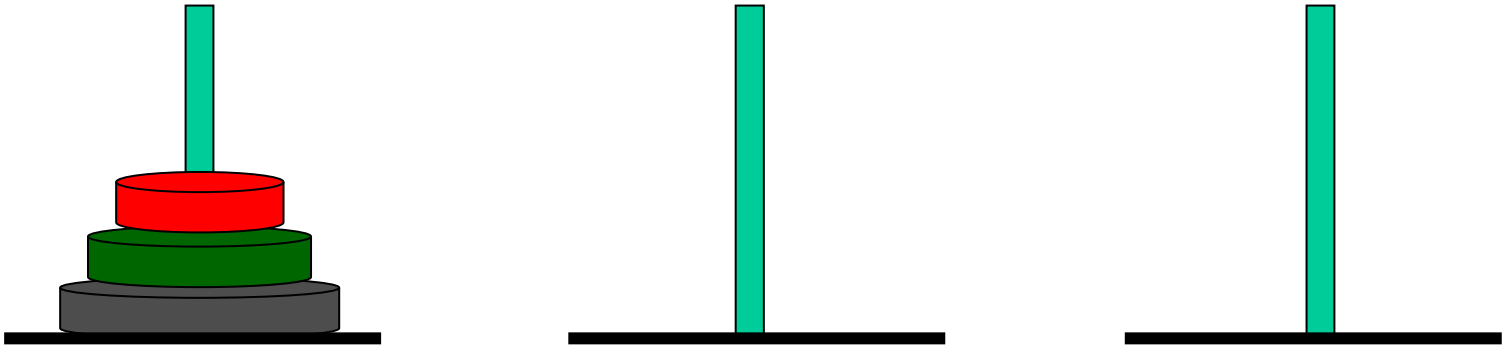
Hanoi Tower Problem

- There are three Towers; A, B, and C.
- N disks are sequentially stacked up in Tow
- We want to move all disks from Tower A to Tower C w.r.t. the following rules
  - Only a single disk can be moved each t
  - Impossible to stack any lager disk onto any smaller disk.
- EX) In case of 3 disks, we need 7 movements as follows:
  - Red→C, Green→B, Red→B, Grey→C, Red→A, Green→C, Red→C

Any Idea for N Disks?

➔ **Dividing the Problem!**

1. **At first, the upper N-1 Disks are moved from A to B, through C!**
2. **Then, the largest Disk in A is moved to C!**
3. **Lastly, N-1 Disks are moved from B to C, through A!**



Problem Solving: Overview (24)

Matrix Multiplication

- Matrices A and B are multiplied; **C = AB**
- What is its (computational) complexity?
- Any efficient matrix multiplication method?

Complexity? $O(n^3)$

$$A=\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \quad B=\begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} \Rightarrow C=\begin{bmatrix} \sum_{k=1}^{n} a_{1k}b_{k1} & \cdots & \sum_{k=1}^{n} a_{1k}b_{kn} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{nk}b_{k1} & \cdots & \sum_{k=1}^{n} a_{nk}b_{kn} \end{bmatrix}$$

$$C=\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix}=\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}\times\begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

Complexity? $O(n^3)$

$$C_1 = A_1B_1 + A_2B_3;\quad C_2 = A_1B_2 + A_2B_4$$
$$C_3 = A_3B_1 + A_4B_3;\quad C_4 = A_3B_2 + A_4B_4$$

$$P_1 = A_1(B_2 - B_4);\ P_2 = (A_1 + A_2)B_4;$$
$$P_3 = (A_3 + A_4)B_1;\ P_4 = A_4(-B_1 + B_3);$$
$$\cdots\cdots\ ;\ P_7 = (-A_1 + A_3)(B_1 + B_2);$$

Complexity? $O(n^{\log 7 = 2.81})$

$$C_1 = -P_2 + P_4 + P_5 + P_6;\ C_2 = P_1 + P_2$$
$$C_3 = P_3 + P_4;\quad C_4 = P_1 - P_3 + P_5 + P_7$$

Problem Solving: Overview (25)

- So far, you saw examples of problem solving techniques.
- You will see more detailed techniques with diverse examples during this course.

Watch these video clips for problem solving

- Problem Solving Technique #1 for Coding Interviews with Google, Amazon, Microsoft, Facebook, etc. (5:52)
  - https://www.youtube.com/watch?v=lD-LuK_VGZI

- 5 Problem Solving Tips for Cracking Coding Interview Questions (19:12)
  - https://www.youtube.com/watch?v=GBuHSRDGZBY

Problem Solving Techniques 문제해결

Jinkyu Lee

Dept. of Computer Science and Engineering,
Sungkyunkwan University (SKKU)

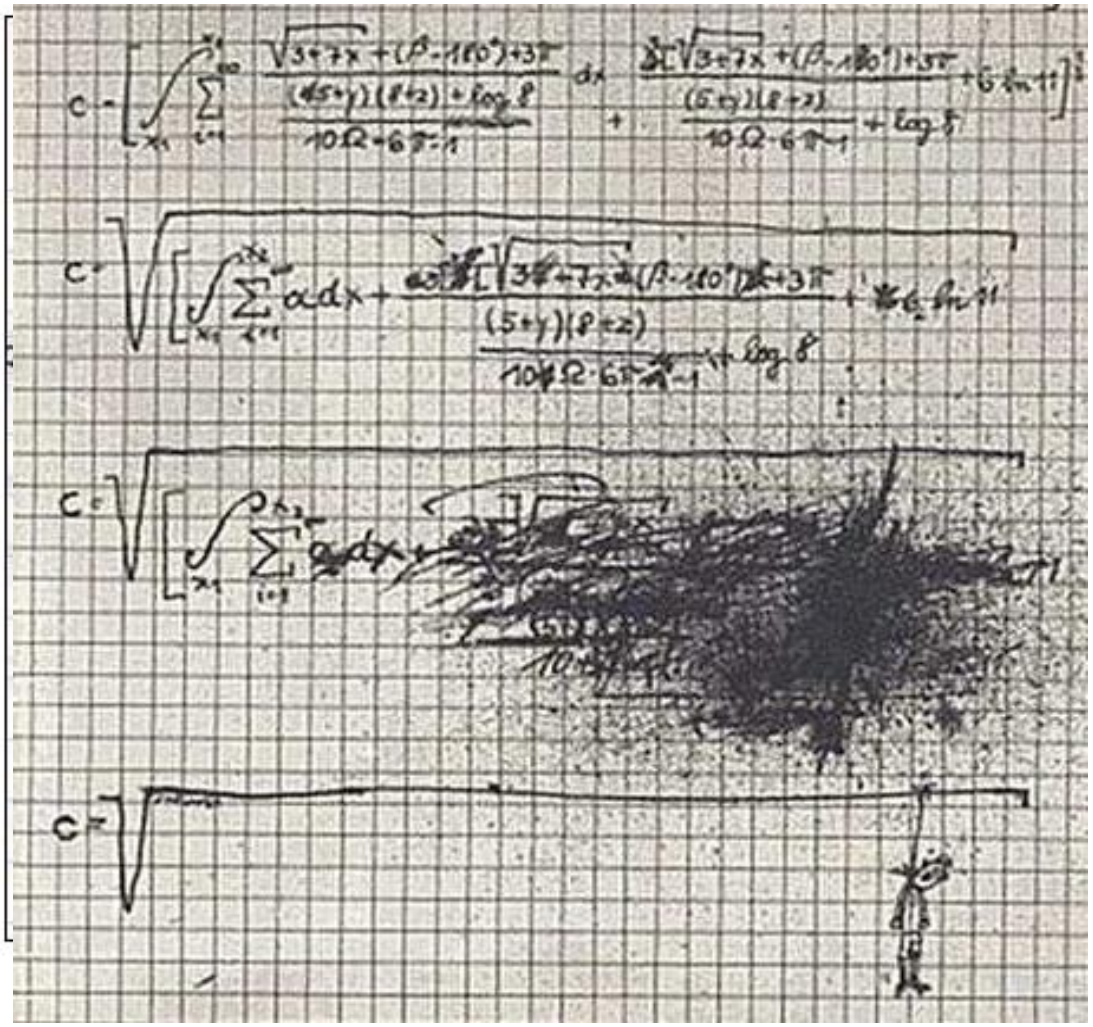
Contents

- Problem solving overview
- **Strategies for problem solving**

Some slides are adapted from Prof. Chang Wook Ahn's slides.

Introduction (1)

- **Examples of "Student Problem Solving" without guidance**



Introduction (2)

- **Why teach Problem Solving Strategies?**

  ☞ Help students deal with problems **creatively** and **effectively**

  ☞ Stimulate students and help develop **thinking skills** in new and unfamiliar situations

  ☞ Develop, reinforce, and enhance **mathematical concepts** and **skills**

  ☞ Help students engage in imaginative and **creative work** arising from **mathematical ideas**

Problem Solving Strategies (1)

- **Many Problem Solving Strategies Exist!**
  - ☞ Generate & Test : Brute-Forth
  - ☞ Hill-Climbing : Heuristic
  - ☞ Subproblems : Divide & Conquer
  - ☞ Working Backwards
  - ☞ Reasoning by Analogy (Pattern)

Problem Solving Strategies (2)

Problem Space

- A useful analogy for problem solving is to think of the problem as a maze
- To solve the problem, need to travel from a starting point (initial state) to an end point (goal state)
  - The initial state: all of knowledge/resources available
  - The goal state: the solution of the problem
- A number of tools, called operators, exist!
  - For the maze, "left turn" and "right turn".

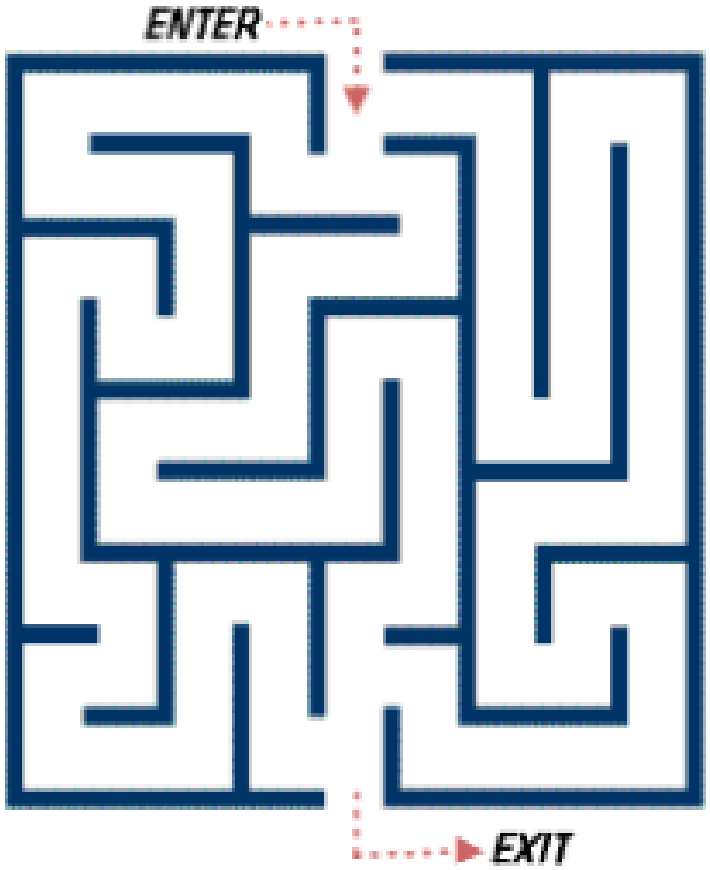
☞ The set of possible ways to travel from the initial state to the goal state is called the problem space

Problem Solving Strategies (3)

Generate & Test (Brute-Forth) Strategy

- The most obvious way: simply test every possible case
  - For the maze problem, all possible operators available at every step are considered
- # operators available is limited by path constraints
  - In a chess game, …
- Moreover, # possible paths is overwhelming!
  - Twenty moves open to you and your opponent
  - In the first cycle, 400 paths should be considered.
  - For two cycles, 640,000 paths; for three cycles, over one billion
- What about go (바둑)?

☞ The generate & test is not a realistic problem solving method!


ENTER
EXIT

Problem Solving Strategies (4)

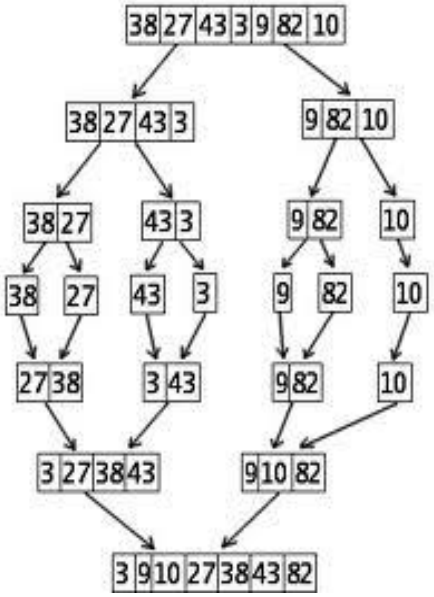
Hill-Climbing (Heuristic) Strategy

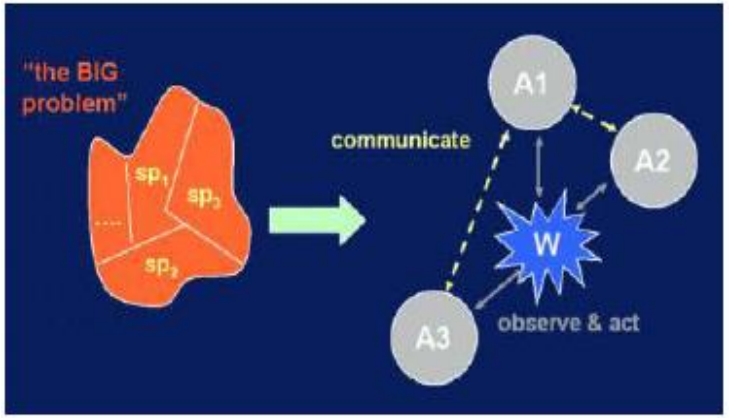
- Consider only a subset of possibilities!
  - Always move toward the goal state.
- Try to solve a maze using this strategy.
  - At every fork, take the direction leading toward the goal state
- This strategy is fallible
  - The correct path often goes away from the goal state
- Advantages
  - Simplicity
  - Effective on finding a local optimum

Problem Solving Strategies (5)

- **Subproblem (Divide & Conquer) Strategy**

  - Break a whole problem down into several subproblem
    - Smaller problems are generally easier to solve because there are fewer possible paths to consider!
  - Very efficient for many real-world applications
    - E.g., Sorting, Searching, etc.





Problem Solving Strategies (6)

Working Backwards Strategy

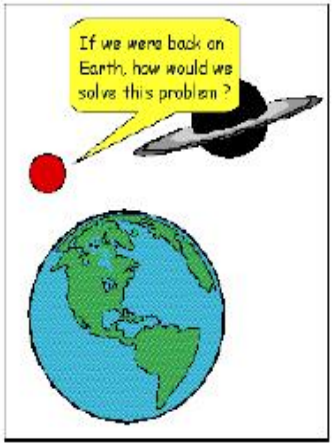
- Some problems are best solved by
  - starting at the goal state and working backward toward the initial state
- Reduce the overwhelming #choices available
  - But not always helpful!

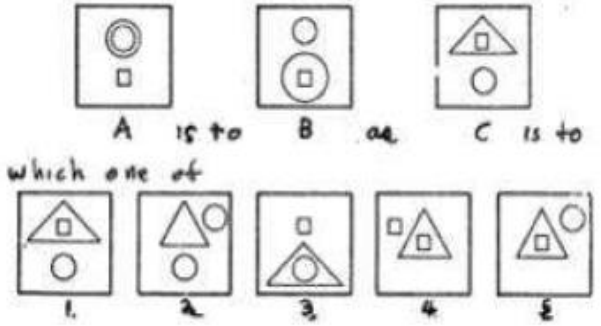
- Example: Water Lilies Problem
  - Water lilies grow rapidly; and thus amount of water surface covered by lilies doubles every 24 hours.
  - On the 1st day of summer, there was one water lily. On the 19th day, the lake was entirely covered.
  - On what day was the lake half covered?

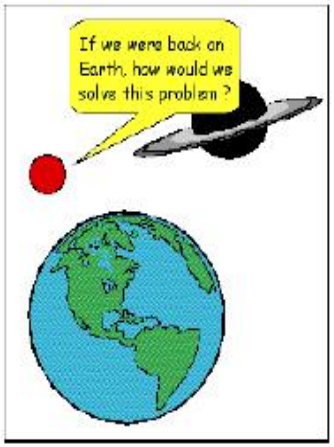
Problem Solving Strategies (7)

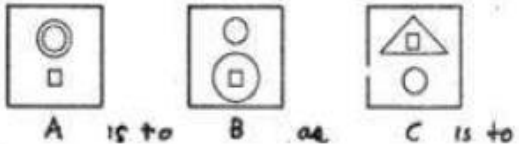
Reasoning by Analogy Strategy

- It works for many different types of problems
  - Use one's knowledge about previous, similar problems
- The use of analogy hinges on familiarity
  - Since analogy is not helpful without previous experience with similar problems.
- Simply, Analogy is Pattern in the problem
  - Best at reasoning by analogy





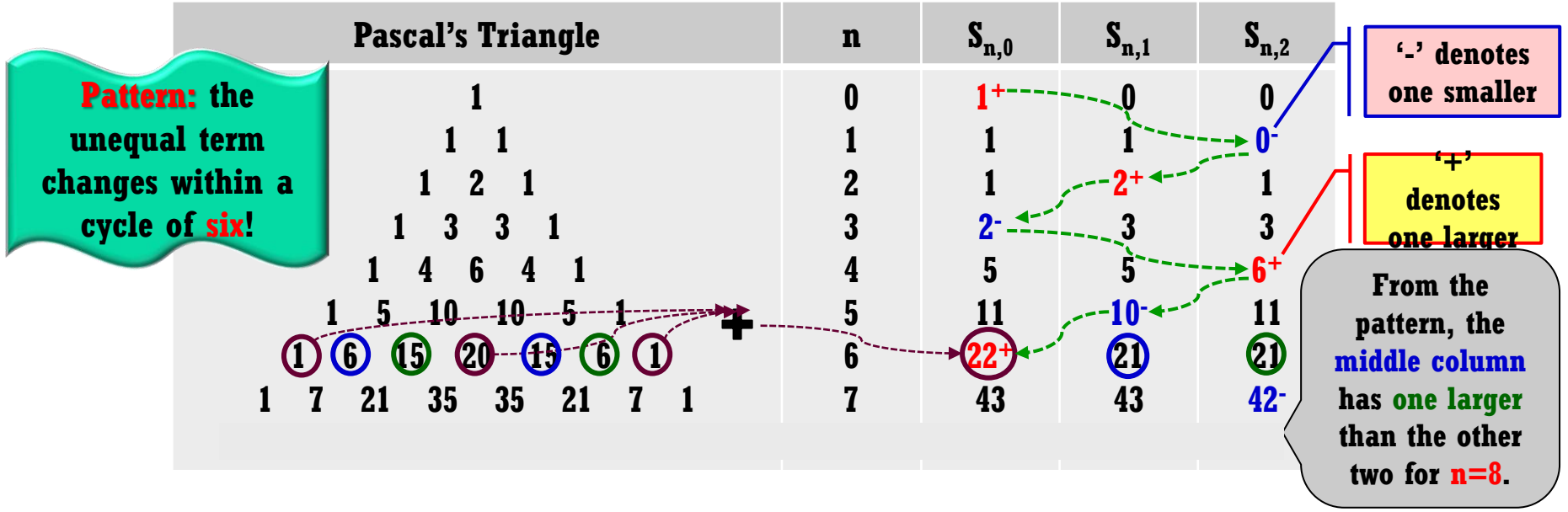

If we were back on Earth, how would we solve this problem ?


A is to B as C is to

Problem Solving Strategies (8)

Searching for a Pattern : Example

- Let $S_{n,0}$, $S_{n,1}$, and $S_{n,2}$ denote the sum of every third element in the nth row of Pascal's Triangle, beginning on the left with the first element, the second element, and the third element respectively.
- Make a conjecture concerning the value of $S_{100,1}$!
  - It begins by examining low-order cases with the hope of finding patterns.



Problem Solving Strategies (9)

- We have $S_{n,0} + S_{n,1} + S_{n,2} = 2^n$
- Since 100 = 6*16 + 4, the unequal term (one larger than the other two) appears in the third column $S_{100,2}$; Thus, $S_{100,0} = S_{100,1} = S_{100,2} - 1$
- Since $S_{100,1} + S_{100,1} + S_{100,1} + 1 = 2^{100}$, $S_{100,1} = (2^{100} - 1)/3$

Pattern: the unequal term changes within a cycle of six!

Pascal's Triangle	n	$S_{n,0}$	$S_{n,1}$	$S_{n,2}$	$\sum S_{n,i}$
1	0	1^+	0	0	1
1 1	1	1	1	0^-	2
1 2 1	2	1	2^+	1	4
1 3 3 1	3	2^-	3	3	8
1 4 6 4 1	4	5	5	6^+	16
1 5 10 10 5 1	5	11	10^-	11	32
1 6 15 20 15 6 1	6	22^+	21	21	64
1 7 21 35 35 21 7 1	7	43	43	42^-	...
1 8 28 56 70 56 28 8 1	8	85	86^+	85	...
.....	.....	.....	.....	.....	2^n
1 100 100	100	$S_{100,0}$	$S_{100,1}$	$S_{100,2}$	
1					

Same pattern

The third column has one larger than the other two for n=100.