GB21802 - Programming Challenges

Week 1 - Basic Problem Solving

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Science

2015-04-13

Last updated April 13, 2015

Some Notes Before the Class

Please check your “Programming Challenge” username!

Some people submitted me invalid usernames:

- “Oda”
- “dai”

Everyone, please don’t forget to send your usernames!

Early submissions

We already had some early submissions – fantastic! Any problems or questions regarding the submission proccess?

Summary for Last Class

How the course works

- Monday Class: Theme exposition;
- Friday Class: Example problem and Q&A;
- Problems: Solve 4 problems every week;

How to submit the problems

- Make an account at `www.programming-challenges.com`
- Send your username to the professor (manaba or e-mail)
- Write the program in C, C++, Java or Pascal

How evaluation works

- Grade = number of problems submitted
- Try to submit one problem per week
- Comments and Participation counts

Summary for This Class

General Problem Solving :
A very important skill which is hard to teach formally;

Data Structures and Programming Challenges :
How to think of data structures outside of the classroom;

Problem Discussion :
Let's introduce last week's problems;

Relax, and ask questions!

No topic here is really new. Listen carefully!

Ask questions any time!

Problem Solving Skills

What are problem solving skills? (1)

I'm sure you are all very good programmers.
You all know many techniques.

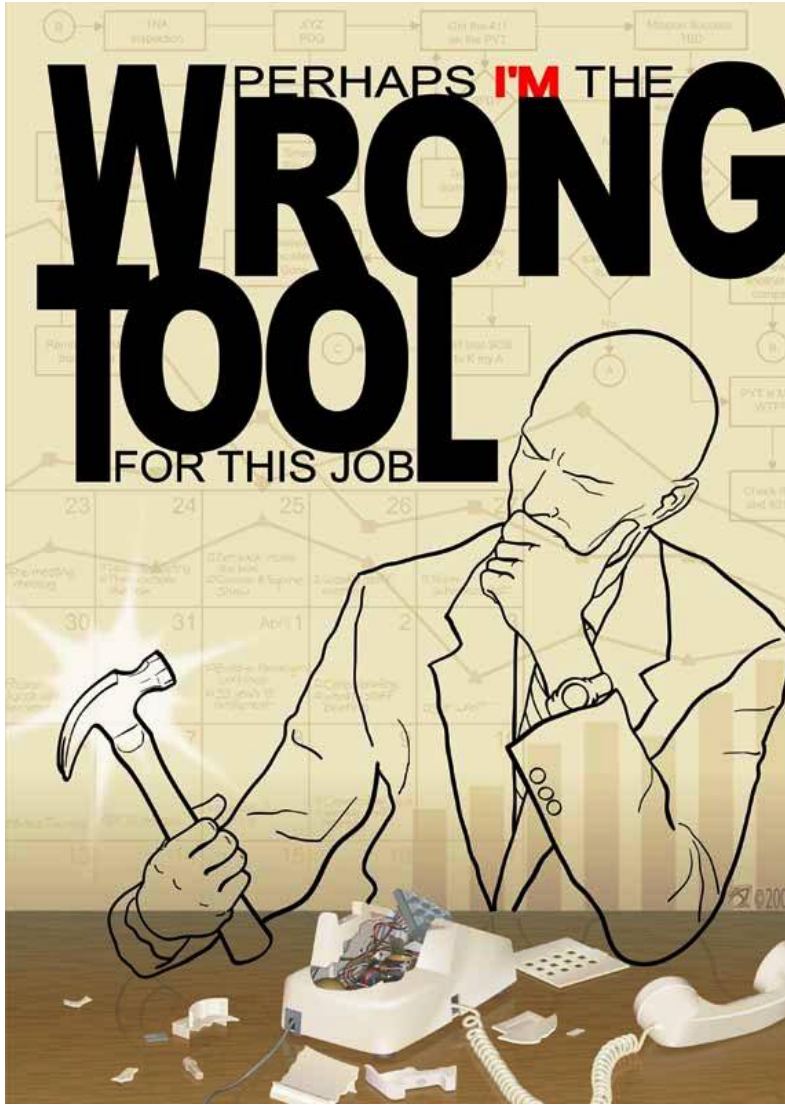
- Data Structures
- Languages and Libraries
- Sorting Algorithms
- Script Tools
- Etc...

But choosing the right technique for a problem is a hard skill.

What are problem solving skills? (2)

Some people don't know what method to solve a problem, unless they are told.

Other people always use their favorite technique, no matter what is the problem.



A Problem can be solved Any way we want. But our brains don't like that:

Difficulty 1: Irrelevant Information

Problems usually have information that is not necessary. If we don't identify this information and erase it, the problem may be impossible.

Difficulty 2: Functional Fixedness

We learn that some tools are for some functions. Sometimes they can be used for other goals (for example, glue and paper for a message)

One of the goals of this course is to help you develop Problem Solving Skills. It will be important for your entire life.

Steps to solve a (Programming) problem

Don't know where to begin? Don't panic, keep calm, and try to follow these steps.

1. Read the input and output
2. Summarize the problem
3. Check for traps
4. Write the program
5. Test/Debug
6. Submit!

Step 1: Input and Output

First read the input and output quickly

Reading the desired input and output gives you a general idea of what is expected. It helps put your mind in the right mood.

- Are you dealing with integers or floats?
- Are strings necessary?
- How big is the input?
- Is the input ordered?

"What does this problem wants from me?"

Step 2: Problem Summary

Read the entire problem and discover the goal

Find out what is the relationship between the Input and the Output.

- Print out the Problem!
- Cross out irrelevant information
- Mark down important information.

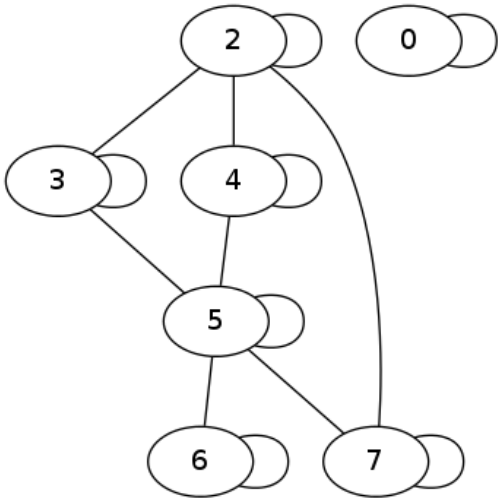
- Write down an algorithm to solve the problem
- Read the problem again
- Repeat until you are confident

Step 3: Problem Traps

Read the problem one more time. There are often “Traps” in the problem description.

- Does the Problem require rounding off of numbers? (Round Up, Round Down, Round to nearest)
- Can the input order be reversed? (Big first, Small First, out of order)
- What is the maximum size of the input? 100? 10000? 1000000000?
- Can the input be negative? Can the input be zero?

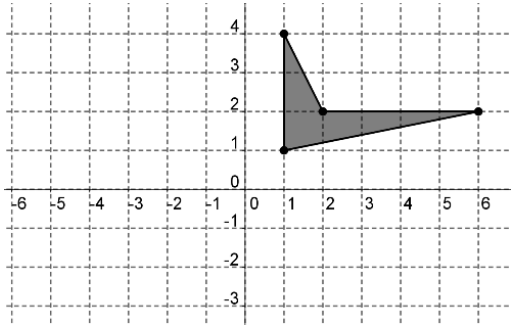
Problem Trap Example 1



Graph Problems:

- Does the graph have self-cycles? (search can go in infinite loops)
- Is the graph connected? (some nodes are unreachable)
- Are all weights positive? (Djikstra does not work)

Problem Trap Example 2



Geometry Problems:

- Can the polygon be convex? (harder algorithms)
- Can two points be in the same place? (division by zero)
- Can points have negative coordinates? (multiplication issues)

Step 4: Write the program

You should do steps 1, 2 and 3 on paper

Only start writing the program after you understand your solution. (easier to avoid bugs!)

Step 4: Write the Program (some hints)

Do Input/output first

- You can't test without IO
- Check the end condition: EOF or Special Case?

Test Often

- Test your program on the test data
- Re-test your program every time you do a small change

Step 4: Algorithm Efficiency, Programmer Efficiency

Algorithm Efficiency

- Pay attention to the size of the input!
- Always calculate the complexity of your algorithm, $O(n), O(n^2), O(n^3)$, etc...
- Multiply the complexity by the input size.

Programmer Efficiency

- Avoid overly complex code!
- Double linked lists are very efficient...
- ... but they are not necessary if your input size is 100.

Step 4: Hints for C/C++

- Love the stl;
- Avoid messing with pointers;

Step 4: Hints for Java

- Object Oriented programming is great, but not very useful for this lecture;
- Abuse java.util:
- All classes must be in the same file;

Step 4: Hints for Pascal

Problem Solving Example: “The Trip”

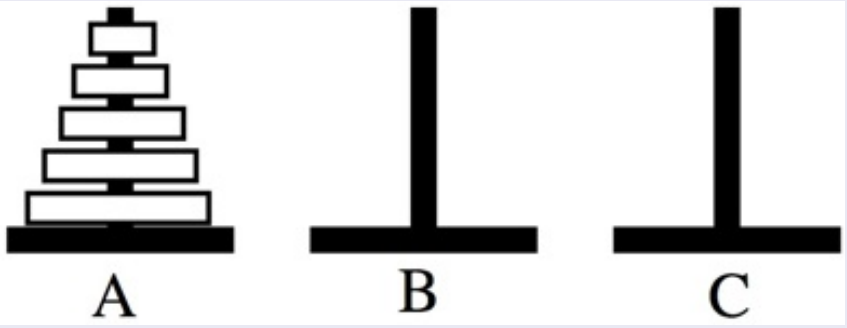
Data Structures

Data Structures

Data structures are the heart of a program

- Using the correct data type can make a problem much easier;
- Using the incorrect data type can make a problem much harder;

The towers of Hanoi



QUIZ: How do you represent the data in this problem?

An easy way to visualize the Towers of Hanoi

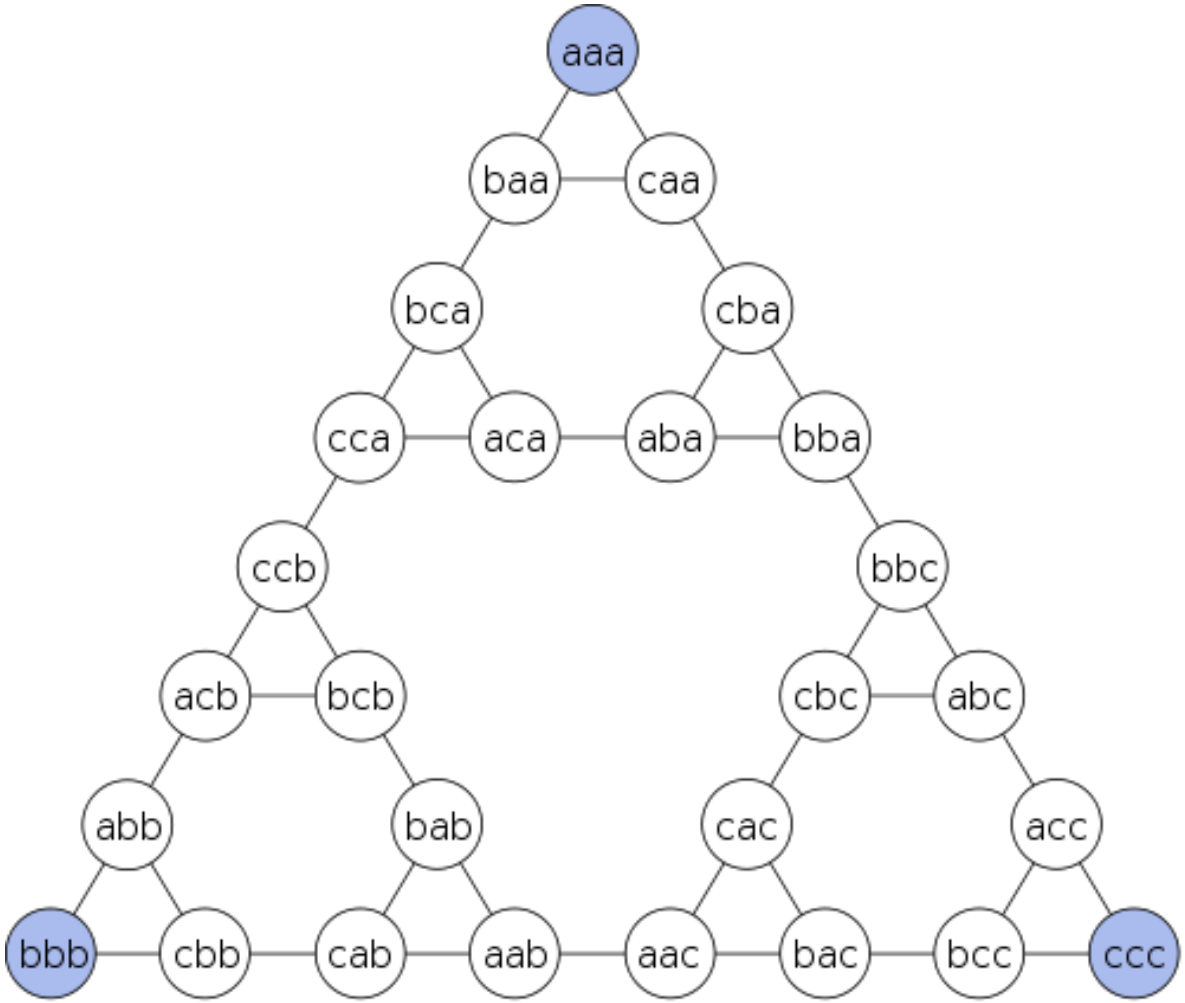
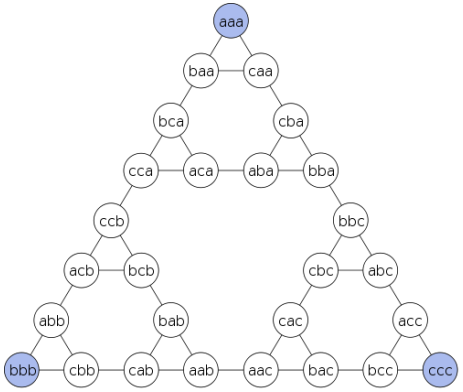


Image created by nonenmac

Explaining the Tower of Hanoi Data Structure

- Each node identifies an state in the problem;
- Each character in the string represents one disk and its position;
- We can have at most 3 state transitions at each state (can you prove it?)
- To solve the Towers of Hanoi problem, we find the path between the start and end states.
- (just beware of state explosion)



Know your data structures!

To be able to do data structure tricks, we need to be familiar with a variety of data structures.

- What is the time and memory efficiency of each data structure?
- What is programming efficiency of each data structure?
- What are the common uses?
- What are the common bugs?

Data Structure Libraries

The “Library” word can have many meanings:

Language Library

- What function is used to create a dictionary?
- What parameters are passed in this function?

Personal Library

- How many data structures do you personally remember?
- Notes on paper can be very useful!

Low Level Data Structures

- Array
- Linked List

The Array is usually simpler, and less bug prone. But be careful with index overflows!

When in doubt, use a bigger array!

Medium Level Data Structures

- Stack
- Queue

The stack is the simplest “complex” data structure. It is implemented with an array and an index.

How do you implement a Queue using two stacks?

Queue and Stack are used very often.

High level data structures

- Sets
- Dictionaries
- Priority Queues

These structures attach extra information to data: Key, Uniqueness, order.

Try to think of them as combinations of the above techniques. How would you implement them?

Other data structures

- Trees
- Graphs

We well talk about these in future classes

String Representation in Computers

When you tyle "*ls*", why do numbers appear before letters?

Strings in a computer system are represented as an Array of characters, and each character is represented as an index.

Characters as numerical indexes: what are the consequences?

- Operation on characters (Addition, comparison);
- Arbitrary order of glyphs;

Encodings

Encodings are mappings between a set of indexes and a set of glyphs. Different encodings cause Mojibake.

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

ASCII Indexes were selected with very specific properties.

This week's problems

List of Problems

- The 3n+1 Problem
- Check the Check
- Erdos Numbers
- Contest Scoreboard

Let's give a quick look on each problem

For the rest of the week:

- Next class: Bring your solutions and questions!
- Submission deadline is 04-19 23:59:59 (Sunday)
- Have a nice week!

Welcome to Friday Class!

Current Solving Stats

- The 3n+1 Problem – Solved:
- Check the Check – Solved:
- Erdos Numbers – Solved:
- Contest Scoreboard – Solved:

Question and hint time!

- The 3n+1 Problem

Question and hint time!

- Check the Check

Question and hint time!

- Erdos Numbers

Question and hint time!

- Contest Scoreboard

Let's solve some different problems