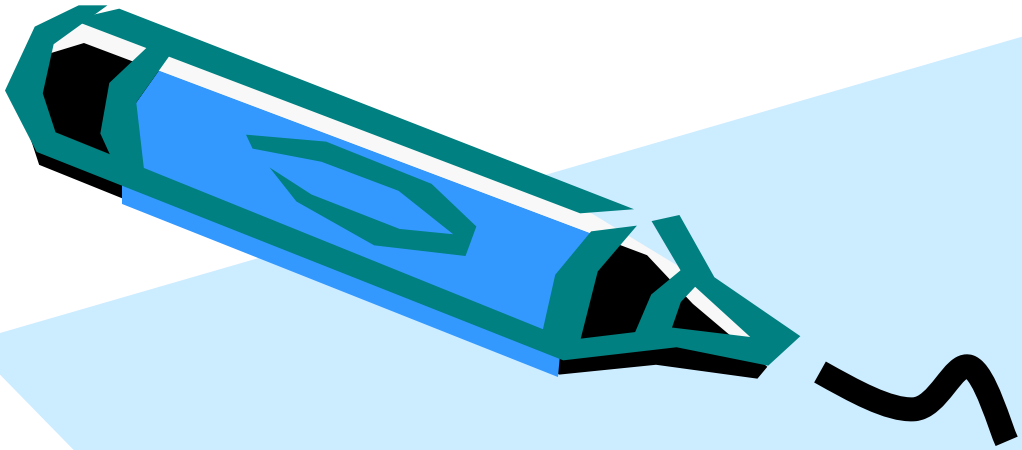
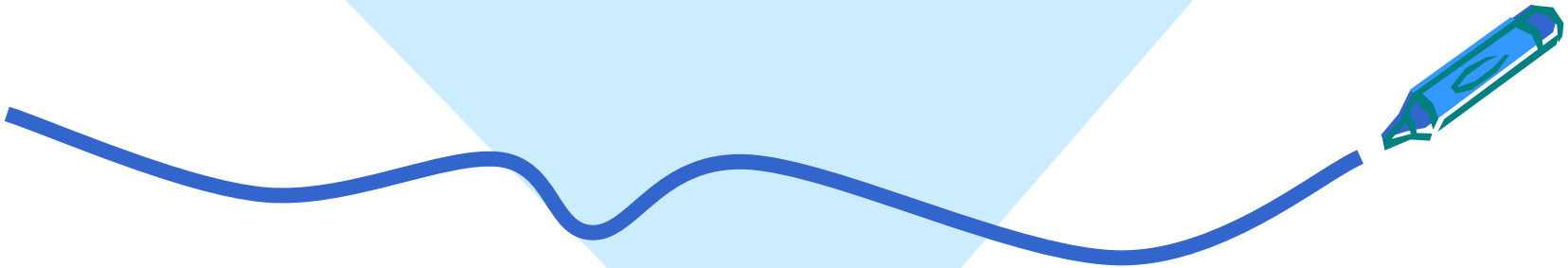
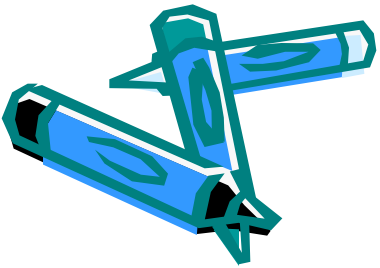


Analisis dan Strategi Algoritma



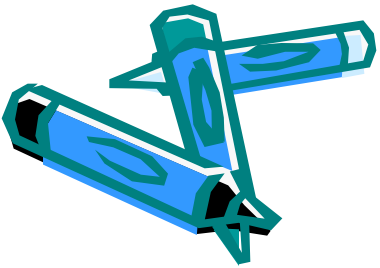
Agenda

- Apa itu Algoritma
- Sekilas Materi Kuliah
- Contoh Penerapan Algoritma
- Tanya Jawab



TigaAn

- 3 Ciri Algoritma
  - Terencana, Dapat dilakukan, Selesai
- 3 AlgoritmaSeminal
  - Al-K, Recipe, DNA
- 3 dalam Paradigma
  - Input, Proses, Output
- 3 jurus dasar:
  - Search, Sort, Merge
- 3 state Algoritma
  - Pre, while, Post
- 3 kondisi Analysis
  - Best, Worst, Average
- 3 macam Solusi
  - Satu, banyak, tanpa

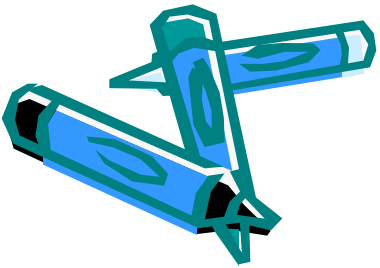


Apakah Strategi, Algoritma dan Analisis Algoritma Itu?

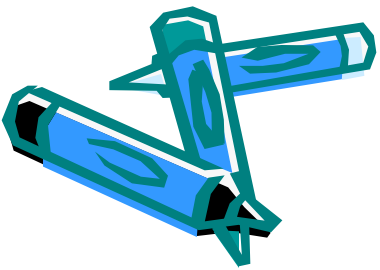
- **Strategi** adalah rencana yang cermat mengenai kegiatan untuk mencapai sasaran khusus
- **Algoritma** :
  - Urutan langkah-langkah untuk memecahkan suatu masalah
  - Terencana, dapat dilakukan, selesai
- **Analisis algoritma** : penentuan kelas algoritma

Referensi Kuliah

- **Text**
  - *Introduction to Algorithms, 2nd edition* T. H. Cormen, C. E. Leiserson, R. L. Rivest, and Clifford Stein Published by: MIT Press or McGraw-Hill
  - *Introduction to the design and analysis of algorithm* Anany Levitin Published by: Addison Wesley
- **Reference:**
  - *Diktat Strategi Algoritmik IF2251,* Ir. Rinaldi Munir, M.T, Departemen Teknik Informatika, Institut Teknologi Bandung

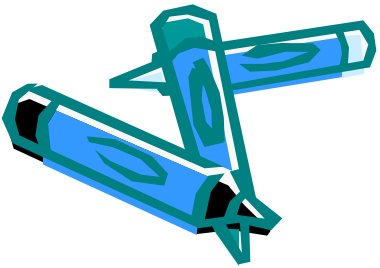


- **Strategi algoritmik** adalah
- kumpulan metode/teknik untuk memecahkan masalah guna mencapai tujuan yang ditentukan,
- yang dalam hal ini deskripsi metode atau teknik tersebut dinyatakan dalam suatu urutan langkah-langkah penyelesaian.



Standar Kompetensi

Setelah mengikuti matakuliah ini diharapkan mahasiswa mampu menganalisis kebenaran suatu algoritma terhadap kasus-kasus tertentu

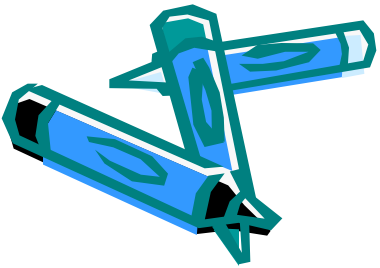


Pokok Bahasan

- **Basic algorithmic analysis**
  - Asymptotic analysis of upper and average complexity bounds
  - Identifying differences among best, average, and worst case behaviors
  - Big "O," little "o," omega, and theta notation
  - Standard complexity classes
  - Time and space tradeoffs in algorithms
  - Using recurrence relations, characteristic equation, and master theorem to analyze recursive algorithms

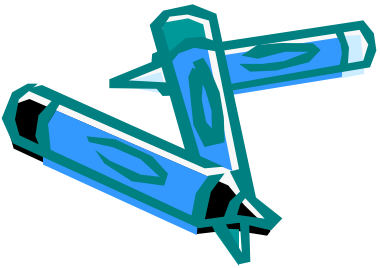
- **Algorithmic strategies**
  1. Brute-force algorithms
  2. Greedy algorithms
  3. Divide-and-conquer
  4. Backtracking
  5. Branch-and-bound
  6. Heuristics
  7. Pattern matching and string/text algorithms
  8. Numerical approximation algorithms
  9. Dynamic Programming

- Fundamental computing algorithms



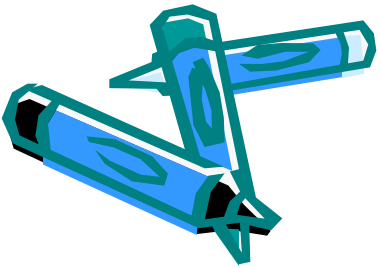
3 Algoritma Seminal

- I: pen and paper
  - Angka Arab, Al-Khawarizmi, Yang mana?
- II: Resep Masakan
  - China, Cap Cay
- III: DNA
  - Kompleks dalam kesederhanaan



I: Pen and Paper

- Bagaimana kita melakukan perkalian bilangan?
  - Hafalan
  - Dengan Bantuan
    - Tangan
    - Kertas
    - Kalkulator
- Manakah yang merupakan Algoritma?



Mengalikan menggunakan Kertas

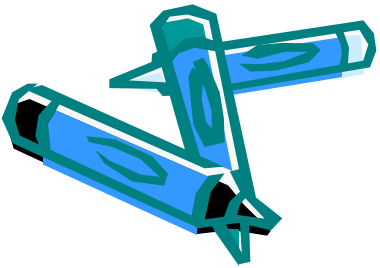
- **Menggunakan Angka Arab**

```
   2345
    678 x
  -------
  18760
 16415
14070   +
-------
1589910
```

- **Angka Romawi**

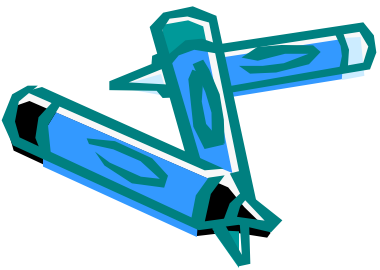
```
  MMCCCVL
DLXXVIII x
```

?



Pencil and Paper

- Penjumlahan
- Pengurangan
- Perkalian
- Pembagian
- Akar Bilangan
- Pangkat Bilangan

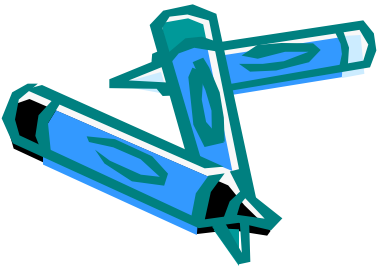


3 Komponen Algoritma

- Paradigma IPO

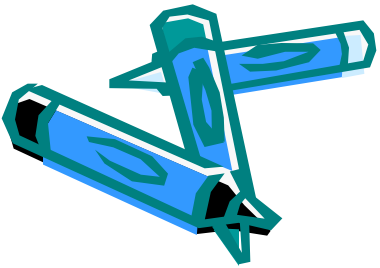
Setiap Algoritma dapat disebutkan:

- Input : masukan
- Proses : me-'*^%$'-kan I → O
- Output : keluaran



Latihan I: Pen and Paper

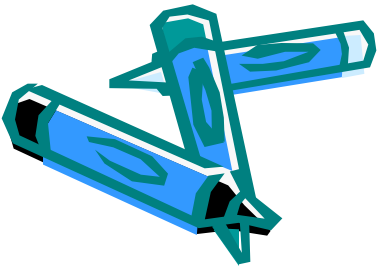
- Pada Algoritma perkalian menggunakan kertas, sebutkan IPOnya.
- Input : Dua buah bilangan
- Proses : *<pre-memory>*
- Output : Bilangan hasil perkalian

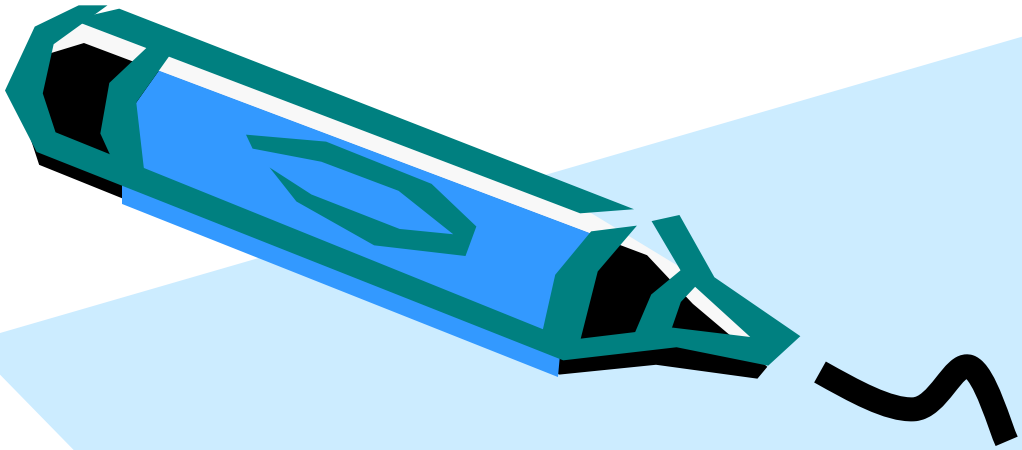


II: Resep Masakan

Resep membuat Cap Cay

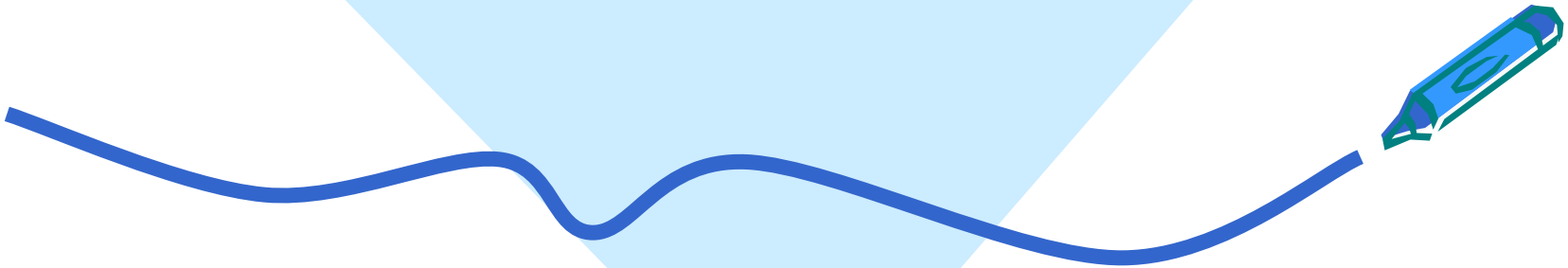
- Input : Bahan Masakan, dan Cara Memrosesnya
- Proses : <interpretasi> <urutan>
- Output : Cap Cay untuk 6 orang

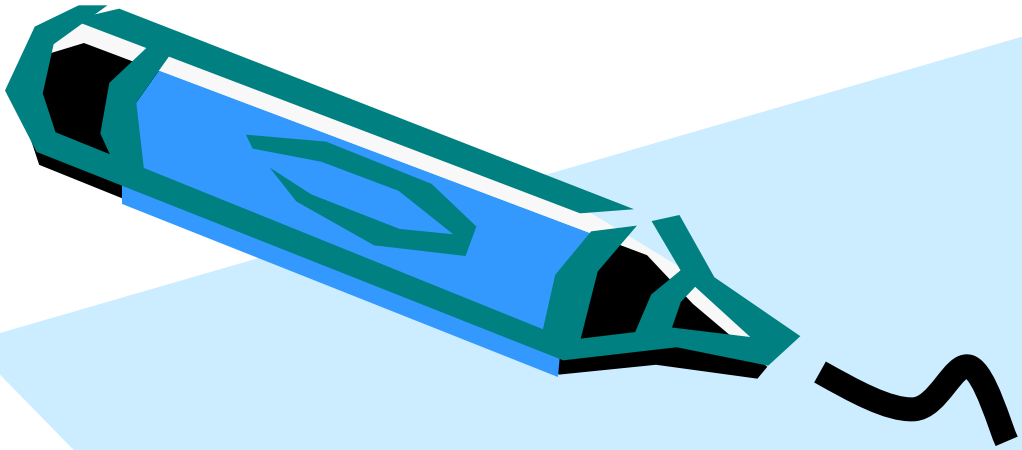




Algorithm in Action

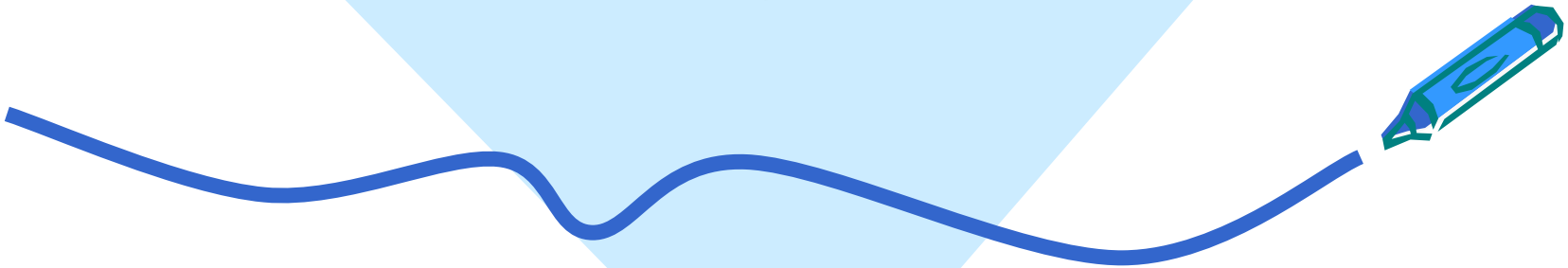
Contoh Kasus





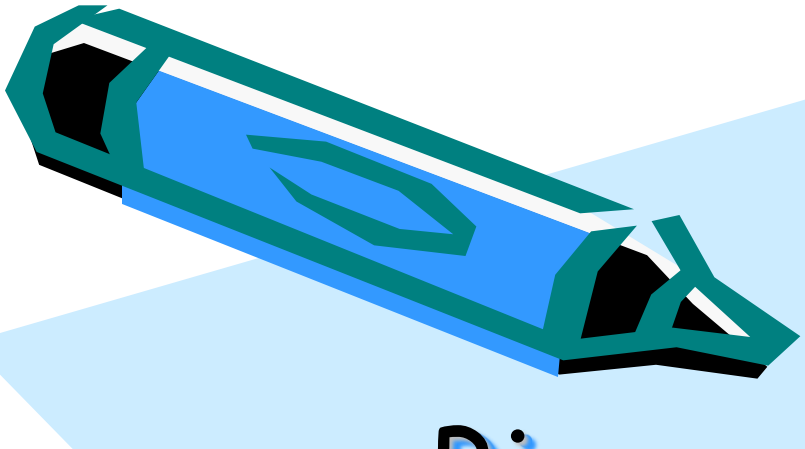
Why we study Theory?

Demo



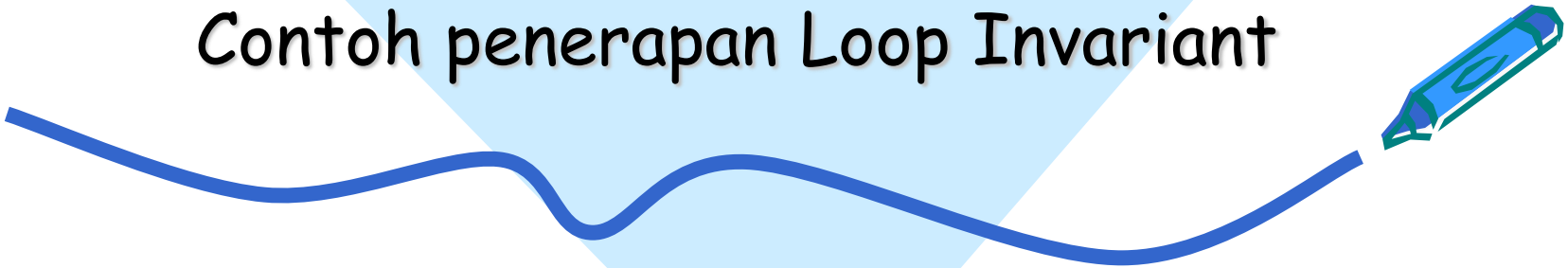
Card Trick

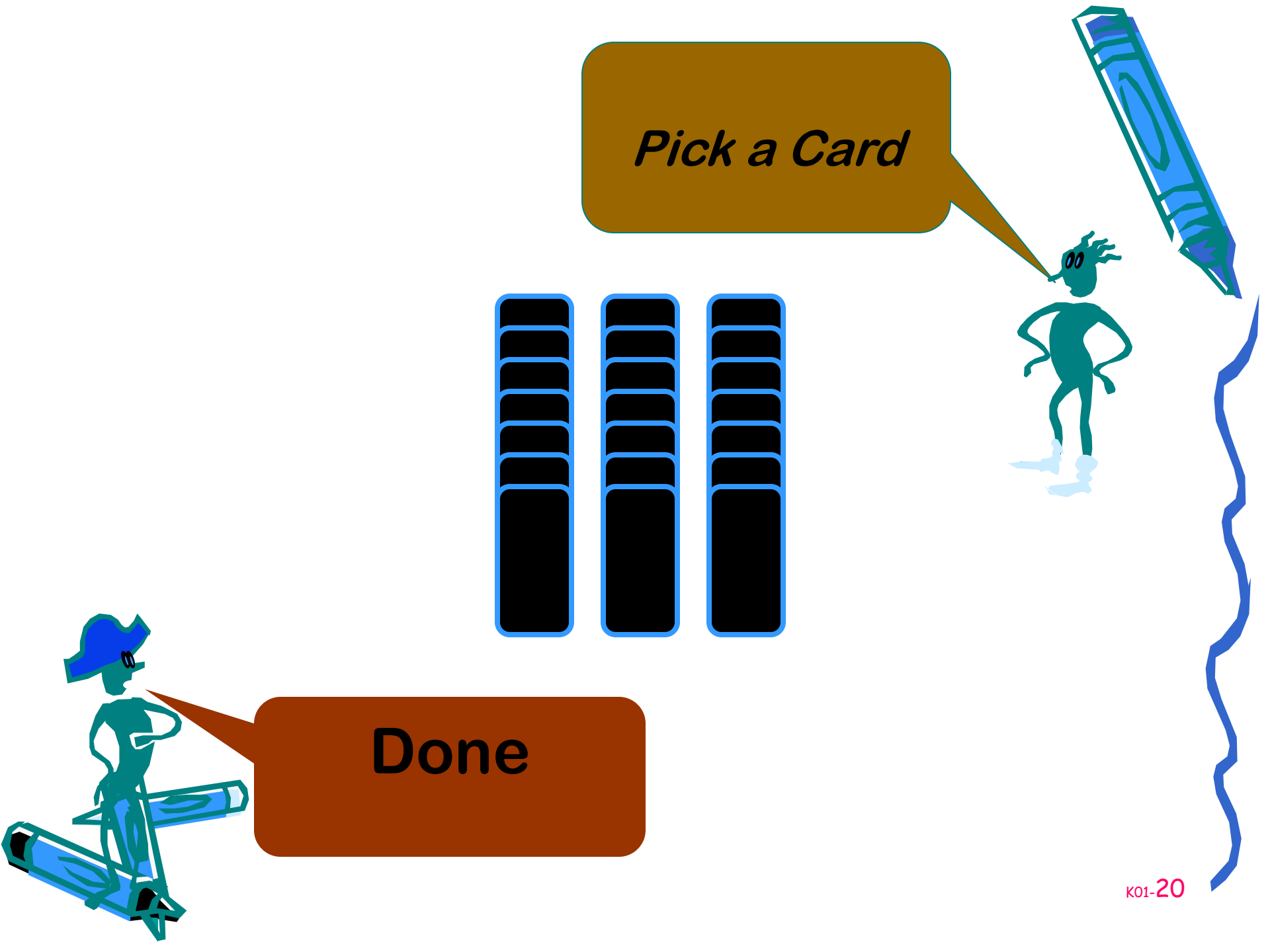
- A volunteer, please.



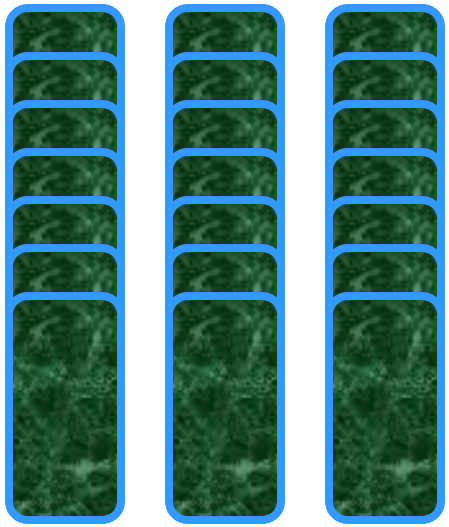
Binary Search like example

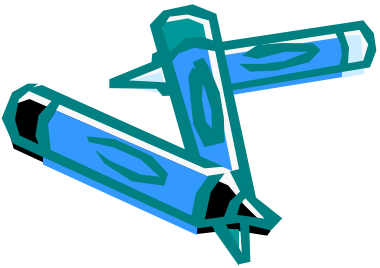
Contoh penerapan Loop Invariant

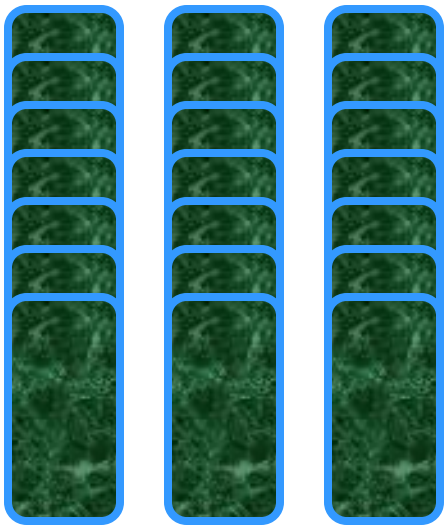



Pick a Card
Done

Loop Invariant:
The selected card is one of these.



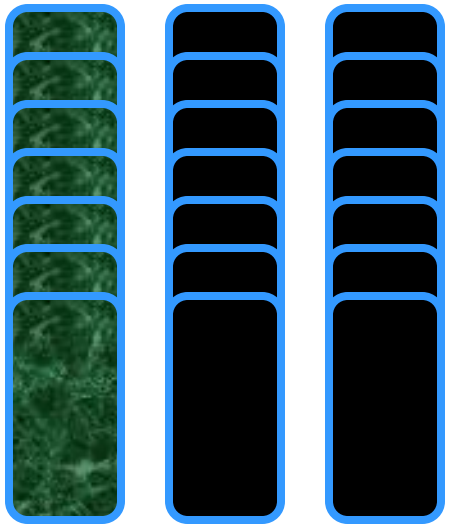


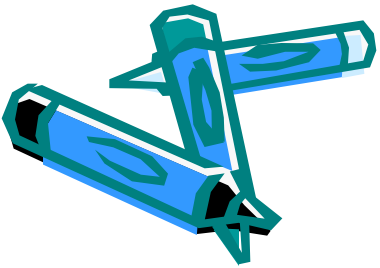


Which column?

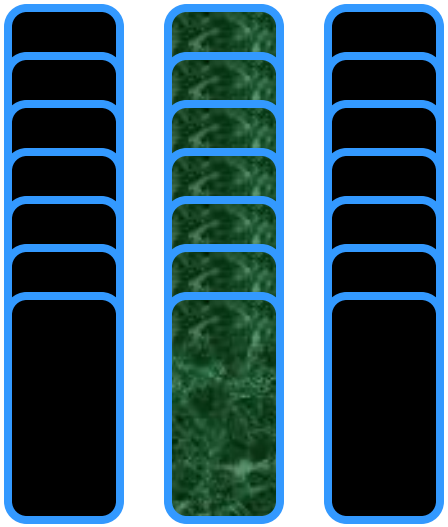
left

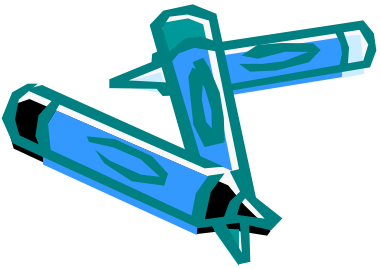
Loop Invariant:
The selected card is one of these.



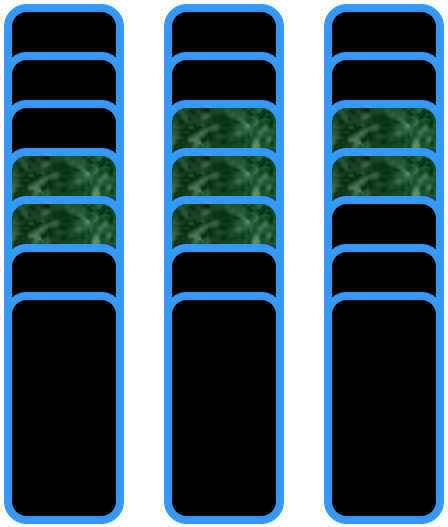


Selected column is placed in the middle

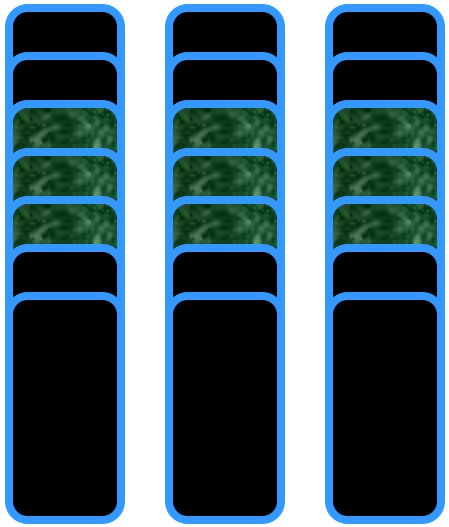


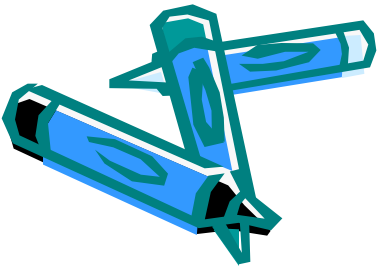


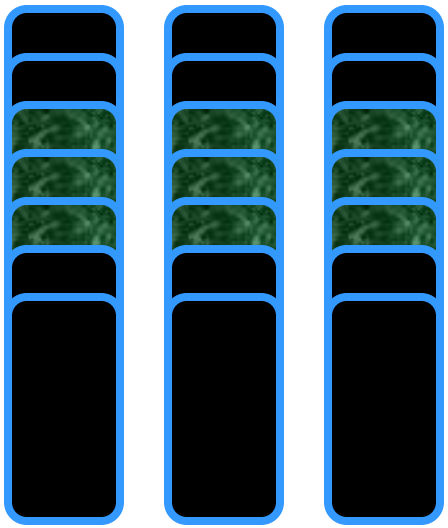
I will rearrange the cards

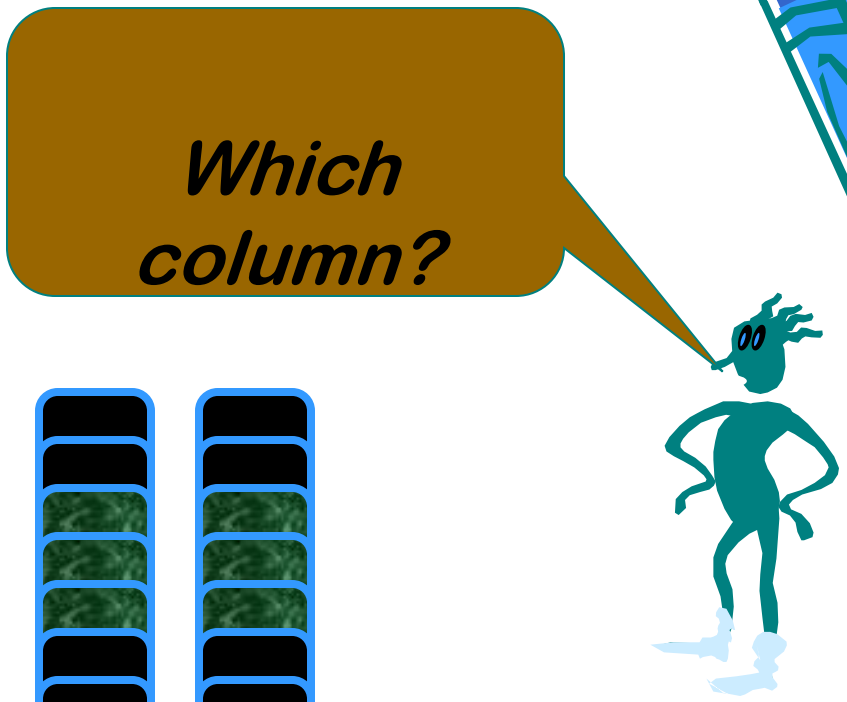


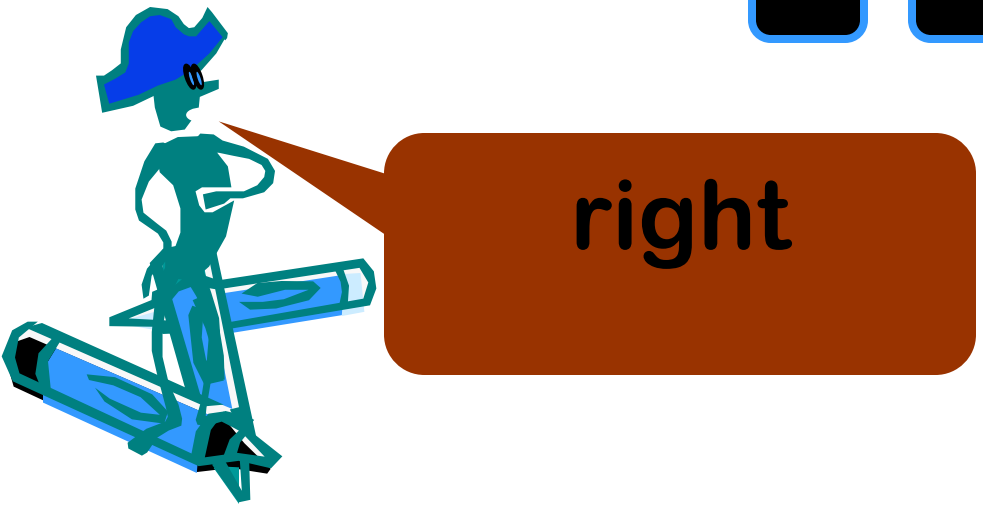
Relax Loop Invariant:
I will remember the same about each column.



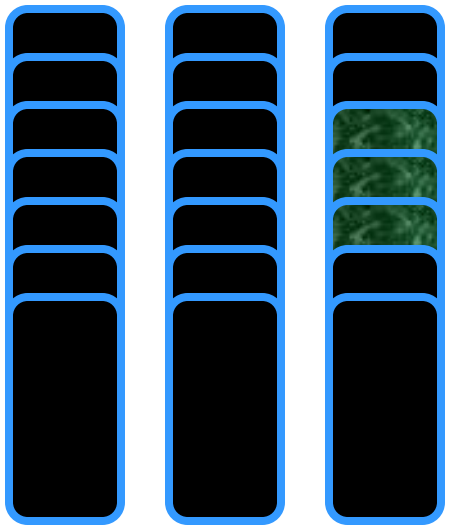


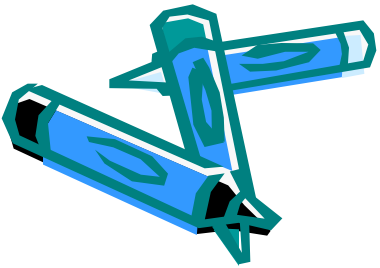



Which column?

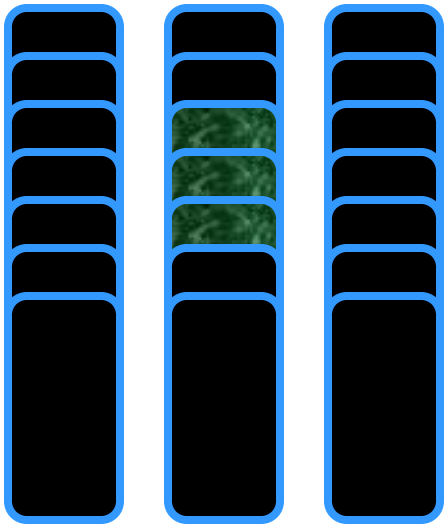

right

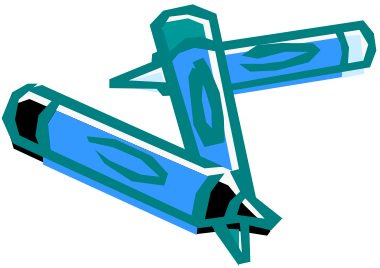
Loop Invariant:
The selected card is one of these.



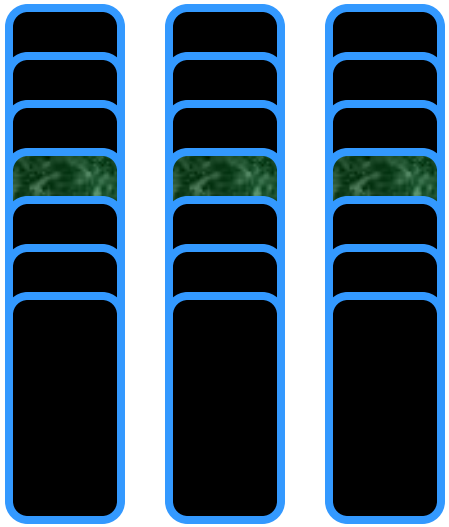


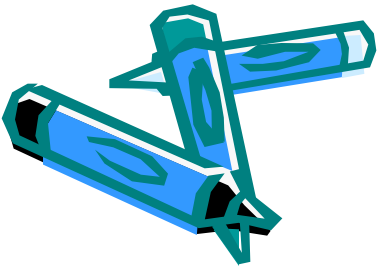
Selected column is placed in the middle

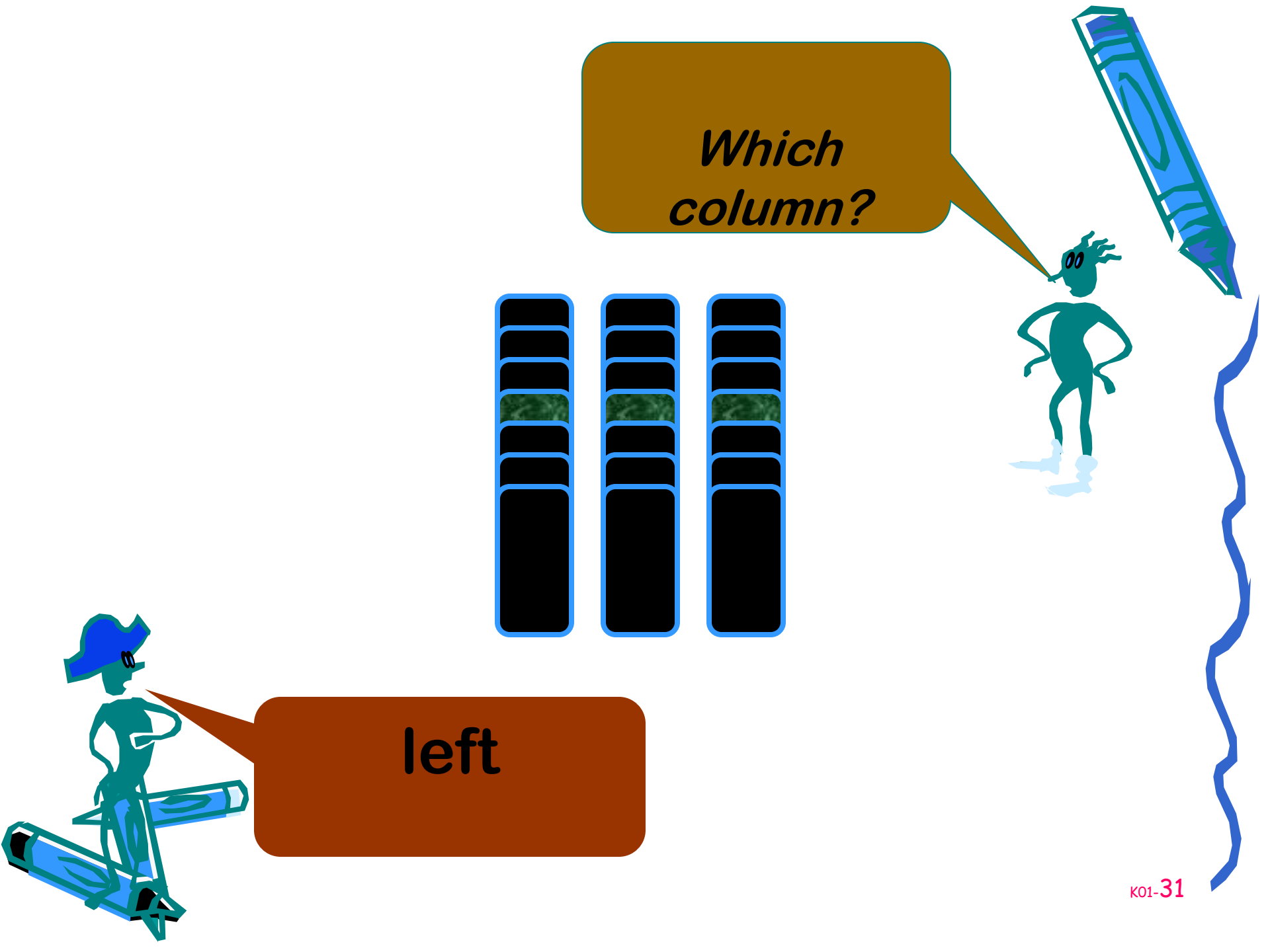




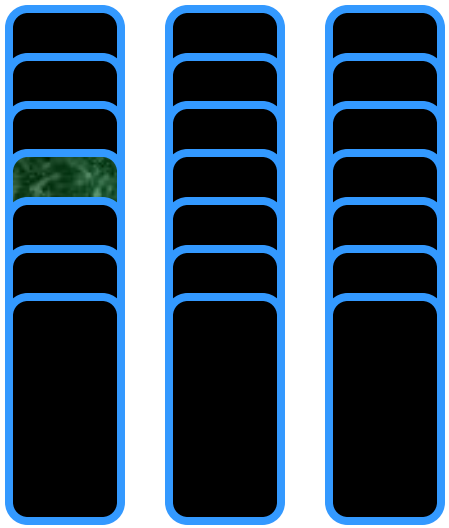
I will rearrange the cards

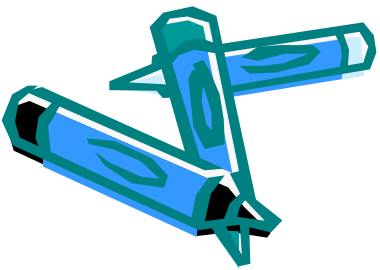




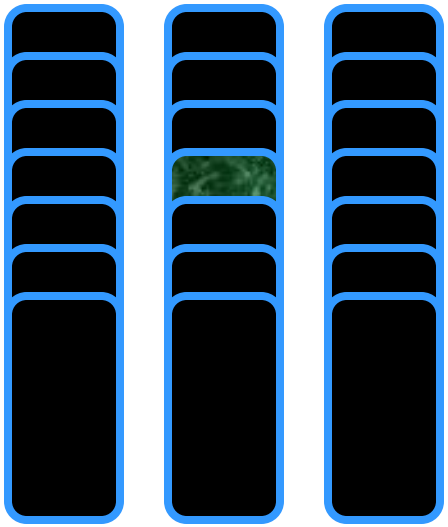

Which column?
left

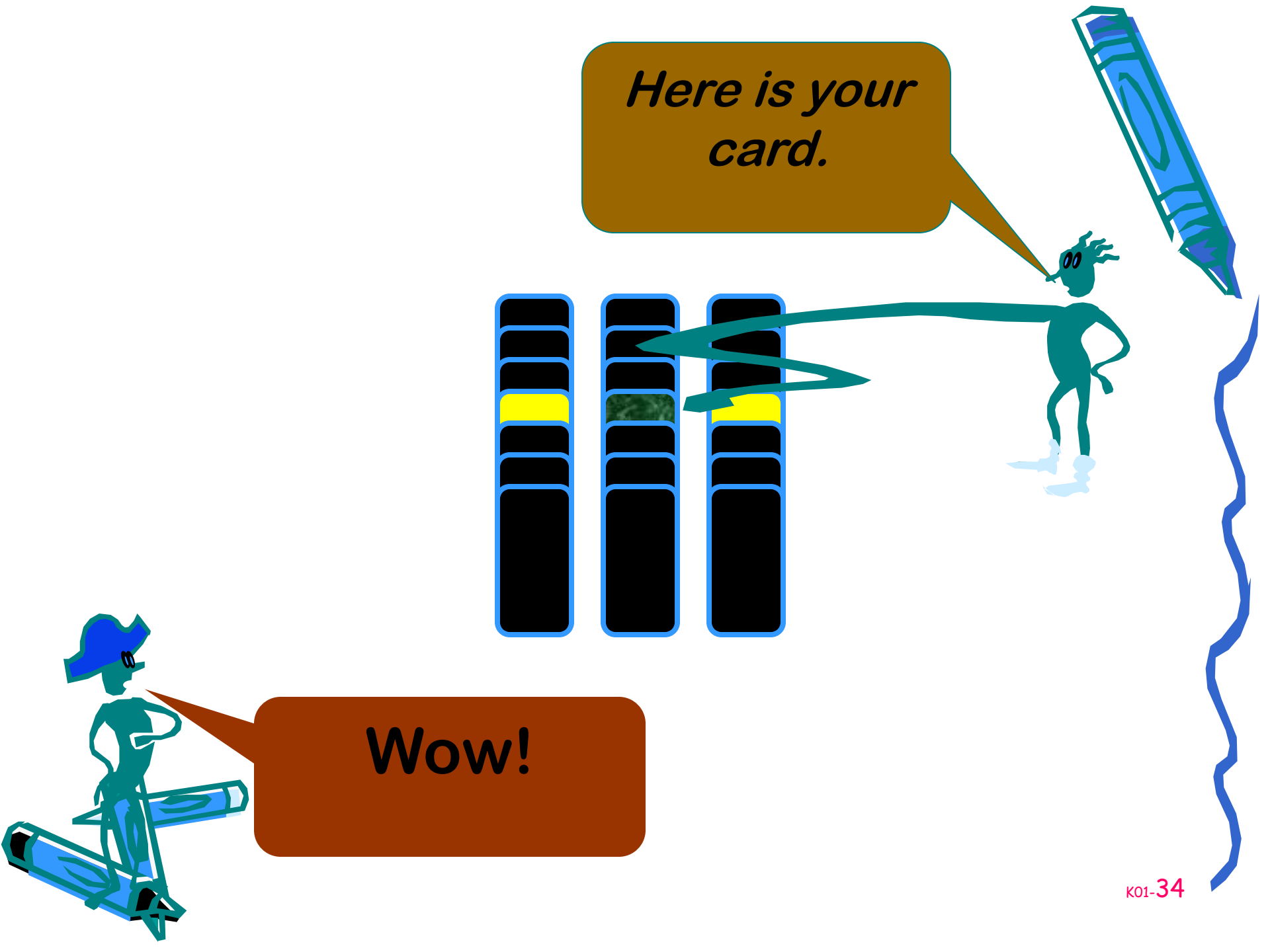
Loop Invariant:
The selected card is one of these.





Selected column is placed in the middle

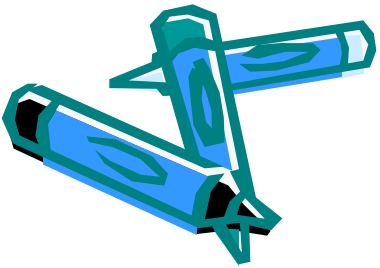



Here is your card.
Wow!

Loop Invariants

Suatu cara yang baik untuk menyusun program:

- Simpan semua yang anda tahu pada basisdata.
- Pada loop utama, update basisdata ini pada saat program dijalankan.



Thinking about Algorithms Abstractly

Introduction

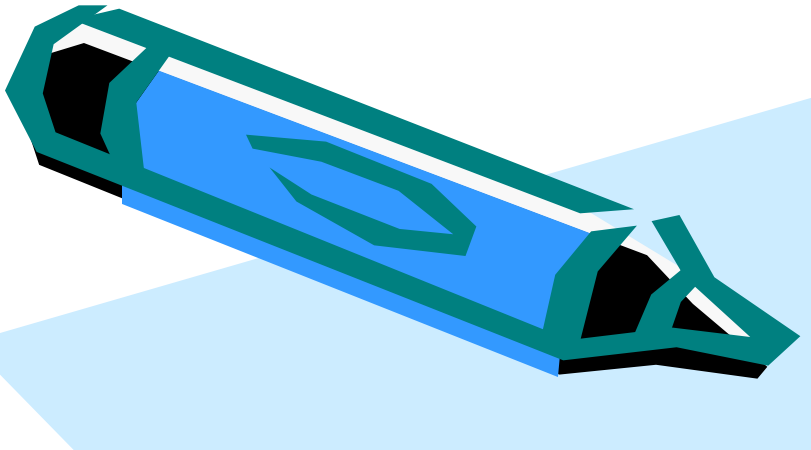
- So you want to be a computer scientist?
- Grade School Revisited: How To Multiply Two Numbers

Jeff Edmonds
York University

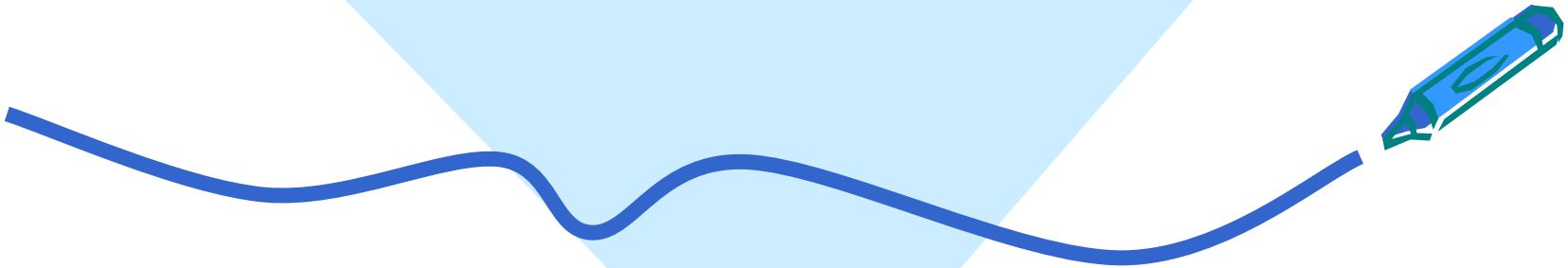
So you want to be a computer scientist?

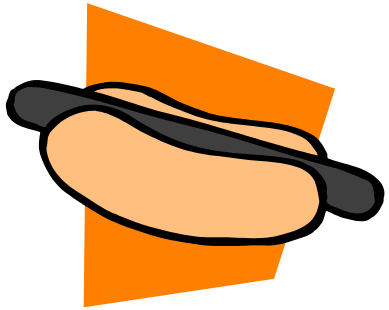
Is your goal to be a mundane programmer?

Or a great leader and thinker?



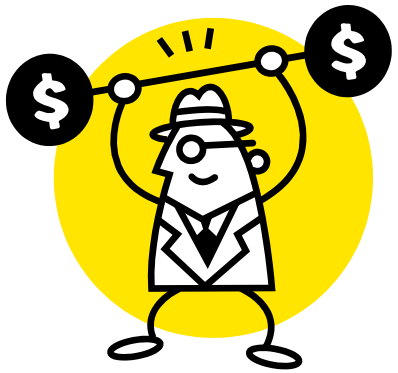
Original Thinking

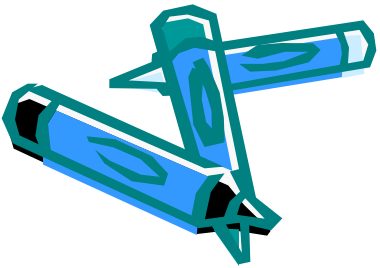




Boss assigns task:

- Given today's prices of sausage, grain, sawdust, …
- Given constraints on what constitutes a hotdog.
- Make the cheapest hotdog.





Everyday industry asks these questions.

Your answer:

- Um? Tell me what to code.

With more suffocated software engineering systems, the demand for mundane programmers will diminish.

Your answer:

- I learned this great algorithm that will work.

Soon all known algorithms will be available in libraries.

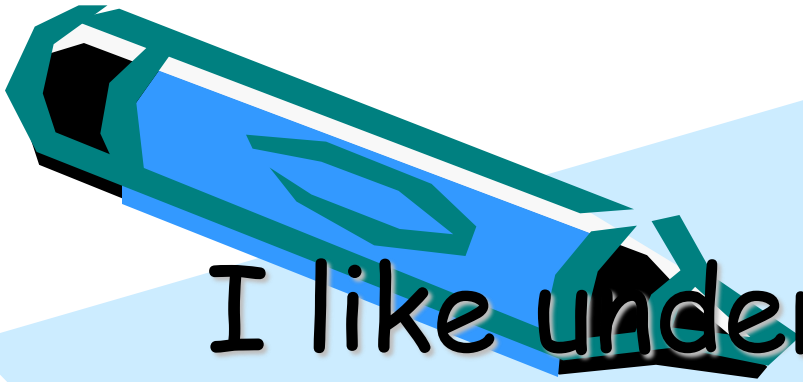
Your answer:

- I can develop a new algorithm for you.

Great thinkers
will always be needed.

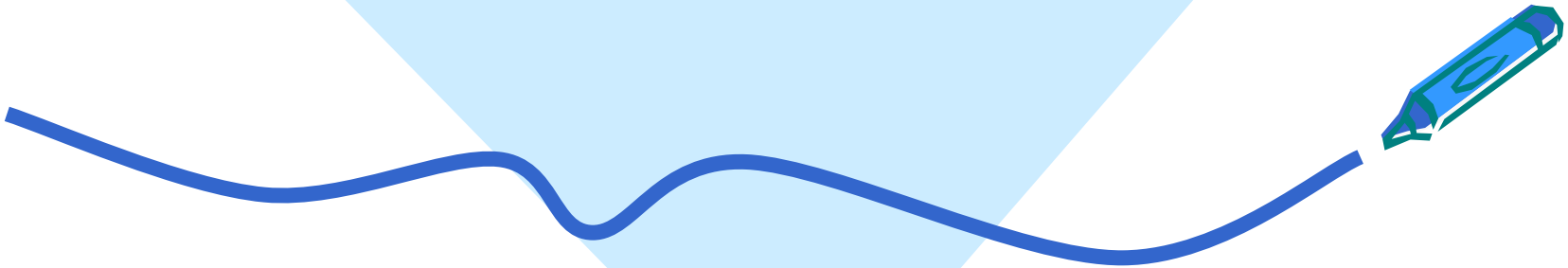
The future belongs to the computer scientist who has

- Content: An up to date grasp of fundamental problems and solutions
- Method: Principles and techniques to solve the vast array of unfamiliar problems that arise in a rapidly changing field



I like understanding things using a story.

I will now tell one.



The Getting to School Problem

Problem Specification

- Pre condition: location of home and school
- Post condition: Traveled from home to school

Algorithm

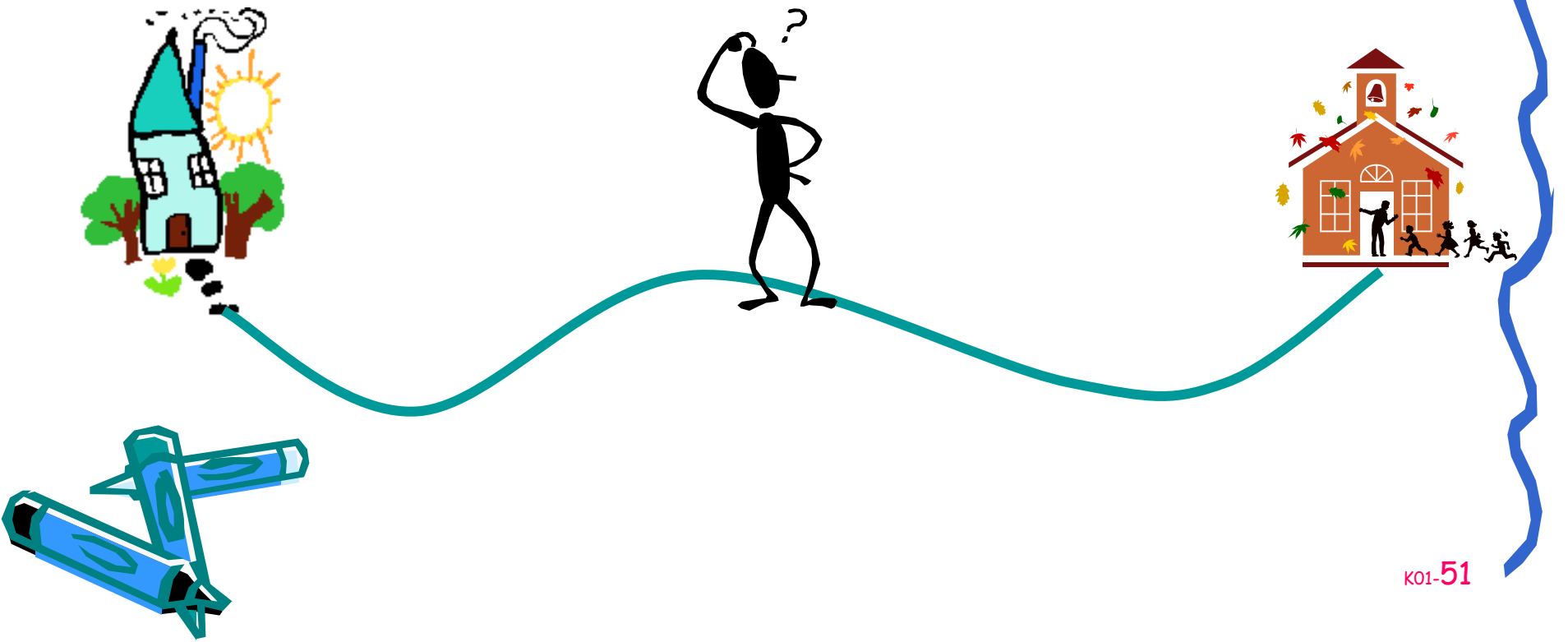
- What is an Algorithm?

Algorithm

- The algorithm defines the computation route.

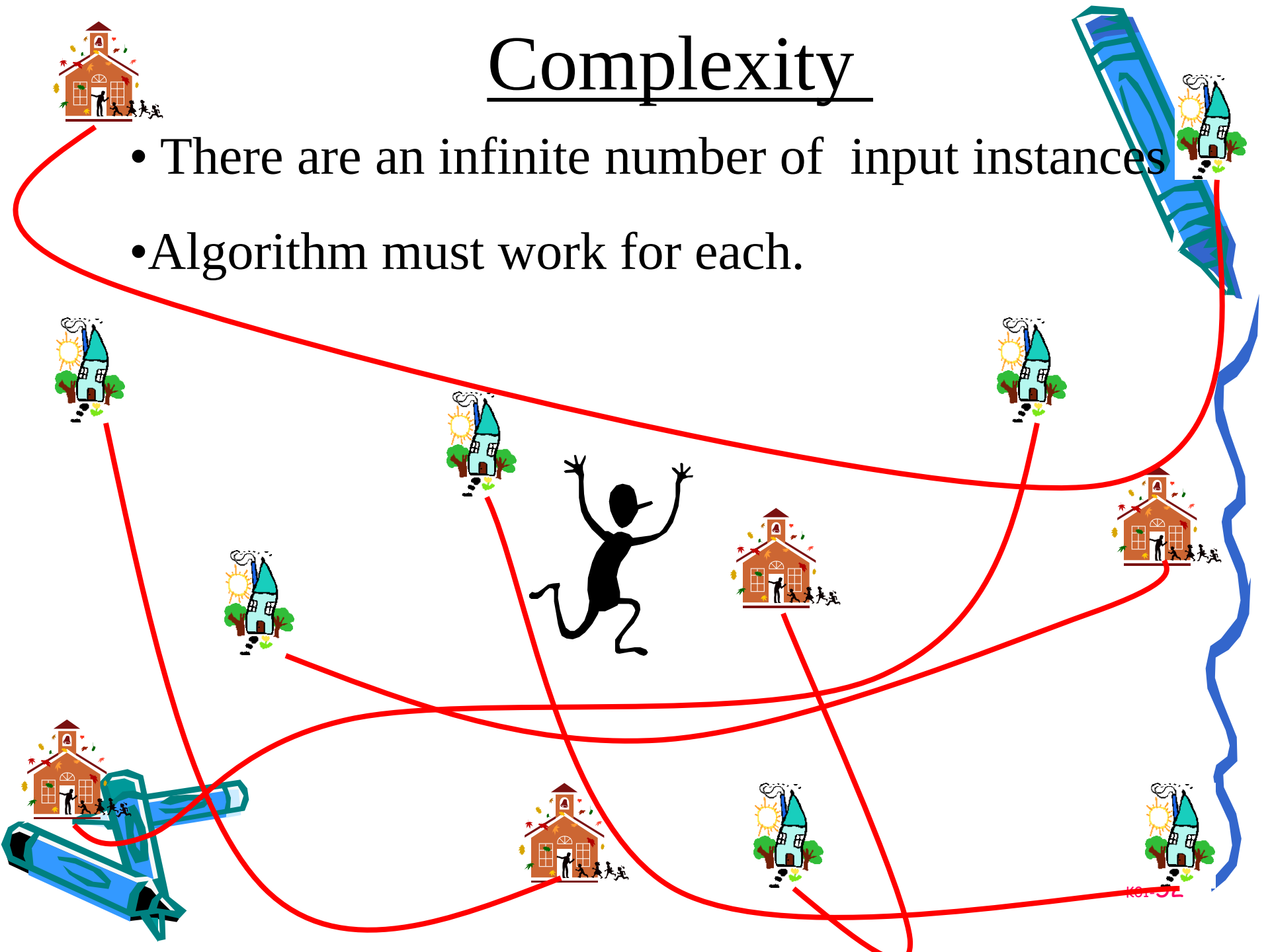
Algorithm

- Is this reasonable?



Complexity

- There are an infinite number of input instances
- Algorithm must work for each.

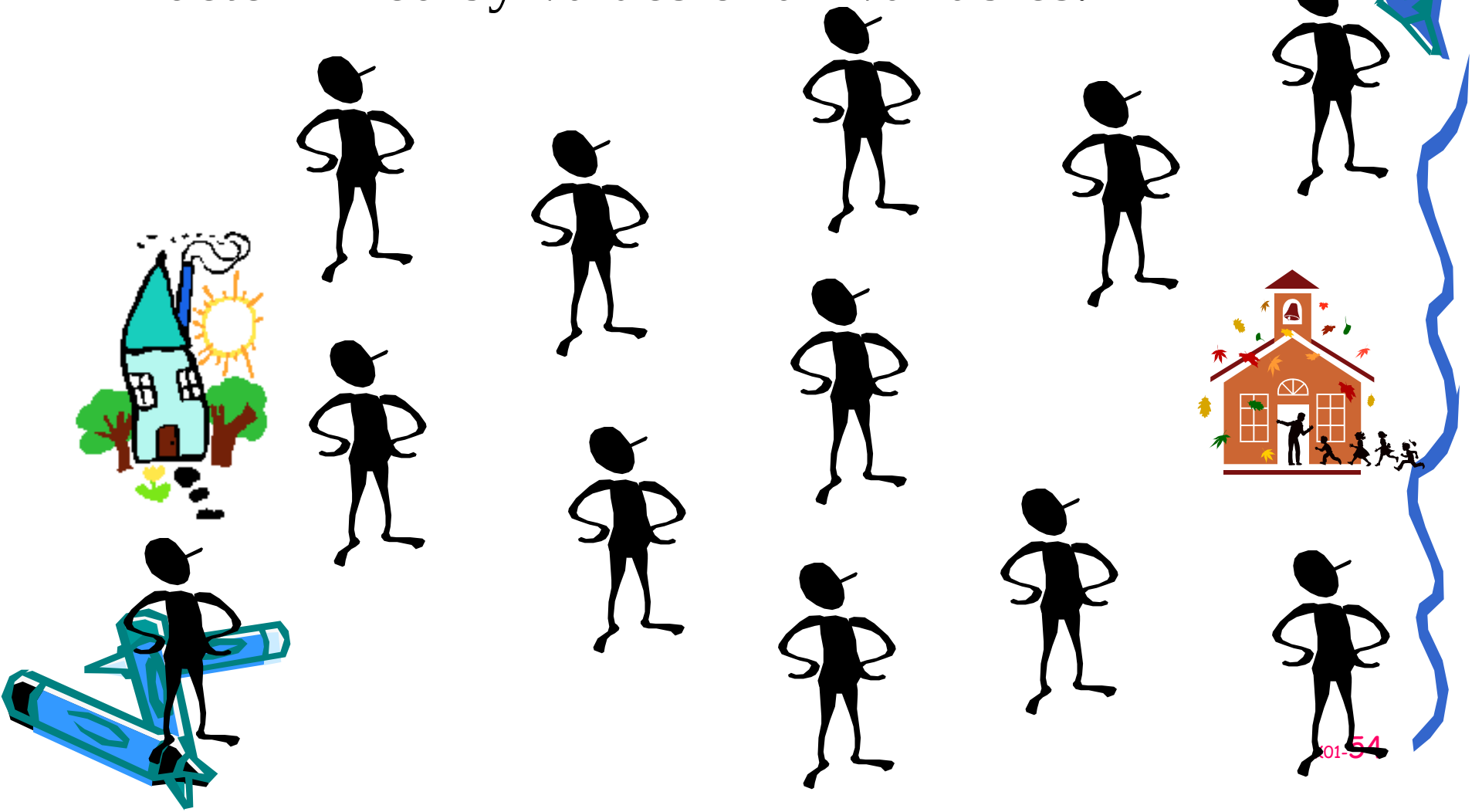


Complexity

• Difficult to predict where computation might be in the middle of the computation.

Location of Computation

- The current "State" of computation is determined by values of all variables.



Location of Computation

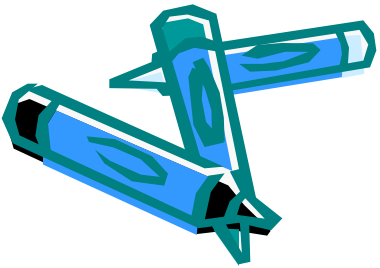
- Suppose the computation ends up here?

Don’t Panic

- Where ever the computation might be, take best step towards school.

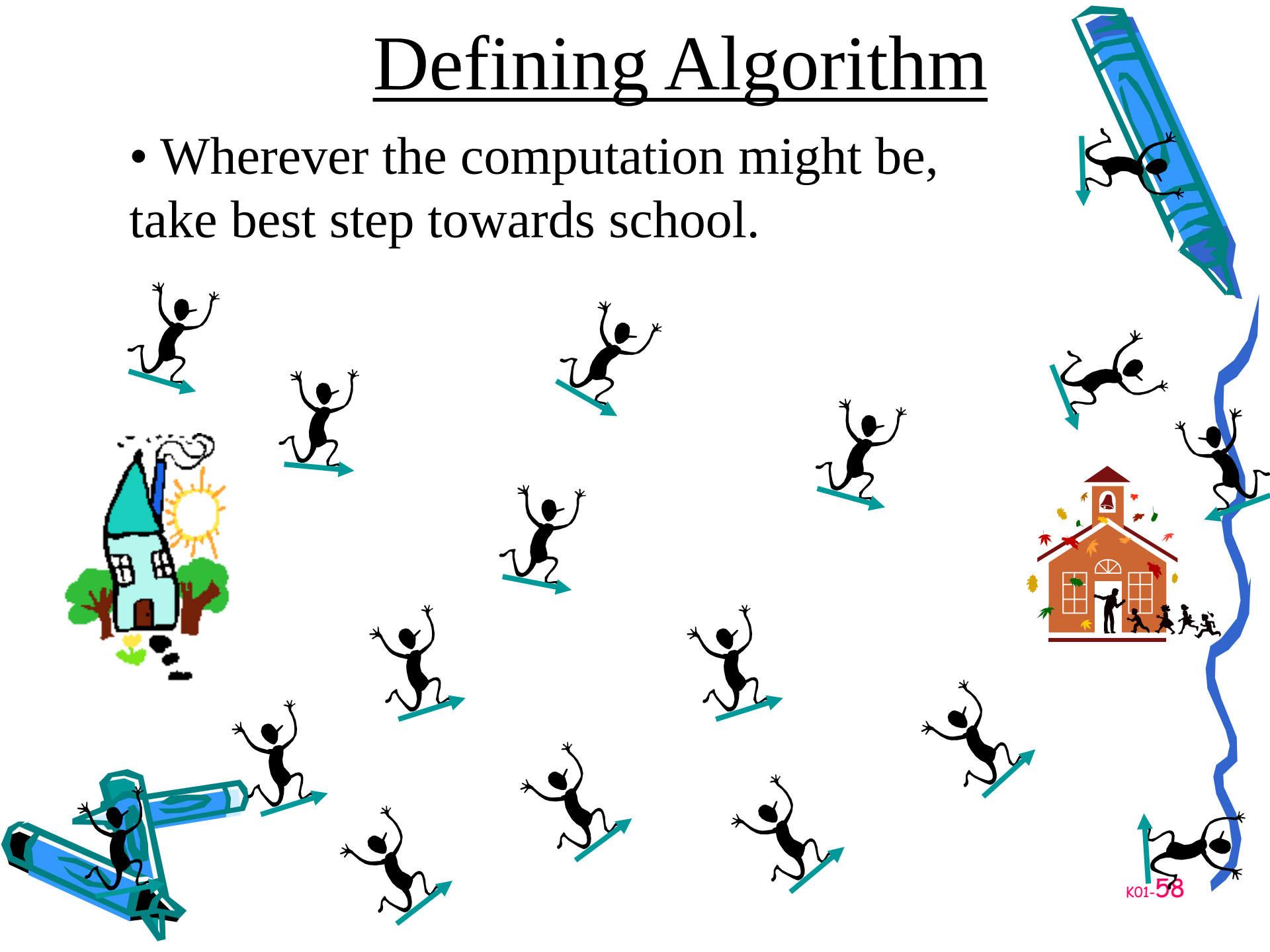
General Principle

- Do not worry about the entire computation.
- Take one step at a time!



Defining Algorithm

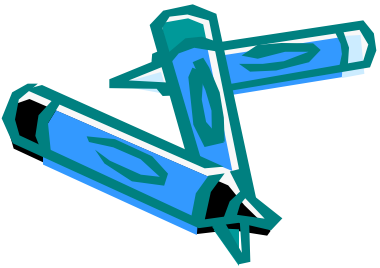
• Wherever the computation might be, take best step towards school.



Take a step

- What is required of this step?

A Measure of Progress



79 km

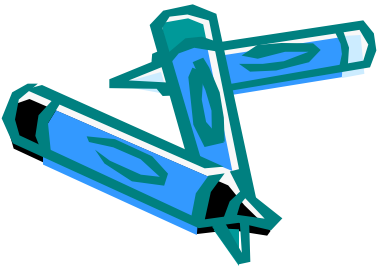
to school

75 km

to school

Defining Algorithm

- Is this sufficient to define a working algorithm?

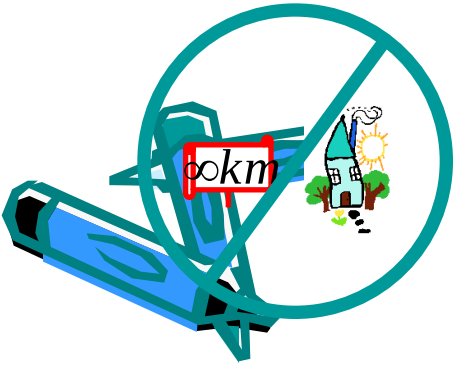


Working Algorithm

- Computation steadily approaches goal

Defining Algorithm

- Extra requirements



Define a Step

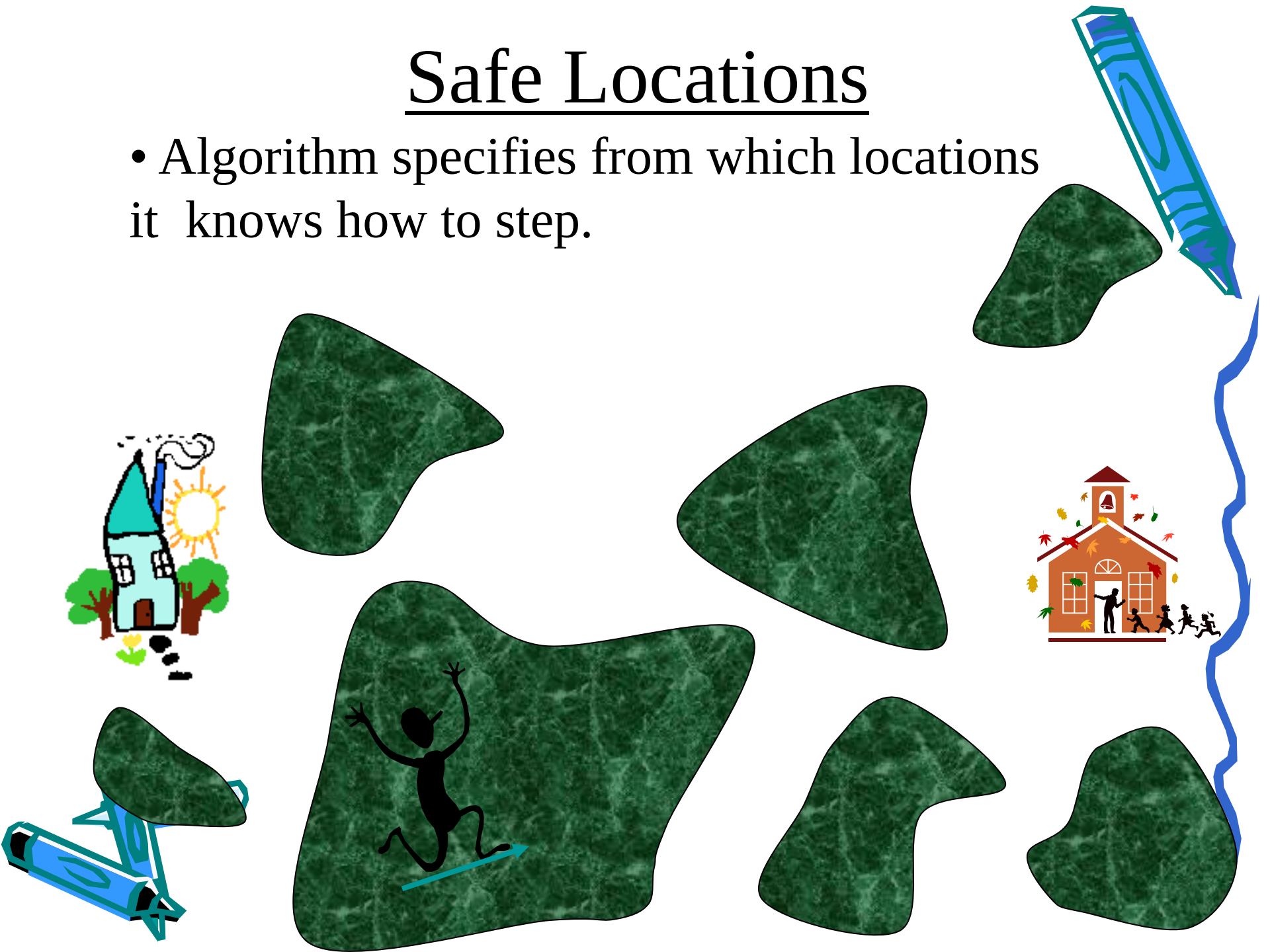
- Wherever the computation might be, take best step towards school.

Computation Stuck

- Location too confusing for algorithm
- Algorithm too lazy to define step for every location.

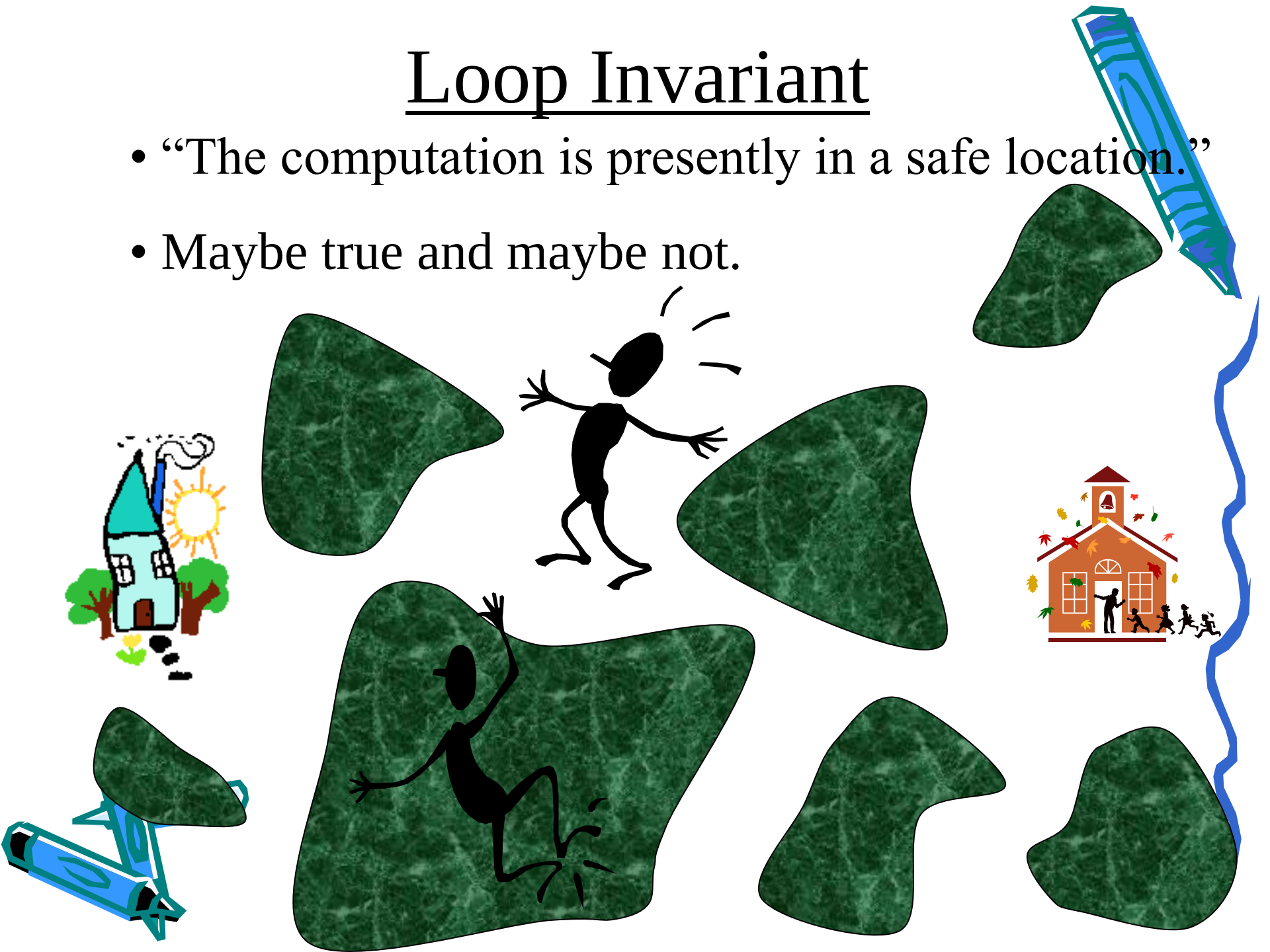
Safe Locations

- Algorithm specifies from which locations it knows how to step.



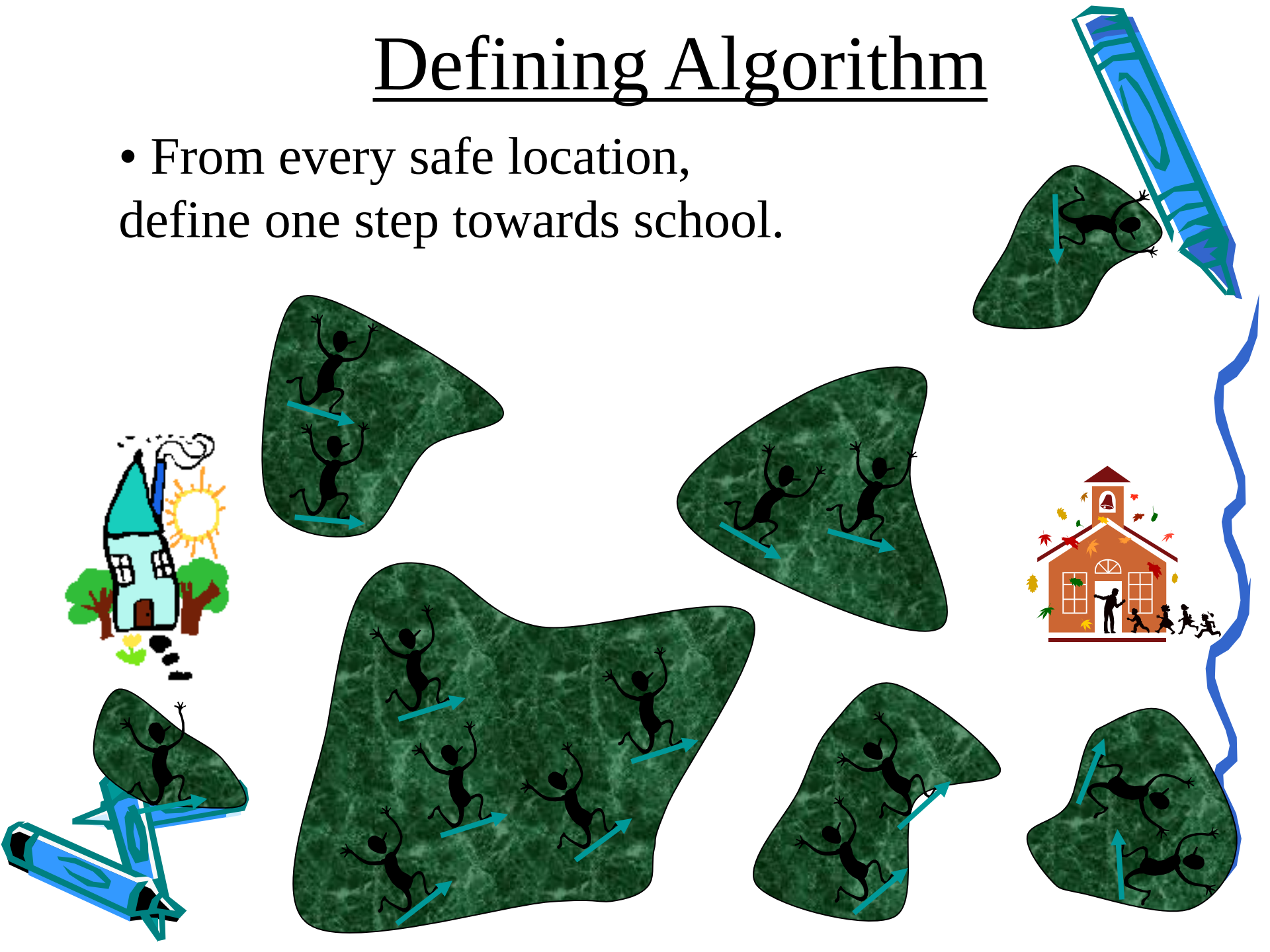
Loop Invariant

- "The computation is presently in a safe location."
- Maybe true and maybe not.



Defining Algorithm

- From every safe location, define one step towards school.

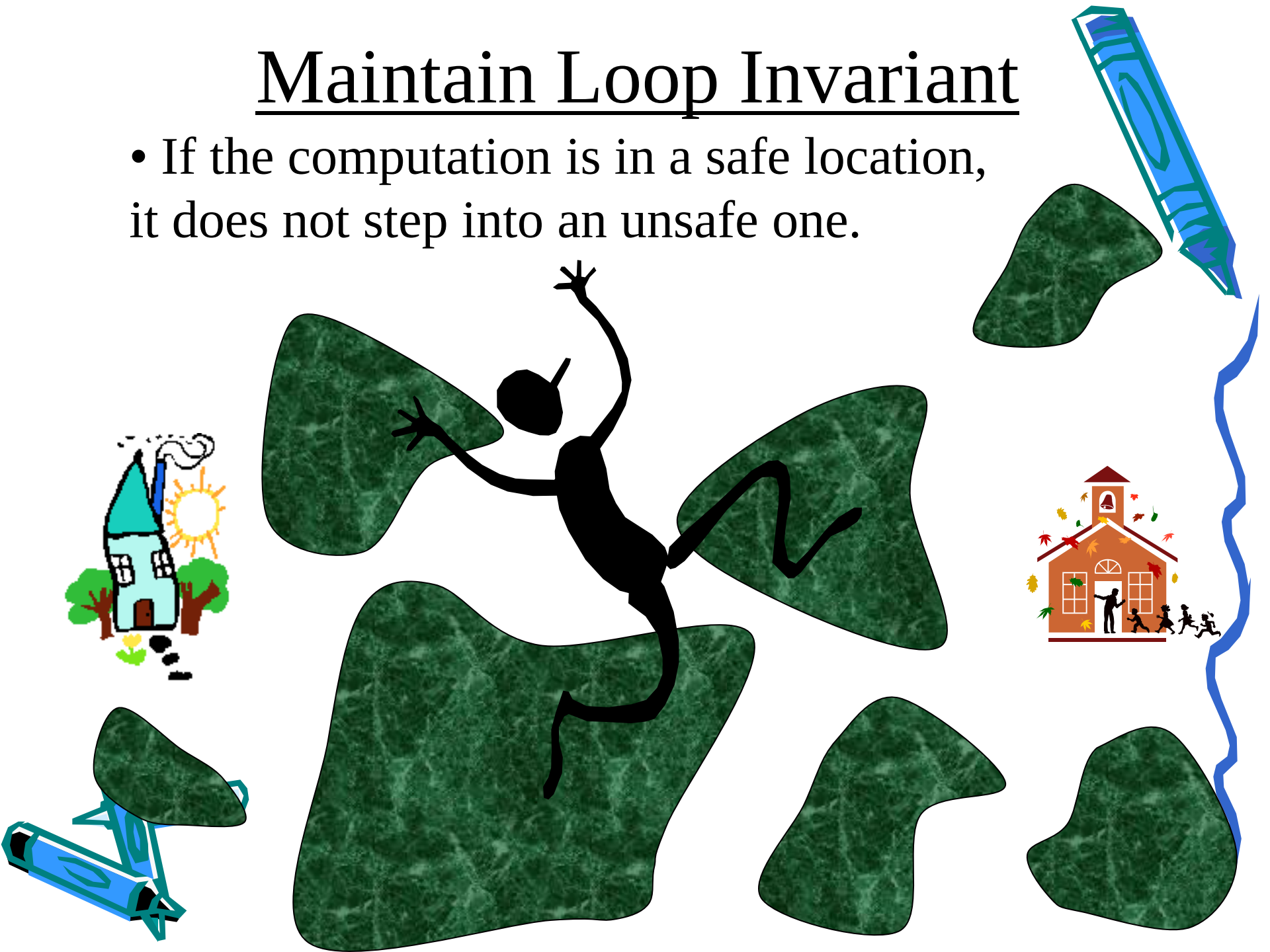


Take a step

- What is required of this step?

Maintain Loop Invariant

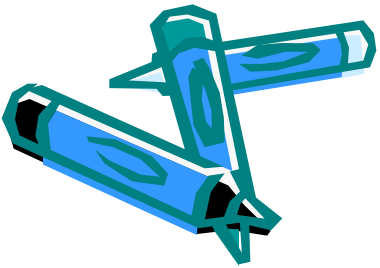
- If the computation is in a safe location, it does not step into an unsafe one.



Maintain Loop Invariant

• If the computation is in a safe location, it does not step into an unsafe one.

• Can we be assured that the computation will always be in a safe location?



Maintain Loop Invariant

• If the computation is in a safe location, it does not step into an unsafe one.

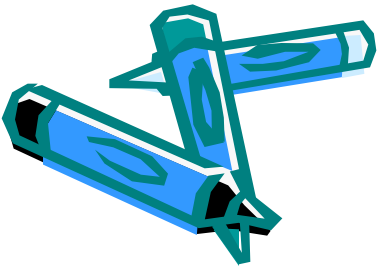
• Can we be assured that the computation will always be in a safe location?

No. What if it is not initially true?

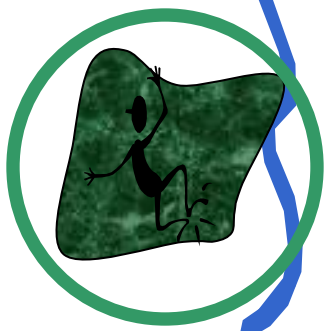
Establishing Loop Invariant

From the Pre-Conditions on the input instance we must establish the loop invariant.

Maintain Loop Invariant



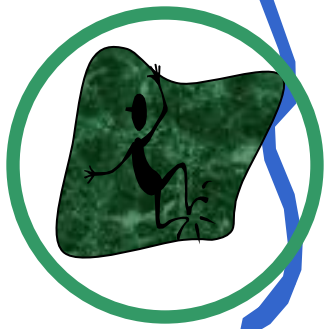
- Can we be assured that the computation will always be in a safe location?
- By what principle?



Maintain Loop Invariant

- By Induction the computation will always be in a safe location.

$$\left.\begin{array}{l} \Rightarrow S(0) \\ \\ \\ \Rightarrow \forall i S(i) \Rightarrow S(i+1) \end{array}\right\} \Rightarrow \forall i S(i) \Rightarrow$$

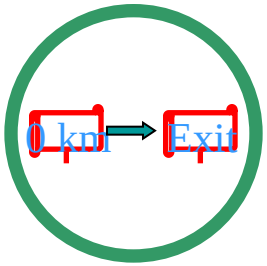


Ending The Algorithm

Define Exit Condition

Termination:

With sufficient progress,
the exit condition will be met.

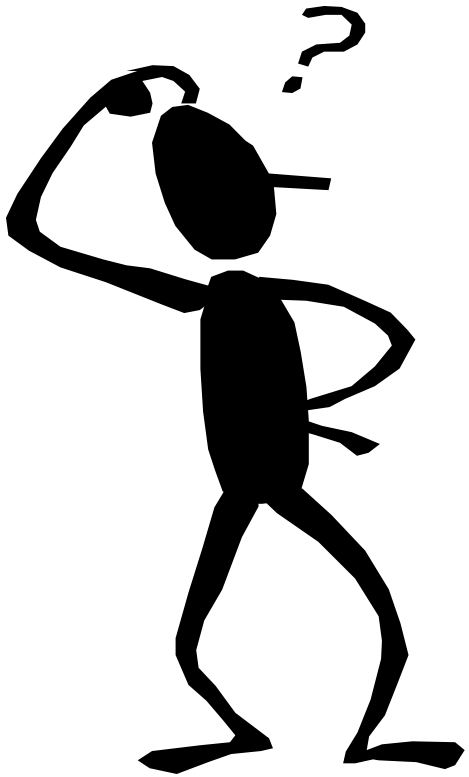


When we exit, we know

- exit condition is true
- loop invariant is true

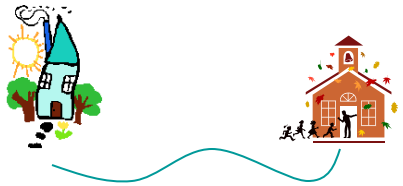
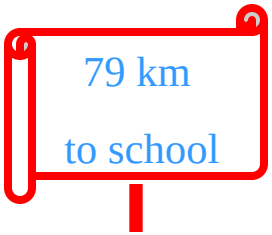
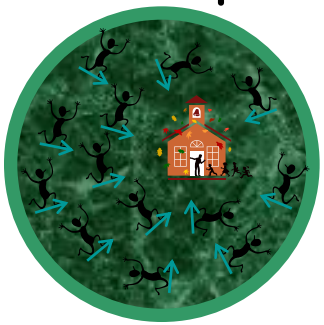
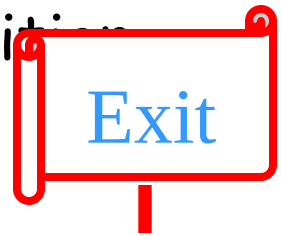
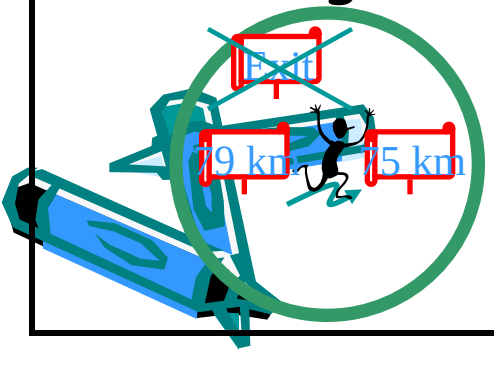
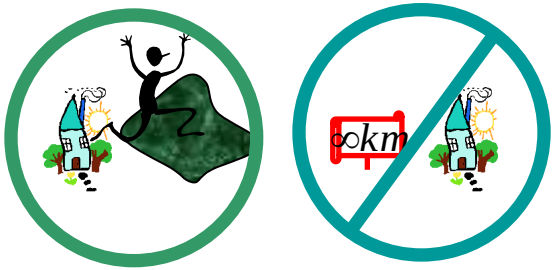
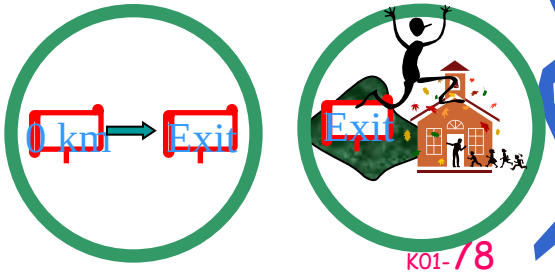
from these we must establish
the post conditions.

Let's Recap

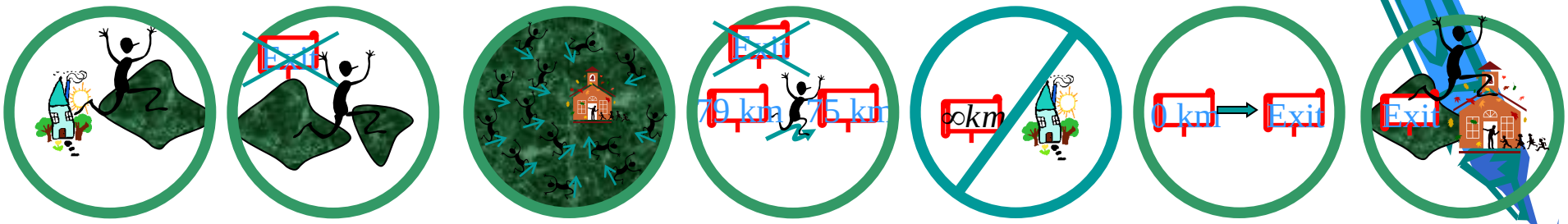


Designing an Algorithm

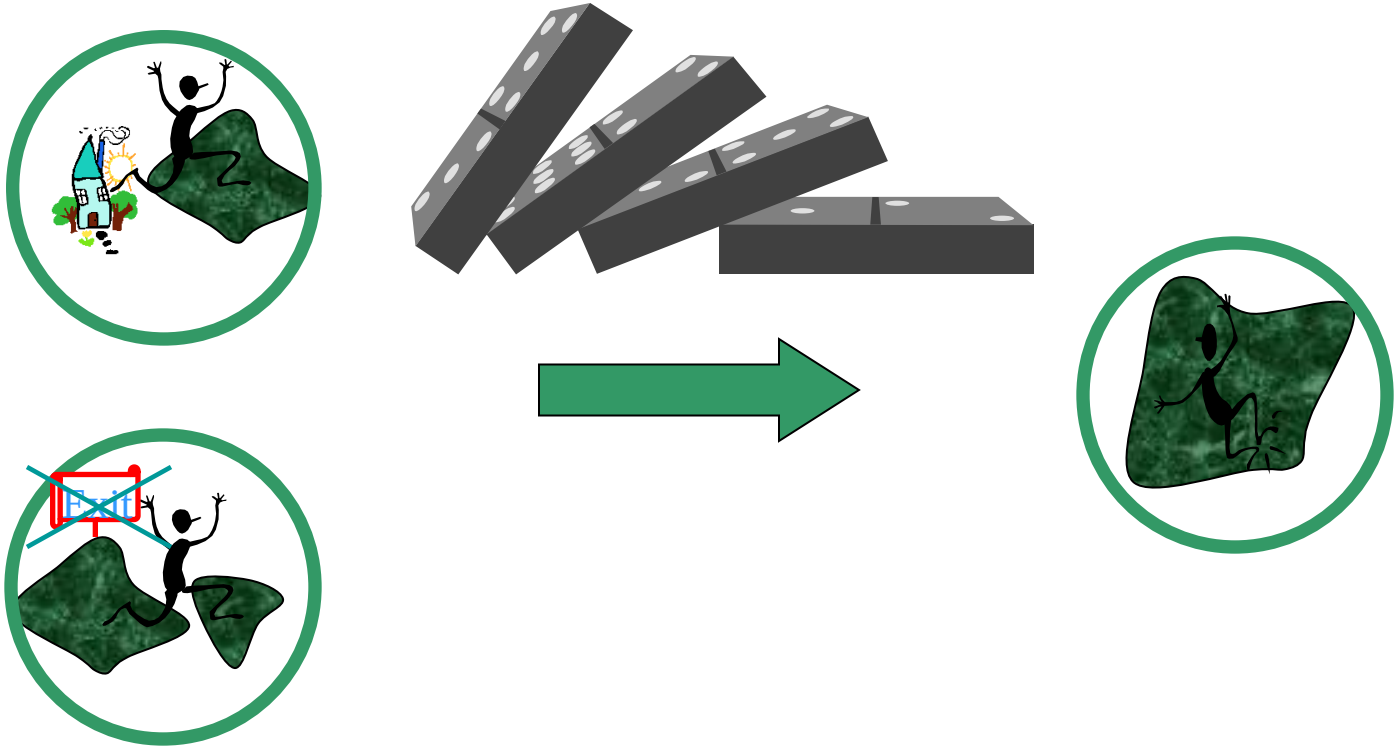
Is this sufficient?

Define Problem 	Define Loop Invariants 	Define Measure of Progress 
Define Step 	Define Exit Condition 	Maintain Loop Inv 
Make Progress 	Initial Conditions 	Ending 

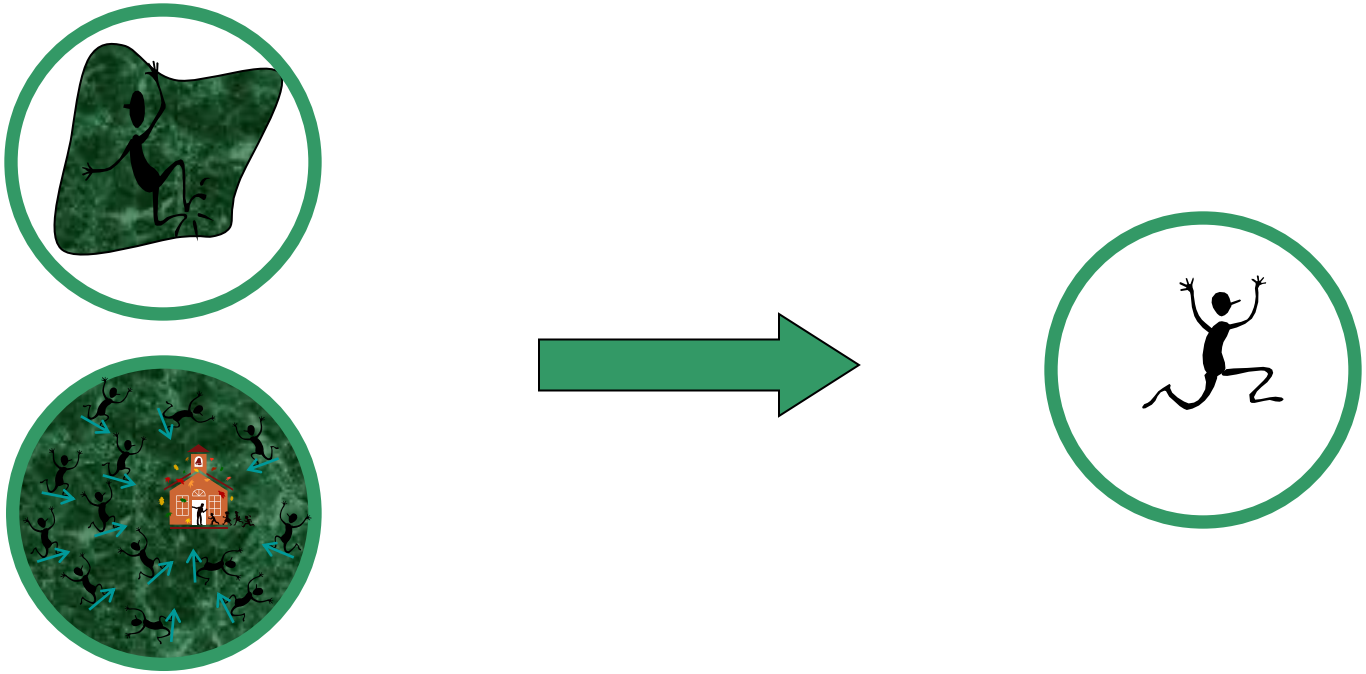
Consider an Algorithm



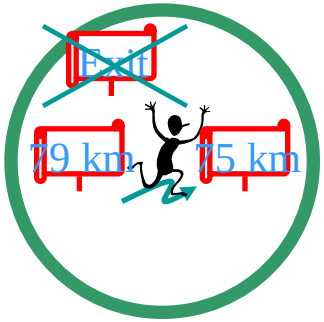
Loop Invariant Maintained

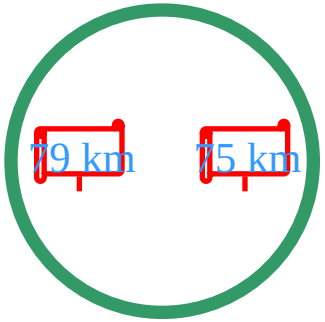


Computation can always proceed

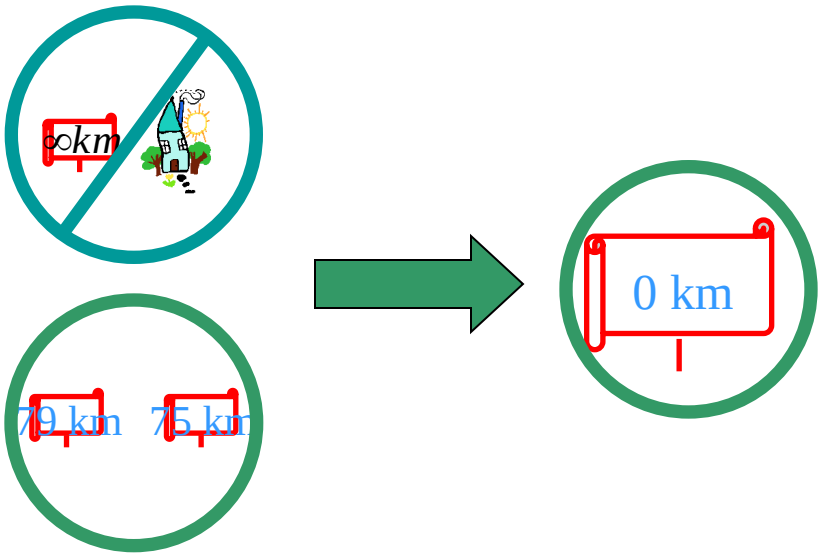


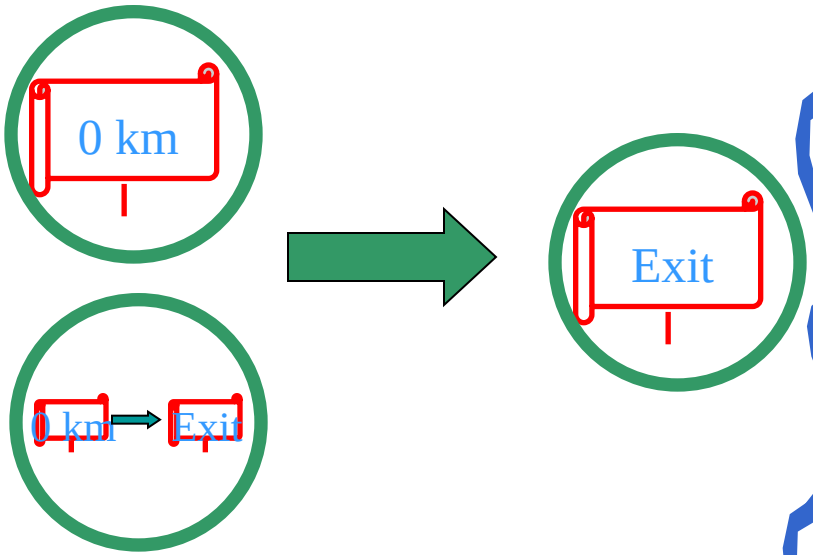
Computation always makes progress



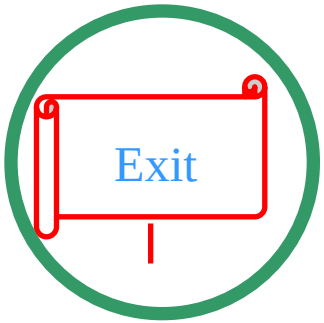


Computation Terminates

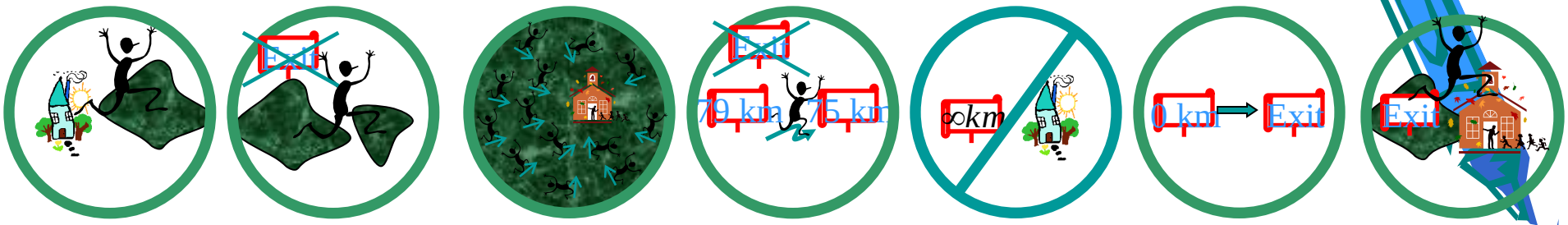




Computation Terminates



Consider an Algorithm



This is sufficient!