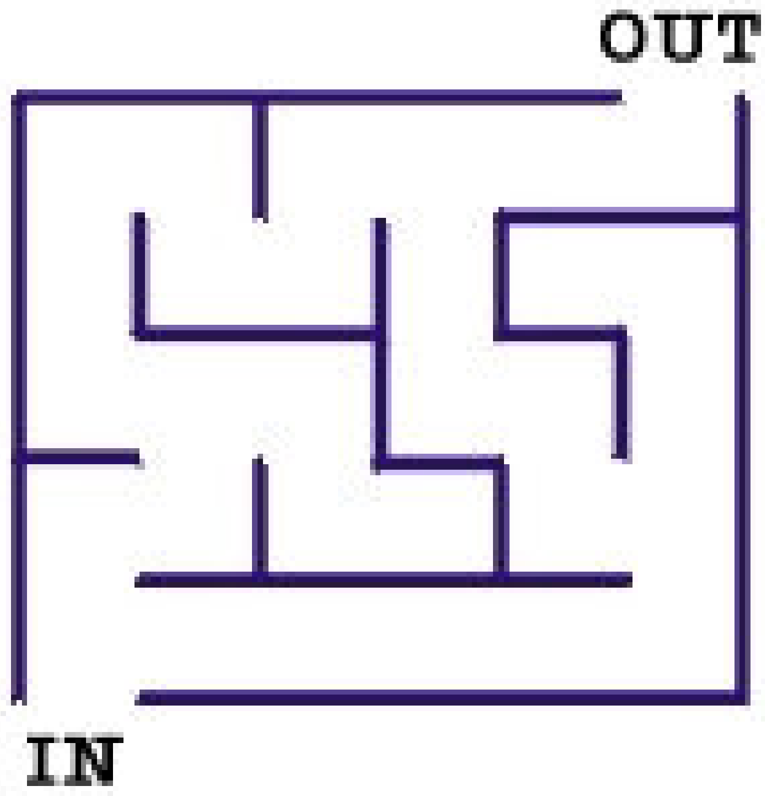
05

Algorithm & Programming Paradigms

Pengantar Teknik Informatika (HUG1M2)

20131

How to solve this maze?



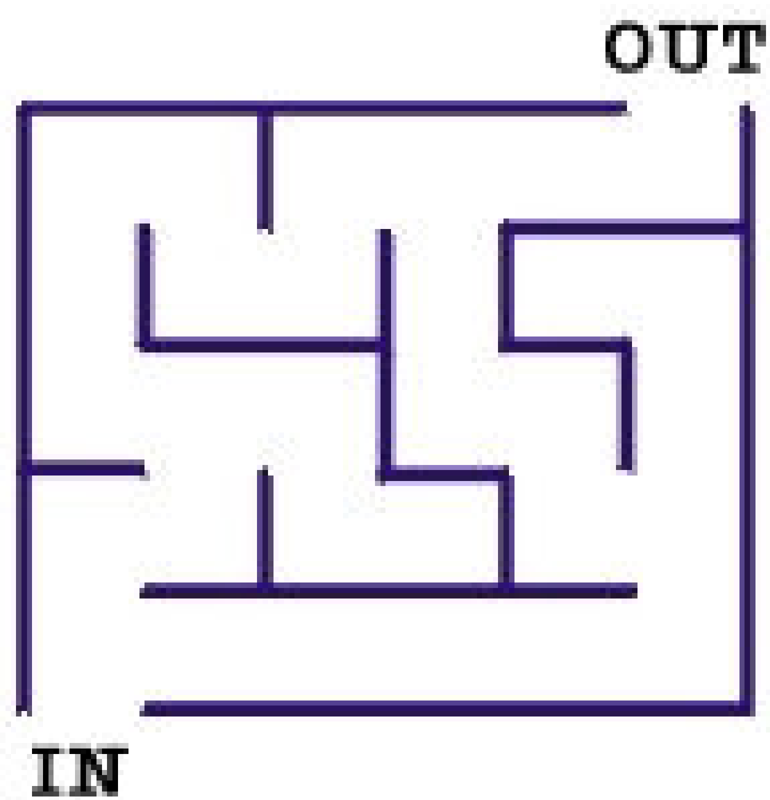
http://www.themazeproject.co.uk/maze-designs/simple-artistic-designs/

Solve these kind of mazes?

http://www.thegardenmaze.com/

http://www.zastavki.com/eng/World/France/wallpaper-22035.htm

How **TO TELL COMPUTER** to solve this maze?



http://www.themazeproject.co.uk/maze-designs/simple-artistic-designs/

Algorithm

- An algorithm is **a set of instructions** that can be **followed precisely** to achieve some **objective**.
- **I**nput – **P**rocess – **O**utput paradigm

The essential properties of an algorithm:

- an algorithm is **finite** (w.r.t: set of instructions, use of resources, time of computation)
- instructions are **precise** and **computable**.
- instructions have a specified **logical order**:
  - **deterministic** algorithms (every step has a well-defined successor), and
  - **non-deterministic** algorithms (randomized algorithms, but also parallel algorithms!)
- produce a **result**.

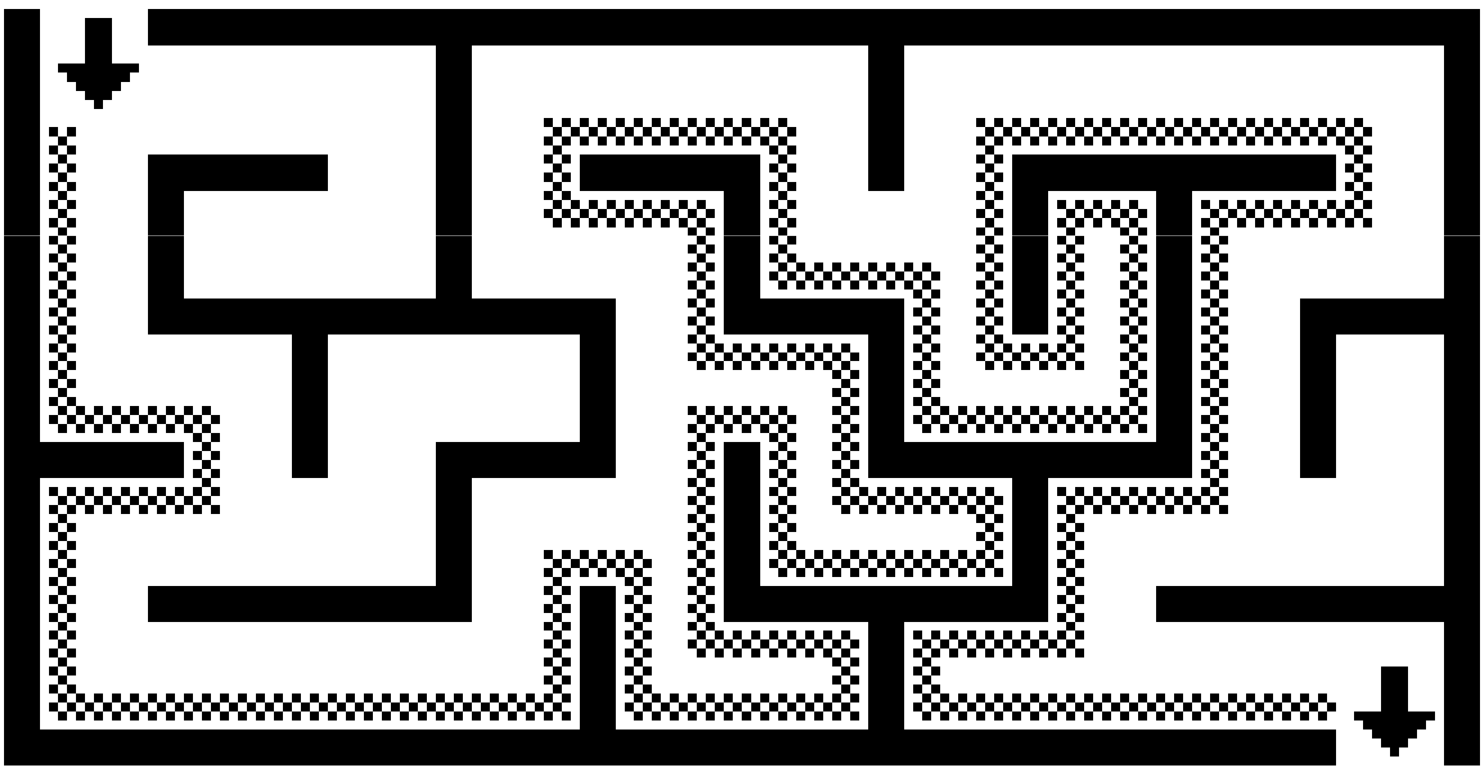
Algorithm = Program?

- Algorithms are **a way of studying programs** in a way that is **independent of implementation details**, such as the programming language or computer hardware

Specifying an algorithm

- Using natural language
- Using flowchart
- Using pseudocode
- Using program source code
- …

Natural Language
Maze alg: wall follower



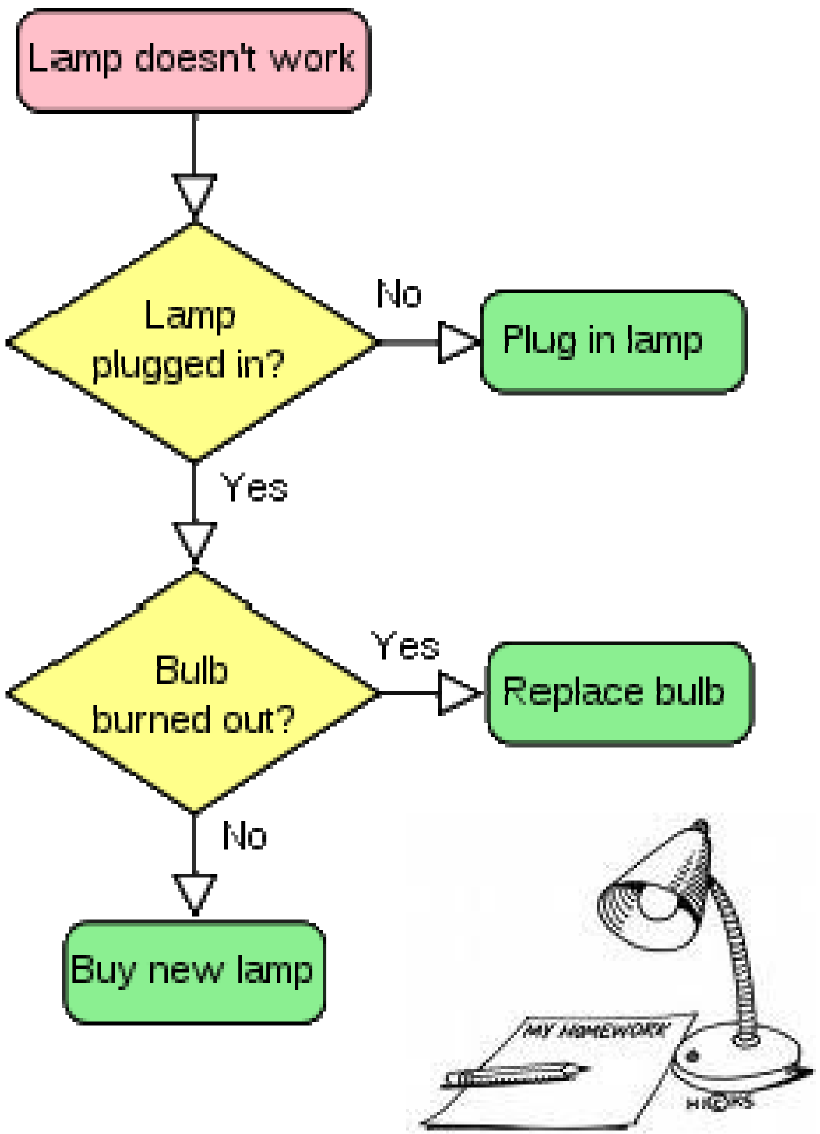
http://en.wikipedia.org/wiki/Maze_solving_algorithm

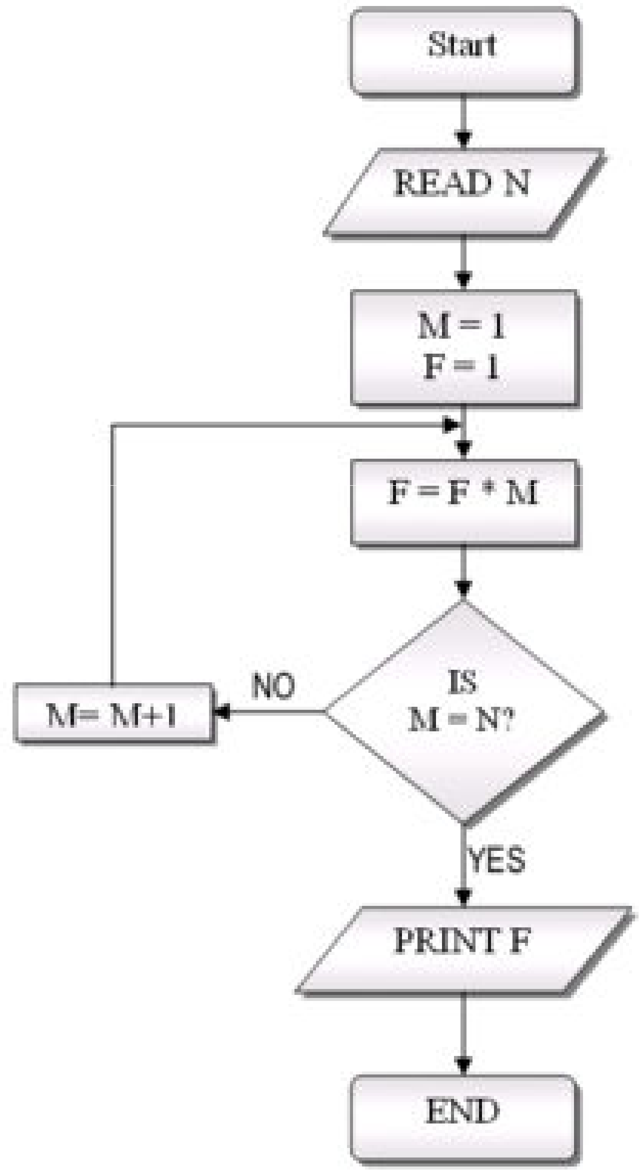
Natural Language: Caffe Latte

- **Bahan**
  - 1/2 cangkir susu (full cream atau non-fat milk)
  - 1/3 cangkir kopi espresso panas
- **Cara Membuat**
  - Panaskan terlebih dahulu susu dalam panci dengan api kecil. Lalu kocok susu dengan cepat hingga berbuih.
  - Masukkan kopi espresso yang telah diseduh ke dalam cangkir minum besar lalu tambahkan susu. Aduk rata.

http://www.kopistory.com

Flowchart: examples





Pseudocode: example

Algorithm LargestNumber

Input: A non-empty list of numbers *L*.

Output: The *largest* number in the list *L*.

largest ← L_0

for each *item* **in** the list $L_{\geq 1}$, **do**

if the *item* > *largest*,

then *largest* ← the *item*

return *largest*

Using program source code

```
program multiplication
var n,m:integer;
begin
  readln(n);
  readln(m);
  writeln (n*m;)
end;
```

Algorithm Qualities

Correctness (effectiveness)

- 100%
- Approx alg → the error < limit

Efficiency:

- Time & Space efficiency

Simplicity

Generality:

- the problem
- input range

The real world

- Computers **may be fast**, but they are **not infinitely fast**
- Memory **may be cheap**, but it is **not free**.

- Bounded resources:
  - Computing time
  - Space in memory
  - Energy (mind your laptop battery) etc.

- These resources must be used wisely, and **efficient algorithms** will help you do so

Efficiency

- Performance: the amount of CPU / memory / disk usage / energy etc.
- **Complexity**: how well the algorithm scales

- Big-O
  - the number of operations required to perform a function
  - expression representing some growth relative to the size of the problem (N)
  - Exp: O(1), O(N), O(N^2), O(log N), …

O(1)

- an algorithm takes constant time to run;
  - performance isn’t affected by the size of the problem

- Exp:
  - addressing main memory in a computer
  - accessing Array Index (int a = ARR[5];)
  - inserting a node in Linked List
  - Pushing and Poping on Stack
  - Insertion and Removal from Queue

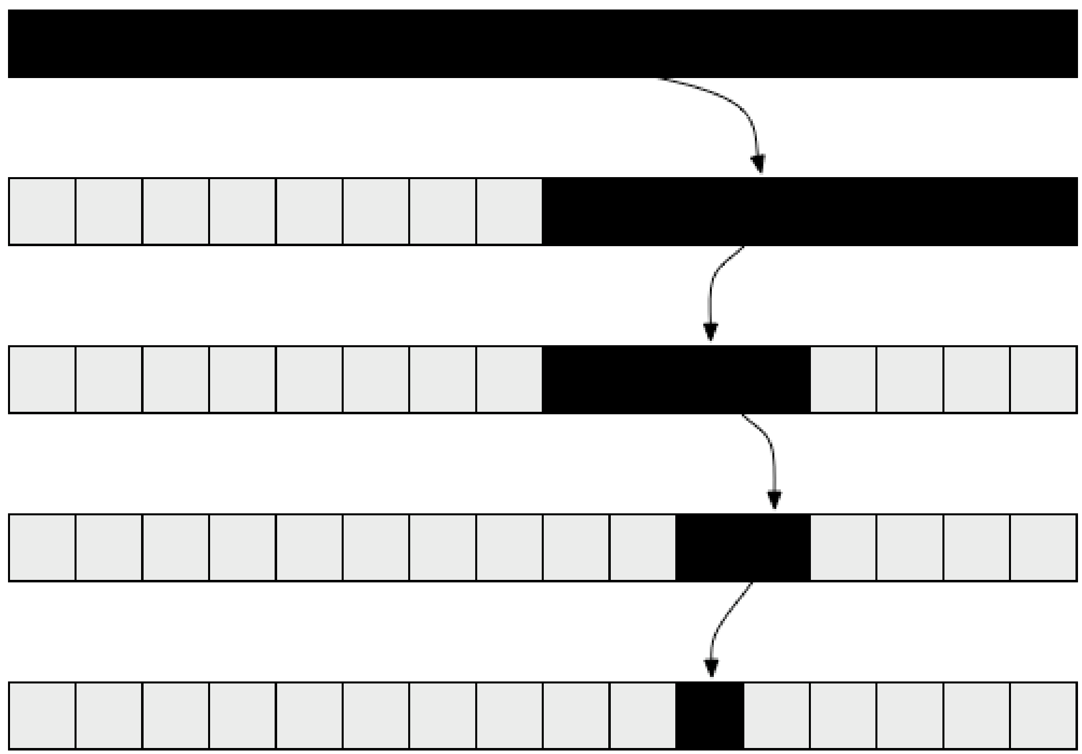
O(N)

- the number of operations required to perform a function is directly proportional to the number of items being processed

- Exp:
  - waiting in a line at a supermarket
    - Assume: 2 mins / cust (avg)
    - 10 cust → 20 mins; 100 cust → 200 mins
  - Traversing an array
  - Traversing a linked list
  - Linear Search

O(log N)

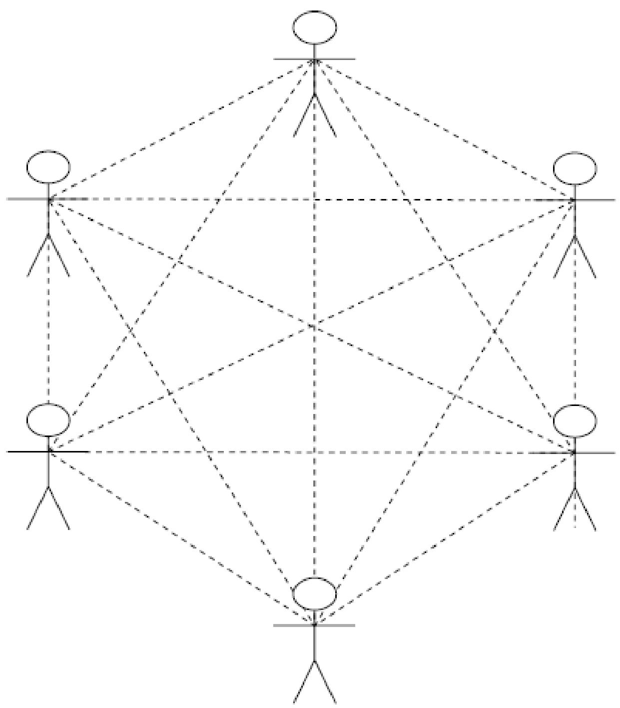
- Example: Finding an item in a sorted array with a binary search



http://discrete.gr/complexity/

$O(N^2)$

- Each member of the group greets every other member
  - 6 persons → 5+4+3+2+1 = 15
  - 7 persons → 21
  - 8 persons → 28
  - …
  - N persons → $(N^2-N)/2$ greets



$O(N^2)$: $(N^2-N)/2 \rightarrow N^2$

- Big O disregard any constant → (N^2-N)
- as N becomes larger, subtracting N from N^2 will have less and less of an overall effect

N	N^2	$N^2 - N$	Difference
1	1	0	100.00%
10	100	90	10.00%
100	10,000	9,900	1.00%
1,000	1,000,000	999,000	0.10%
10,000	100,000,000	99,990,000	0.01%

Implementation Complexity

- Fast algorithms often make use of very complicated data structures, or use other complicated algorithms as subroutines

- Challenge: making more complicated algorithms worthy of consideration in practice

Some Algorithmic strategies

- Brute-force
- Greedy
- Divide-and-conquer
- Backtracking
- Branch-and-bound
- Heuristics
- Pattern matching and string/text
- Dynamic Programming
- Numerical approximation …

Greedy & Brute-force

Soal kembalian minimum:

- Input:
  - nominal uang = 25, 10, 5, 1
  - bayar = 50
  - beli = 18
- Output:
  - kembalian = 25, 5, 1, 1

Write your own algorithm

Test case

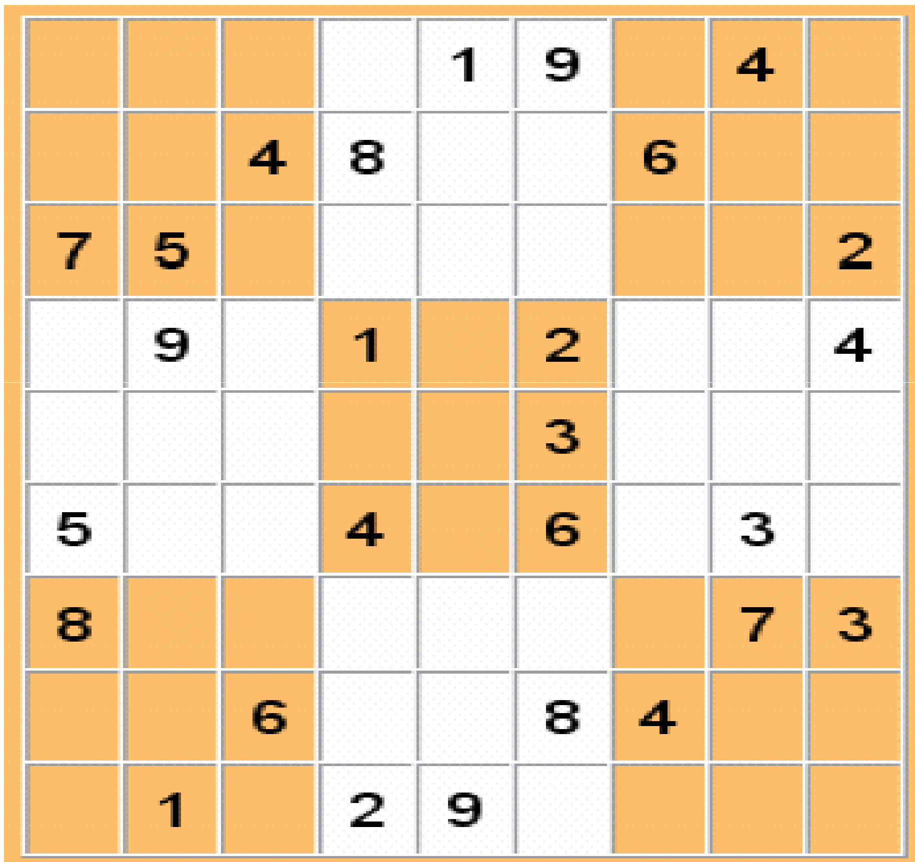
- Input:
  - nominal = 15, 10, 1
  - bayar = 25
  - beli = 5

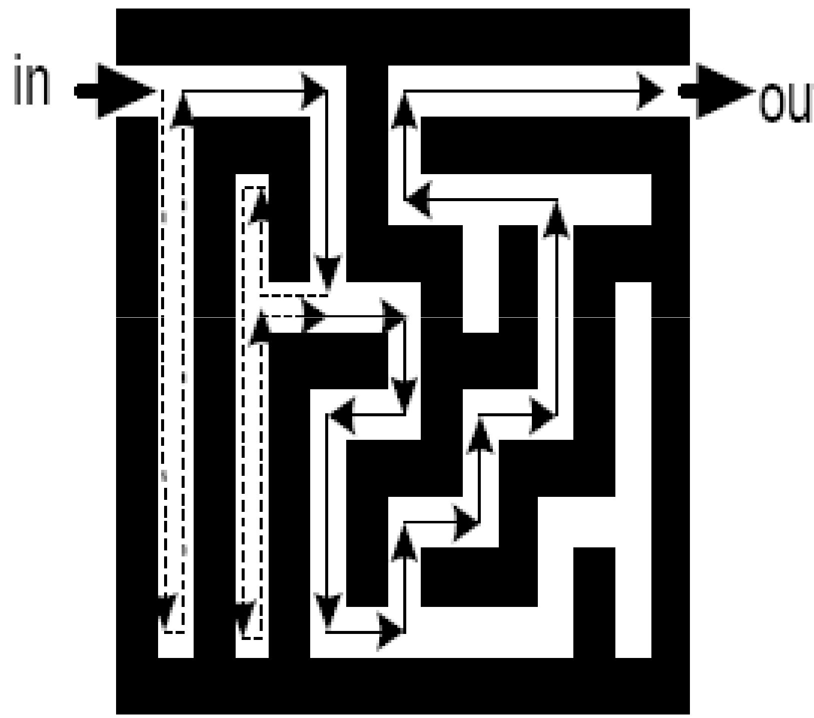
- Output:
  - kembalian = ???

kembalian = **15,1,1,1,1,1** → Greedy

kembalian = **10,10** → Brute-Force

Backtracking





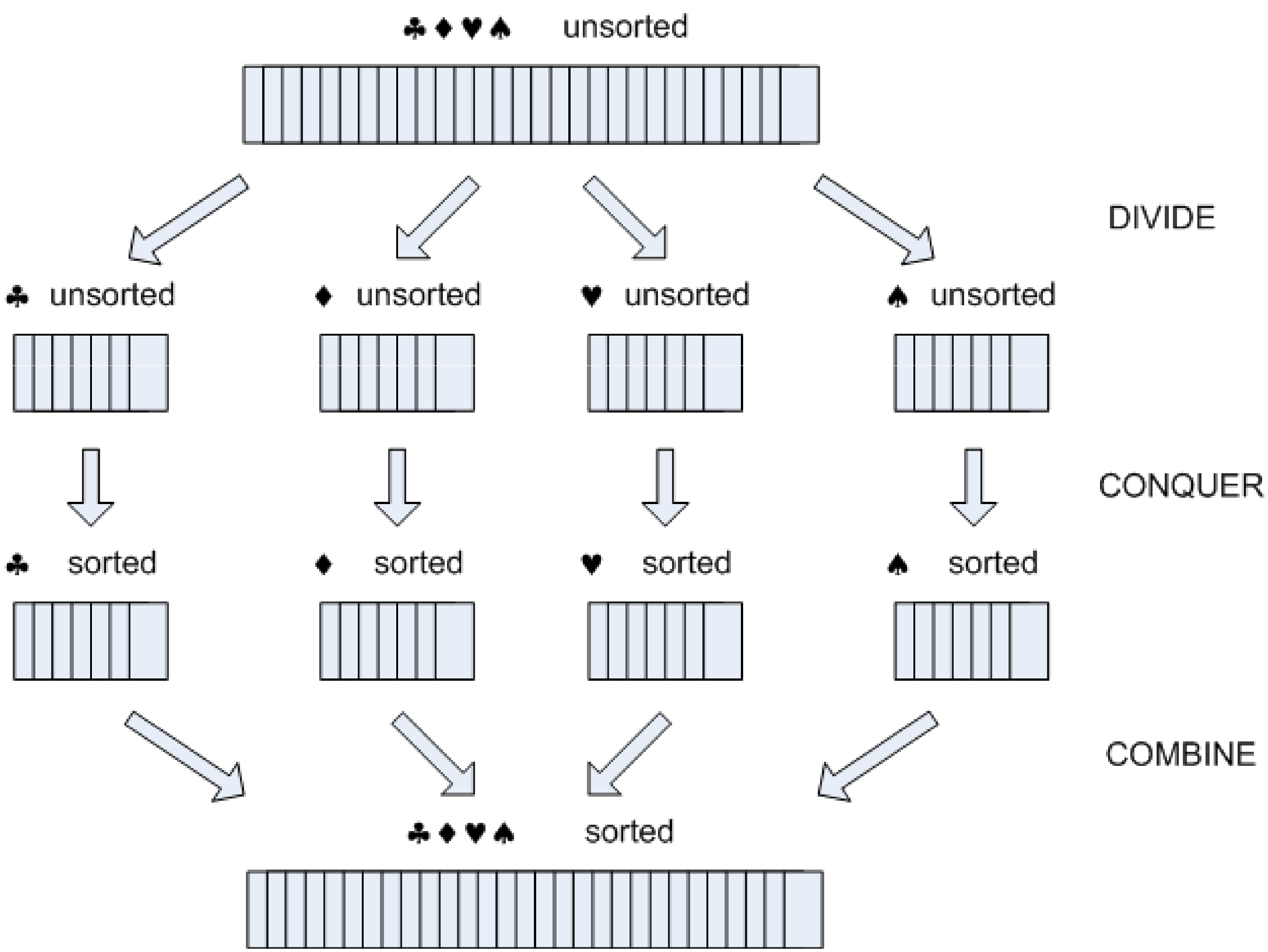
Heuristics

Tanjung Priok
Kedaung
Candi Batujaya
Batujaya
Tanjungpasir
JAKARTA
Sungaibuntu
Kosambi
Taman Impian Jaya Ancol
Tanjung Priok
Babakan
Rengasdengklok
Sumur Gede
Kramat
Srengseng
Cilamaya
Cengkareng
Pamanukan
Batuceper
Pulogadung
Sengir
Wadas
Bekasi
Cikarang
Karawang
Sukamandi
Binong
Curug
Ciredak
Kosambi
Lebakbulus
Kp. Rambutan
Leuweungmalang
Water Boom Cikarang
Cikampek
Pabuaran
Pagadenba
Serpong
Simpang Jomin
Cipanas
Cikopo
Cibubur
Cileungsi
Sawangan
Depok
Cibarusah
Pangkalan
Kalijati
Sadang
Parung
Jonggol
Cibinong
Cariu
Purwakarta
Bojong
Citeureup
Jatiluhur Resort
Danau Jatiluhur
Ketapang
Nanggerang
Pasir Kalong
Ciampea
Bogor
Sagalaherang
Jalancagak
Kedungmanggu
Bojong Kole
Plered
Prasasti Ciaruteun
Gn. Paper
Wanayasa
Cisalak
Ciparahu
Leuwiliang
Kebun Raya Bogor
Curug Cilember
Darangdan
Gn. Tangkuban Perahu
Ciater
Ciawi
Cisarua
Taman Bunga Nusantara
Cikalongkulon
Cikalongwetan
Sari Ater
Cibago
Gudung Mejan
Puncak
Cipanas
Curug Luhur
Taman Safari
Lembang
Curug Cileat
Cianten
Gn. Salak
Gn. Pangrango
Cianjur
Ciranjang
Cisarua
Padalarang
Gn. Gede
Gua Jepang
Curug Omas
Tanjung
Cicurug
Taman Nasional Pangrango/Gede
Cugenang
Cimahi
Ujungberung
Cileunyi
Batujajar
Danau Saguling
Cipetir
Parungkuda
Situ Gunung
BANDUNG
Sukabumi
Cililin
Cibadak
Cibeber
Soreang
Ciparay
Banjaran
Gunungkalum

Divide-and-conquer

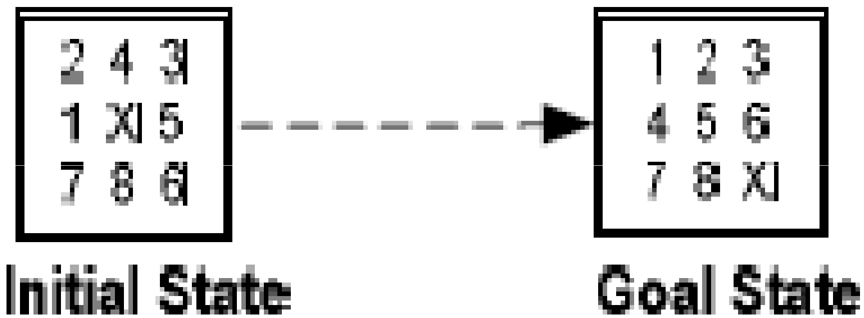
- Divide – conquer - combine

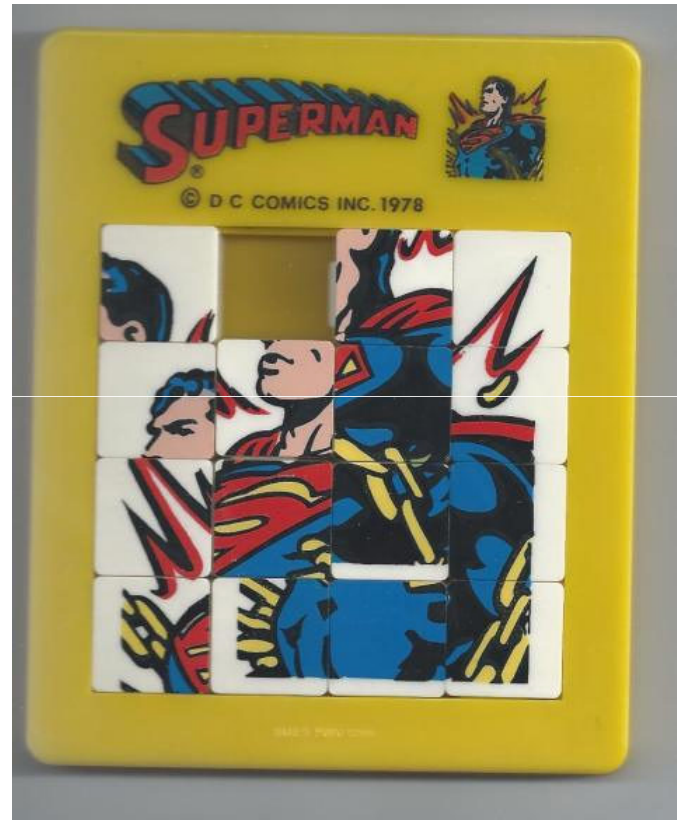
Card Sorting: divide-and-conquer



Branch-and-bound

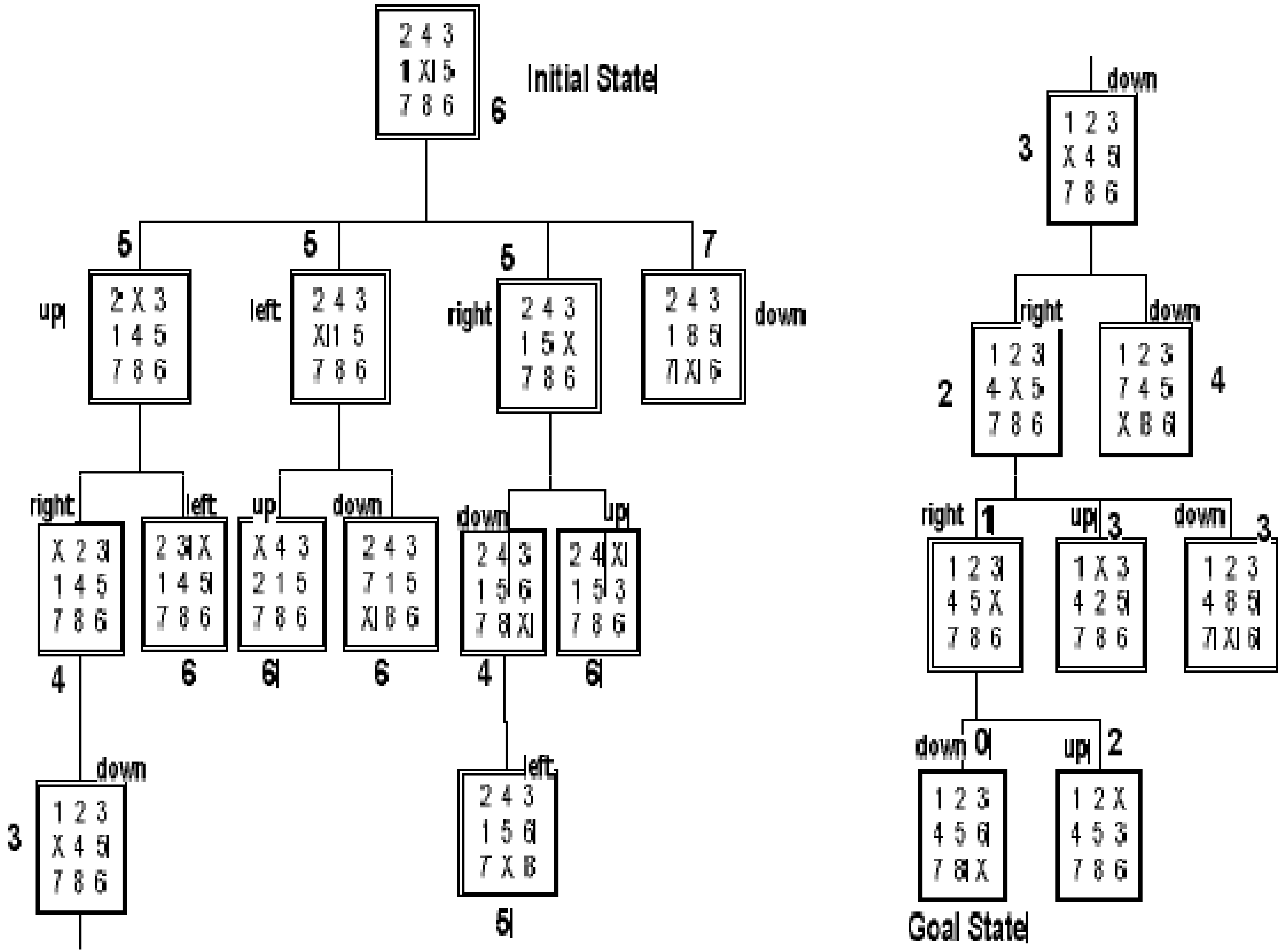
- 8-puzzle





http://timquilts.com/2012/07/26/fan-quilt-layout-2/

- g(i) : jarak kotak yang salah ke kotak yang sebenarnya



Initial State
2 4 3
1 X 5
7 8 6
6
5
up
2 X 3
1 4 5
7 8 6
5
left
2 4 3
X 1 5
7 8 6
5
right
2 4 3
1 5 X
7 8 6
7
down
2 4 3
1 8 5
7 X 6
right
X 2 3
1 4 5
7 8 6
4
left
2 3 X
1 4 5
7 8 6
6
up
X 4 3
2 1 5
7 8 6
6
down
2 4 3
7 1 5
X 8 6
6
down
2 4 3
1 5 6
7 8 X
4
up
2 4 X
1 5 3
7 8 6
6
down
1 2 3
X 4 5
7 8 6
3
left
2 4 3
1 5 6
7 X 8
5
down
3
1 2 3
X 4 5
7 8 6
right
2
1 2 3
4 X 5
7 8 6
down
1 2 3
7 4 5
X 8 6
4
right
1
1 2 3
4 5 X
7 8 6
up
3
1 X 3
4 2 5
7 8 6
down
3
1 2 3
4 8 5
7 X 6
down
0
1 2 3
4 5 6
7 8 X
Goal State
up
2
1 2 X
4 5 3
7 8 6

4 Main Programming Paradigms

Imperative

- Foundation: Turing machine

Object-oriented

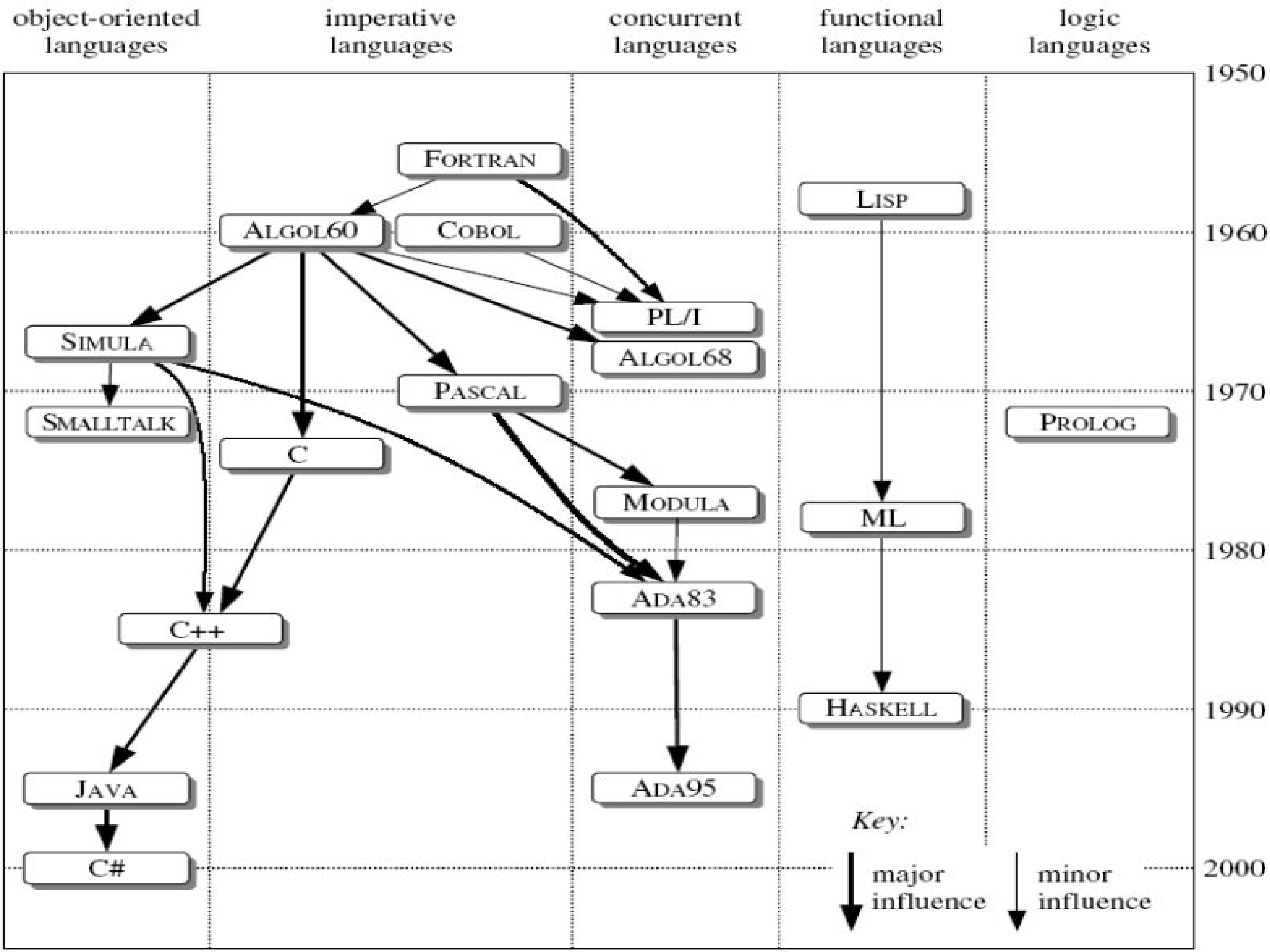
- Foundation: Turing machine

Functional

- Foundation: lambda calculus

Declarative

- Foundation: first order logic



object-oriented languages
imperative languages
concurrent languages
functional languages
logic languages
1950
1960
1970
1980
1990
2000
FORTRAN
ALGOL60
COBOL
LISP
SIMULA
PL/I
ALGOL68
PASCAL
SMALLTALK
PROLOG
C
MODULA
ML
ADA83
C++
HASKELL
JAVA
ADA95
C#
Key:
major influence
minor influence

Styles of programming language

- **Imperative**
  - the programmer states exactly how the program is to achieve its desired result → “First **do this** and next **do that”**
  - Examples: C, Basic, Pascal, Ada, …
- **Functional**
  - have been used in artificial intelligence and other research applications
  - Examples: Lisp, Scheme, Haskell, …

Styles of Prog. Lang. (cont'd)

- Logic (***declarative*** *programming*)
  - the programmer states what is the result that he or she wants to achieve, and it is up to the language as to how it achieves it
  - Example: Prolog
- **Object oriented**
  - *object* encapsulates items of *data* and the *operations* (methods) that can be performed on them
  - object is an *instance* of a *class*
  - *Send messages between objects to simulate the temporal evolution of a set of real world phenomena*
  - Examples: C++, Java, …
- …

predicate calculus

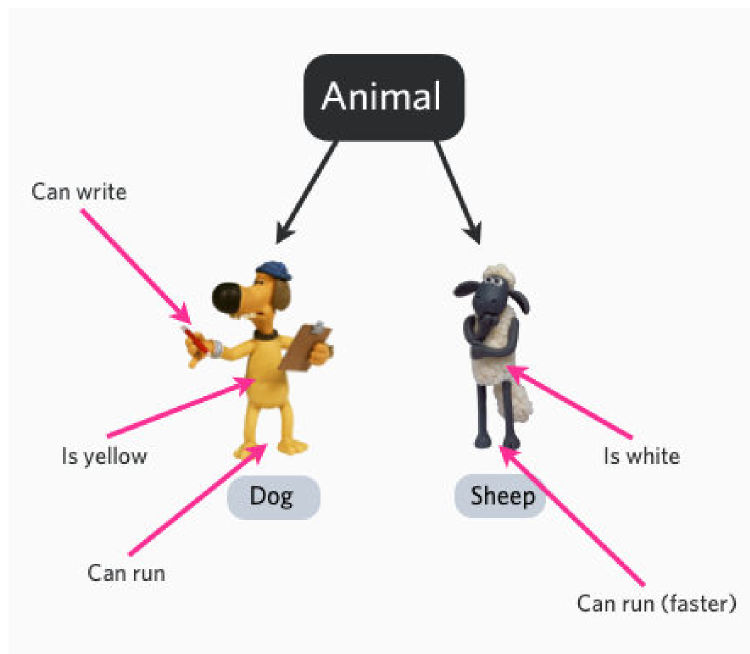
- programmer can formulate *propositions* (a logical statement which may or may not be true)

- 'Fido is a dog' → `isa(fido, dog)`
- 'a dog is an animal' → `isa(dog, animal)`
- 'Is Fido an animal?' → `?isa(animal, fido)`
- Result = *`True`*

- Example application: expert systems

OO: powerful approach

- It seems to be an approach that matches the way that people (programmers) think.
- Concept of *inheritance between classes*; reusability
- …



High level & Low level: analogy

- Instruction in a recipe:
  - ‘Make a white sauce with the butter, milk and flour’
    - → high level
  - ‘Heat the butter gently and then add the flour a bit at a time, taking care to thoroughly stir the flour in as you add it . . .’
    - → low level

Levels

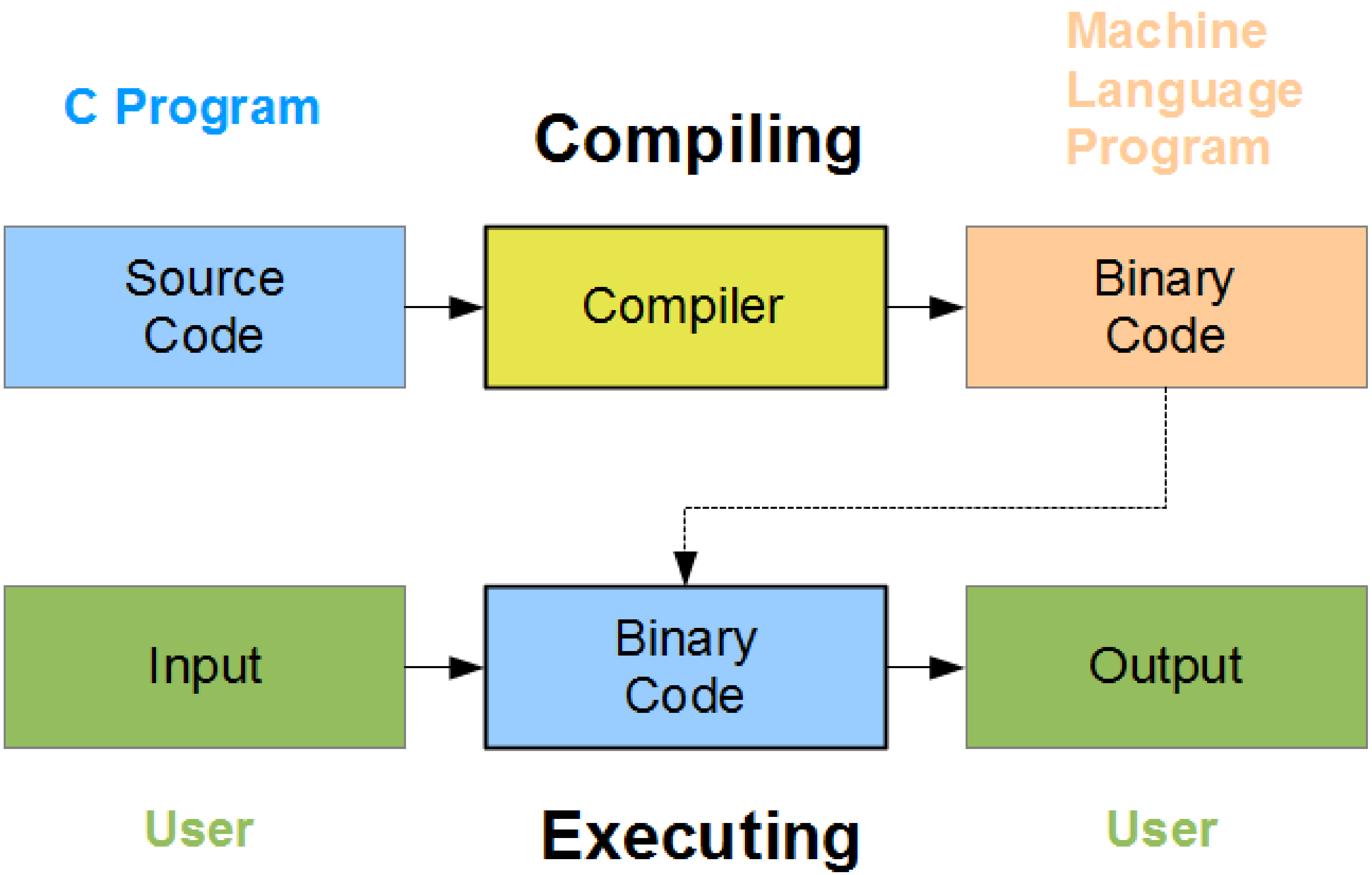
- High-level:

```
A := B + C;
```

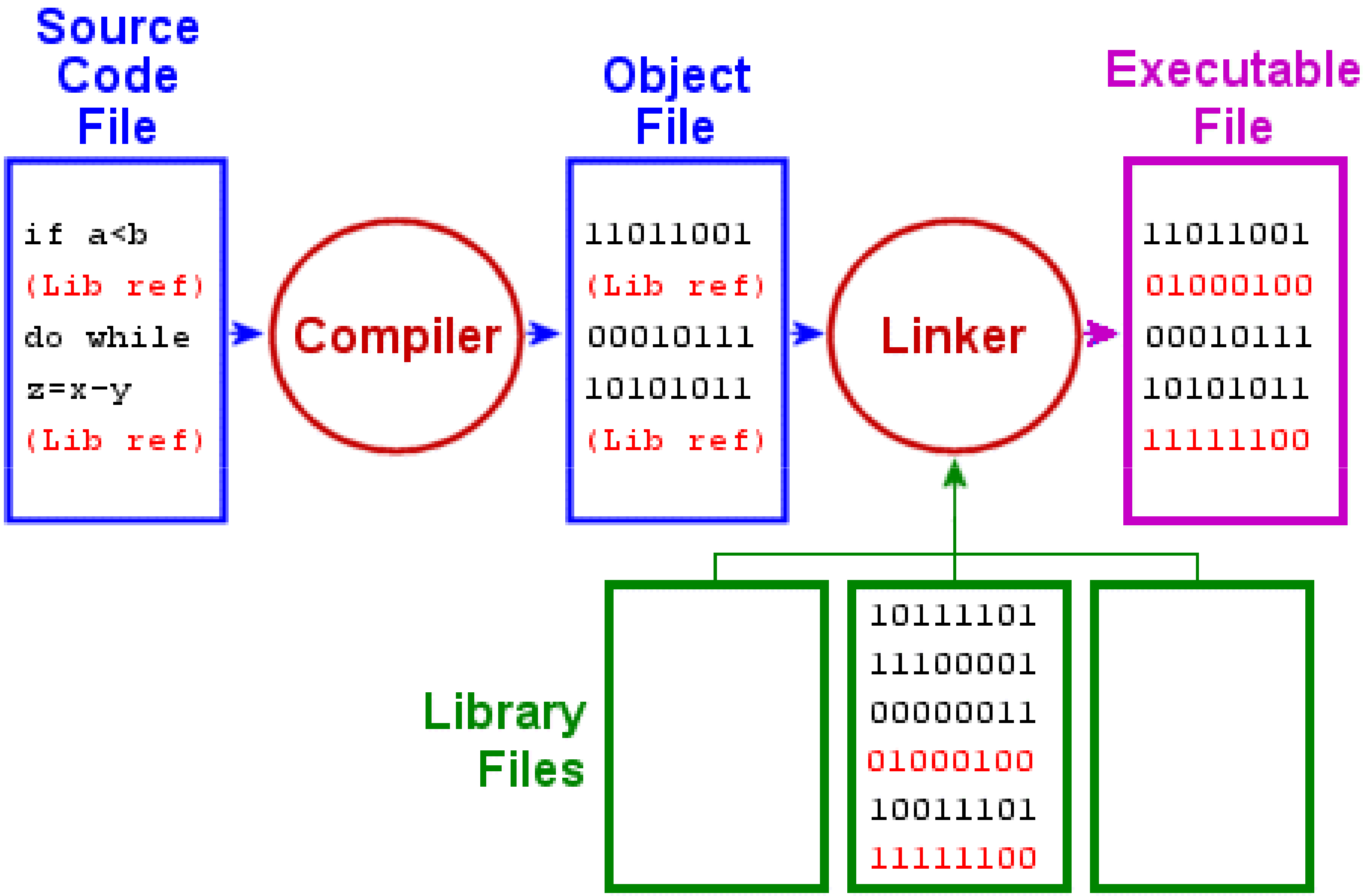
- ….
- Assembly:

```
LOAD B
LOAD C
ADD
STORE A
```

- Machine code



https://scs.senecac.on.ca/~ipc144/pages/content/compi_p.html



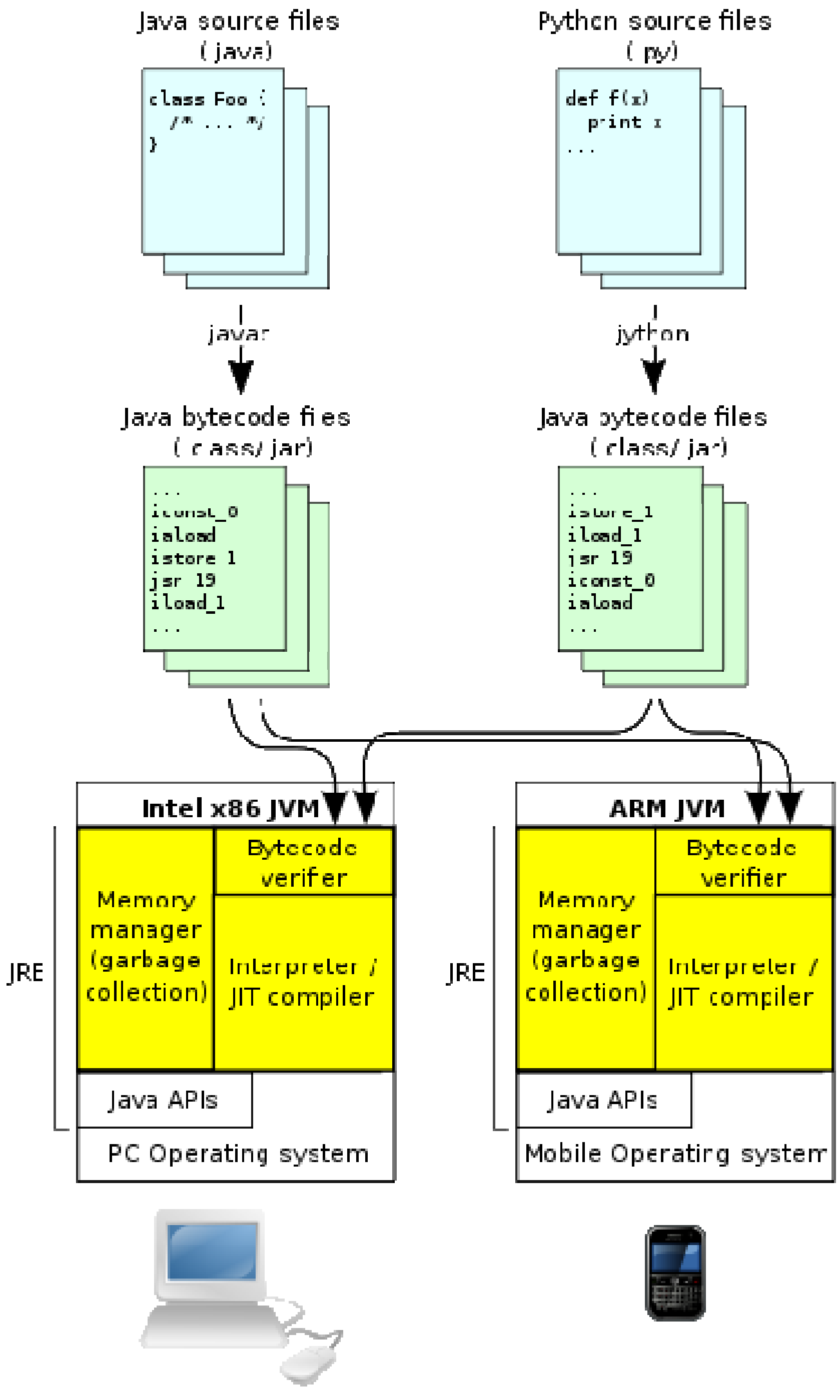
http://www.aboutdebian.com/compile.htm

Interpreter

- takes a high-level language **instruction** (one by one),
- **converts** it to a machine language instruction,
- **executes** it

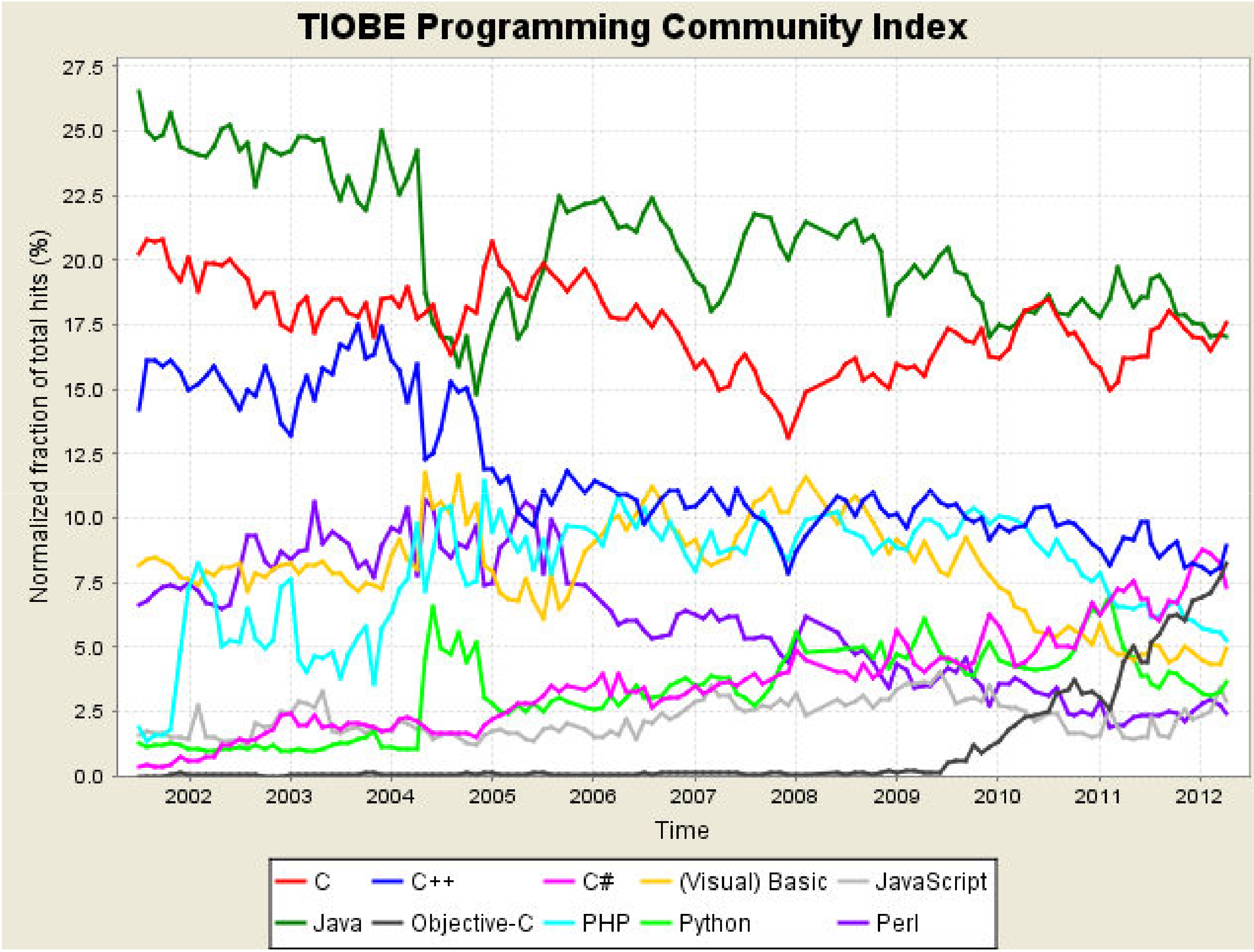
- does **NOT** save the object code

Case: JAVA



Source code is compiled to Java bytecode, which is verified, interpreted or JIT-compiled for the native architecture.
The Java APIs and JVM together make up the Java Runtime Environment (JRE).

http://en.wikipedia.org/wiki/Java_virtual_machine



www.tiobe.com