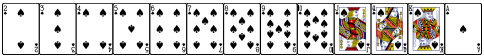
COMP 2011: Data Structures

Lecture 1. Introduction

Dr. CAO Yixin

yixin.cao@polyu.edu.hk

September, 2021



QA sessions: Weeks 2, 4, 6, 10, 12.

Tips: use emails only for private matters.

- Non-private matters: Blackboard discussion forum (details in video I.3).

- Grading matters: Ke Yuping, yuping.ke@connect.polyu.hk
- Other matters: Wang Shenghua, shenghua.wang@connect.polyu.hk

  cc yixin.cao@polyu.edu.hk

Prerequisite: COMP1011

- Two assignments: 30%
- Project: 3%
- Two quizzes: 12%
- Mid-term exam: 15%

- Final exam: 40%

No late submission will be entertained!

The project, although carrying a low eight,
is mandatory if you aim for A (A+, A-).

Quiz arrangements

Each quiz will have a large number of (mostly multiple-choice) questions.
Don't try to answer all of them: I myself cannot do it in the quiz time.
The purpose is to keep you busy so that you don't have time to "help" others.

For each quiz, there will be a grading number n (not announced before the quiz).
Suppose that you've c correct answers and w wrong answers, your score of the quiz is

$$\begin{cases} \frac{c}{c+w} * 100 & \text{if } c+w \geq n \\ \frac{c}{n} * 100 & \text{otherwise,} \end{cases}$$

and if $c > n$, then you get bonus $(c-n) * 0.5$, disregarding the number w.
For example, if $n = 20$, then the score is

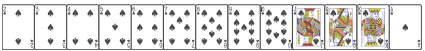
90 if $c = 40, w = 10$;

80 if $c = 20, w = 5$;

75 if $c = 15, w = 5$;

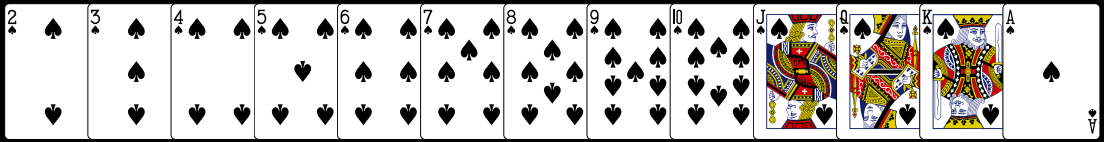
50 if $c = 10, w = 5$;

50 if $c = 10, w = 0$.



- Don’t read other’s codes for assignments.
- Don’t share your codes for assignments.

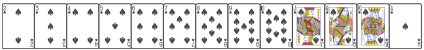
If two students submit similar codes for an assignment,
they'll be treated as plagiarism.

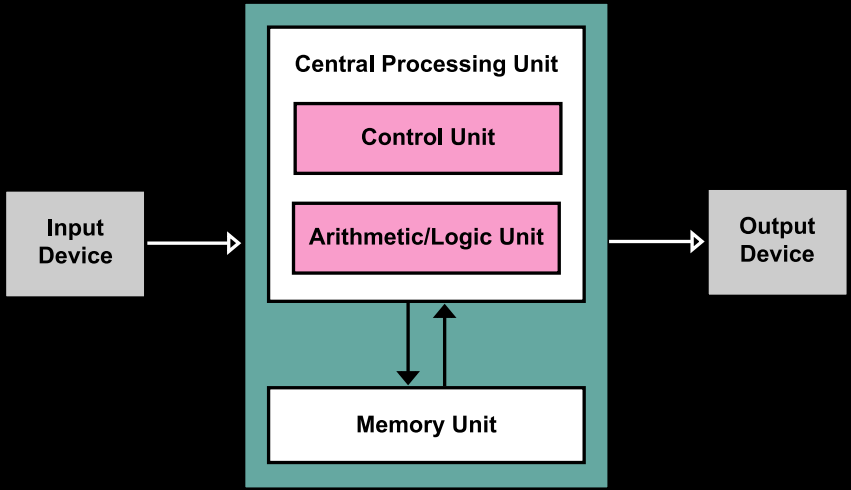


0.2 0.5 1 1.4

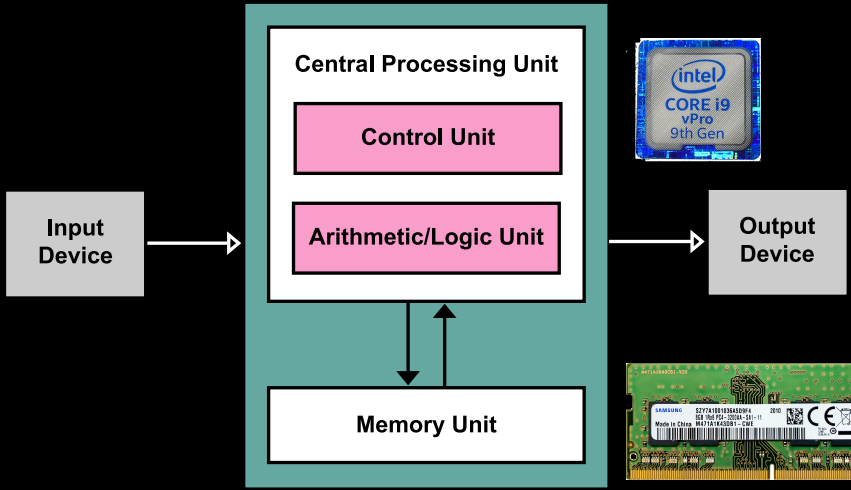
Bonus cards for good questions and answers.

Essentials of Computers

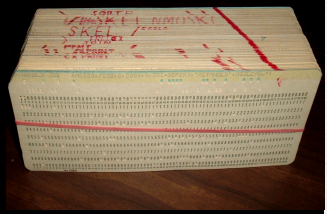


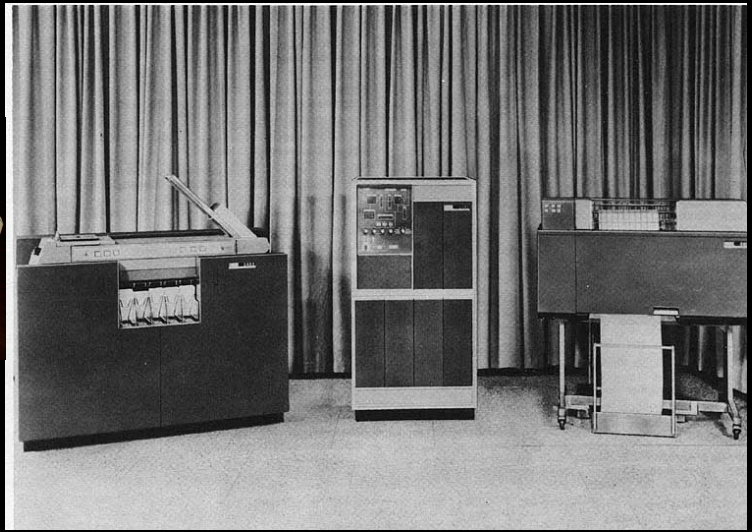


The von Neumann architecture ()



The von Neumann architecture (🌍)



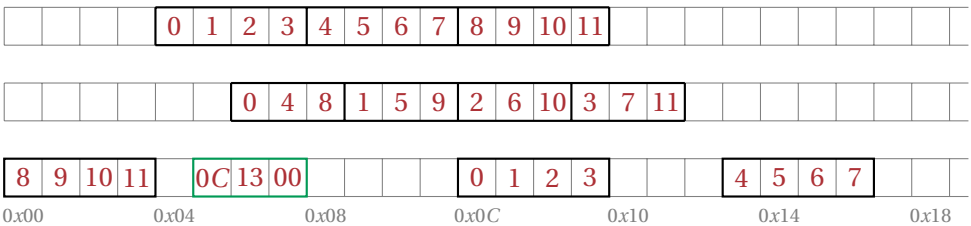


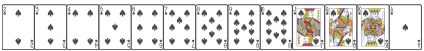
IBM 1401 MainframeIBM 1401 Mainframe

Two-dimensional arrays

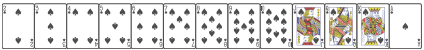
Now that the memory is a "tape," how to accommodate a two-dimensional array?

Three ways to store $\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix}$



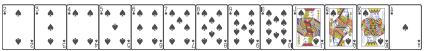


Key Messages



Physical addresses

Why street numbers are *usually* non-consecutive?



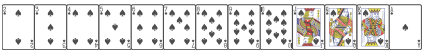
Physical addresses

Why street numbers are *usually* non-consecutive?

Hong Kong Government (3.1 of Practice Note):

There should be a reasonable balance
between the odd numbers on one side and
the even numbers on the other side of a street.

Message 1:
Be prepared for
change.



Which one is better?

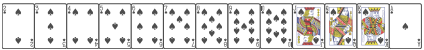
Which one is better?

You want to buy Pot rice or
Two dishes with rice?

Message 2:
How your data are organized
depends on your applications.

Message 3:
Data structures are everywhere.

Data Structures & Algorithms



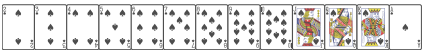
Data Structures and Algorithms

Data Structures and Algorithms
?

Data structures are

ways to organize data (information).
examples:

- simple variables—primitive types
- objects — collection of data items of various types
- arrays — collection of data items of the same type, stored contiguously
- linked lists — sequence of data items, each one pointing to the next one
- stack, queue, tree, table, graph...



An algorithm is (like) a recipe

Input → Procedure → Output

Ingredients

- $1\frac{1}{2}$ cups all-purpose flour
- $3\frac{1}{2}$ teaspoons baking powder
- 1 teaspoon salt
- 1 tablespoon white sugar
- $1\frac{1}{4}$ cups milk
- 1 egg
- 3 tablespoons butter, melted

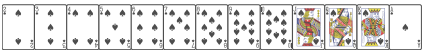
Directions

1. In a large bowl, sift together the flour, baking powder, salt and sugar.
   Make a well in the center and pour in the milk, egg and melted butter; mix until smooth.
2. Heat a frying pan over medium high heat.
   Pour or scoop the batter onto the pan, using approximately $\frac{1}{4}$ cup for each pancake.
   Brown on both sides and serve hot.

Source: allrecipes.com (demo)

Problem solving with a computer

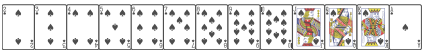
1. model the problem as a computational problem,
2. design an algorithm,
3. implement the algorithm as a program.



Problem solving with a computer

1. model the problem as a computational problem,
2. design an algorithm,
3. implement the algorithm as a program.

you need to choose data structures

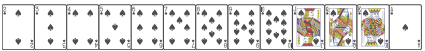


Problem solving with a computer

1. model the problem as a computational problem,
2. design an algorithm,
3. implement the algorithm as a program.

you need to choose data structures

you may have to revise data structures

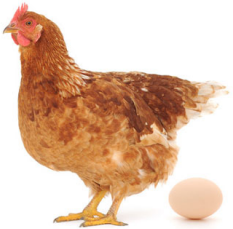


Problem solving with a computer

1. model the problem as a computational problem,
2. design an algorithm,
3. implement the algorithm as a program.

you need to choose data structures

you may have to revise data structures



source:woodsideeggs.com

- Data structures are about the organization of data in a computer's memory.

  You have seen arrays, queues, stacks, and linked lists in COMP1011.
- Most (maybe all) data structures are motivated by physical objects.

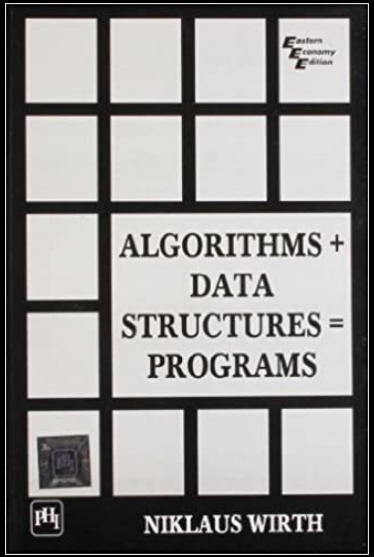
  What's different between a physical queue and a queue data structure?
- The choice of data structures has huge impact on efficiency (will see examples).

- An algorithm is a procedure for carrying out a particular task.
- It works by apply operations on data structures.

I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships.

Linus Torvalds
(The inventor of Linux and Git)

A book by Niklaus Wirth
(Turing Award winner 1984)



Plain words

Repeatedly find the minimum element from the unsorted part and put it at the beginning.

Pseudocode

1. **for** $i \leftarrow 1..n\text{–}1$ **do**
2. $\quad k \leftarrow i$;
3. $\quad$ **for** $j \leftarrow i+1..n\text{–}1$ **do**
4. $\quad\quad$ **if** $A[k] > A[j]$ **then** $k \leftarrow j$;
5. $\quad$ swap $A[i]$ and $A[k]$.

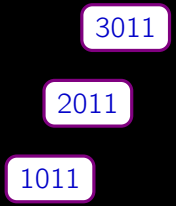
(Java) code

```
void selection(int[] a) {
  int n = a.length;
  int min;
  for (int i=0; i<n-1; i++) {
    min = i;
    for (int j=i+1; j<n; j++)
      if (a[min] > a[j])
        min = j;
    swap(a, min, i);
  }
}
```

Three ways of expressing algorithms.

A common mistake my students make is to use pseudo-code to dress up an ill-defined idea so that it looks more formal.

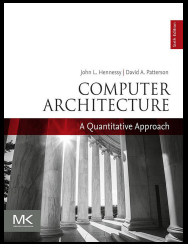
Steven Skiena
The Algorithm Design Manual (Page 12)

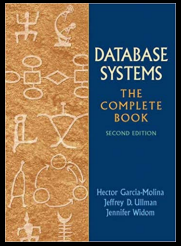

3011
2011
1011

3011

2011

1011

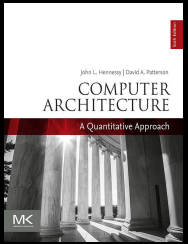


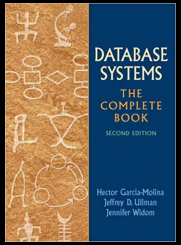


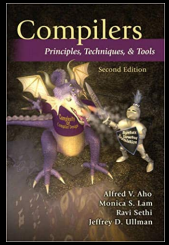
3011

2011

1011



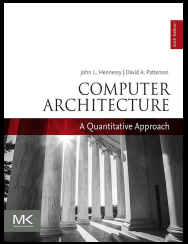


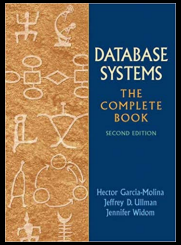


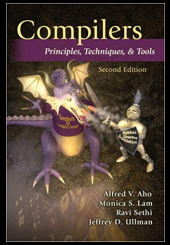
3011

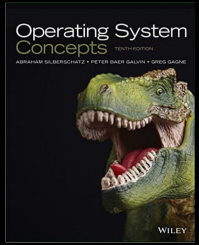
2011

1011





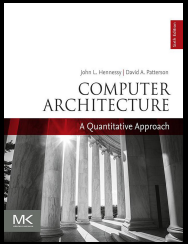


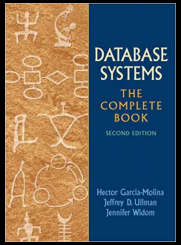


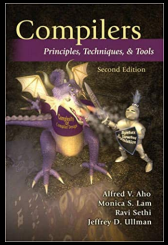
3011

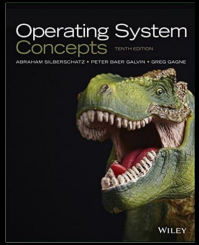
2011

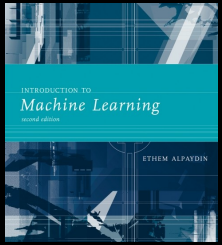
1011







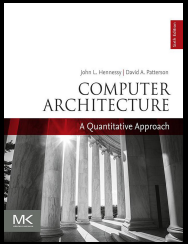


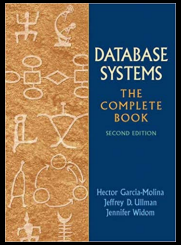


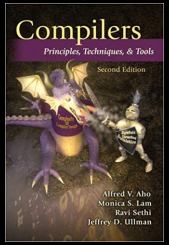
3011

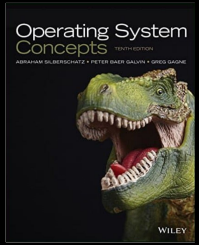
2011

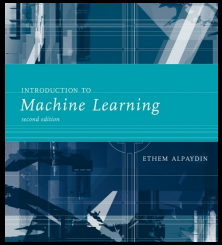
1011

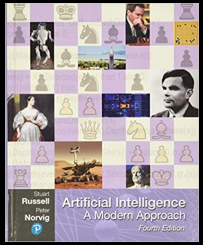








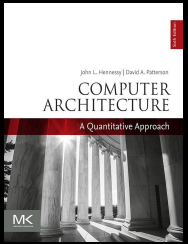




3011

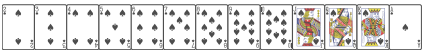
2011

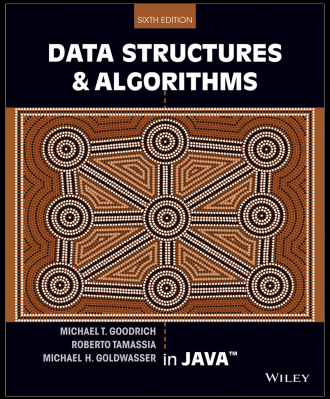
1011

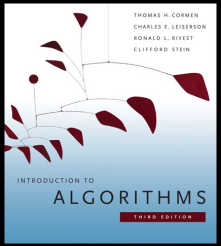


Data structures are easy:
All materials are (naively) natural. They are discovered instead of invented.

Data structures are difficult:
Knowing the concepts is not enough; you have to practice to master them.

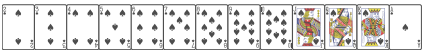






Too difficult for 2011,
but important for a deep understanding.

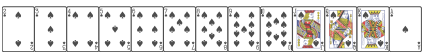
Java



Courtesy http://g8906011.pixnet.net

```
int find(int key, int low, int high) {
    int mid = (low + high) / 2;
    if (a[mid] == key) return mid;
    if (low == high) return -1;
    if (a[mid] < key) return find(key, mid, high);
    return find(key, low, mid);
}

int find(int key) {
    if (length == 0) return -1;
    return find(key, 0, length - 1);
}
```

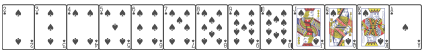


Arrays in Java

```
int[] a = {11, 16, 9, 4, 12, 30, 3};
```

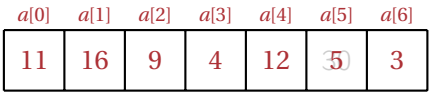
what if `x = a[2];`
and `a[5] = 5;`

Another way to define an array in Java: `int[] a = new int[7];`



Arrays in Java

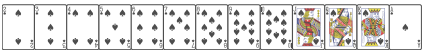
```
int[] a = {11, 16, 9, 4, 12, 30, 3};
```



$x = 9$

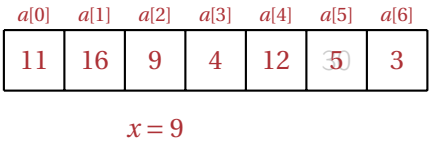
what if `x = a[2];`
and `a[5] = 5;`

Another way to define an array in Java: `int[] a = new int[7];`



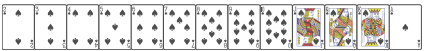
Arrays in Java

```
int[] a = {11, 16, 9, 4, 12, 30, 3};
```

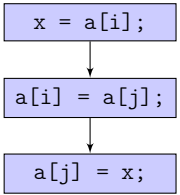


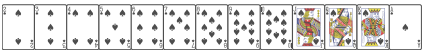
what if `x = a[2];`
and `a[5] = 5;`

Another way to define an array in Java: `int[] a = new int[7];`

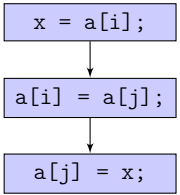


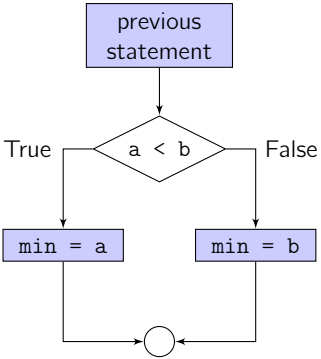
Basic control flow

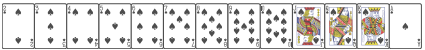




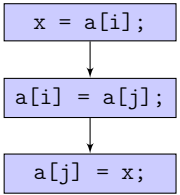
Basic control flow

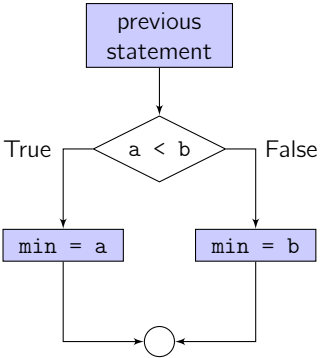




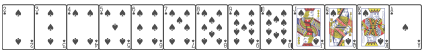


Basic control flow

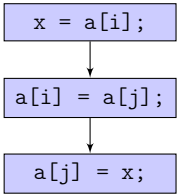


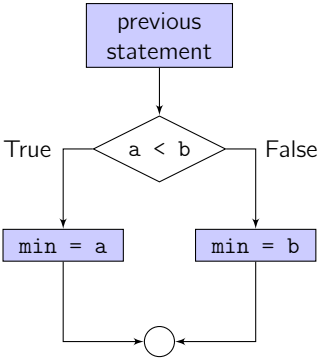


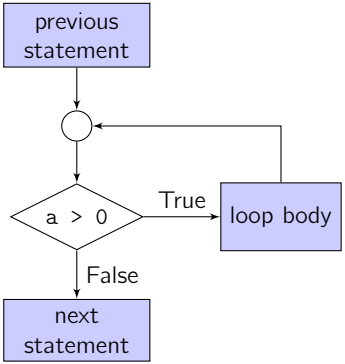

```
min = a < b? a : b;
```



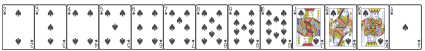
Basic control flow



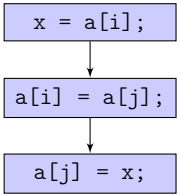


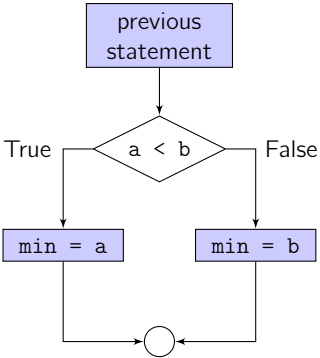


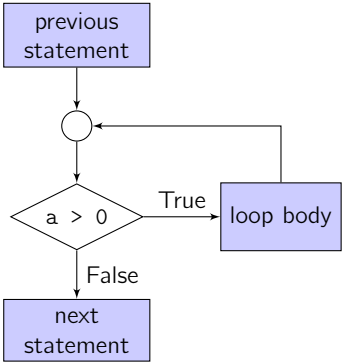

```
min = a < b? a : b;
```



Basic control flow

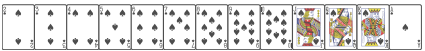







```
min = a < b? a : b;
```

```
while (a > 0) {
    ...
    a--;
}
```



Control flow II: subroutine call

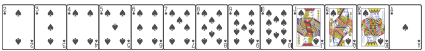
return value type

method name

list of formal parameters

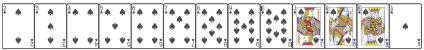
```
int find (int key, int low, int high) {
    int mid = (low + high) / 2;
    if (a[mid] == key) return mid;
    if (low == high) return -1;
    if (a[mid] < key) return find(key, mid, high);
    return find(key, low, mid);
}
```

local variables declaration



Java has provided all the basic structures,
as well as all the algorithms we'll discuss during this course.

My implementations (mostly stupid) are for demonstration purpose only.



Eclipse

IntelliJ IDEA

NetBeans

Important links

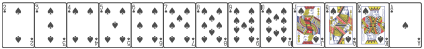
(Almost) everything about java can be found at:
Java SE 16[1] API Specifications (for download)

Java collections (data structures) documents:

- Collections Framework
- Collections Framework Tutorial

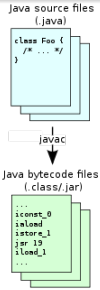
[1] Java 17 will be released in September 2021: replace 16 by 17 in all the links.

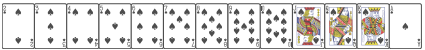
How Java works (the very basic)



How Java works (the very basic)

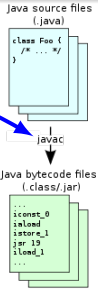
A Java source file (`.java`) is compiled (using `javac`), which generates the Java bytecode file (`.class`).

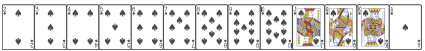




How Java works (the very basic)

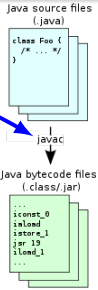
A Java source file (`.java`) is compiled (using `javac`), which generates the Java bytecode file (`.class`).



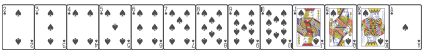


How Java works (the very basic)

A Java source file (`.java`) is compiled (using `javac`), which generates the Java bytecode file (`.class`).



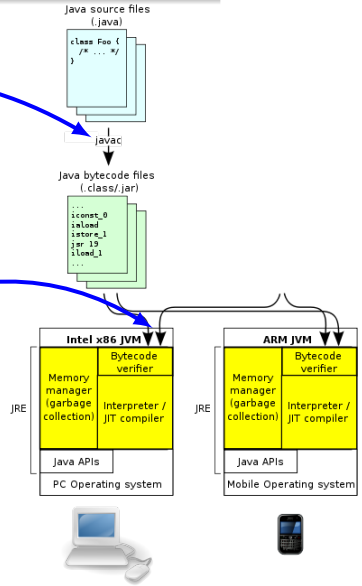
A Java bytecode file (`.class`) can be executed on any JVM, using `java`.



How Java works (the very basic)

A Java source file (`.java`) is compiled (using `javac`), which generates the Java bytecode file (`.class`).

A Java bytecode file (`.class`) can be executed on any JVM, using `java`.

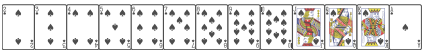


Course topics

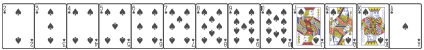
- vocabulary for design and analysis of algorithms
  - Dry and boring, but that is what defines computing, and defines us.
  - Everybody in PolyU can program; everybody in FENG knonws machine learning.
  - The difference between CS and others lie in data structures and algorithms.
- use and implementation of data structures
- design paradigm (divide and conquer)
- how to sort and how to not sort.

expected outcomes

- improve your programming
- sharpen your mathematical and analytical skills
- start “thinking algorithmically”
- prepare your technical interviews.



Basic Sorting Algorithms



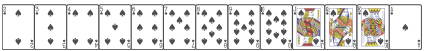
Basic sorting algorithms

Three basic sorting algorithms

- bubble sort
- insertion sort
- selection sort

Do you still remember how to write them?

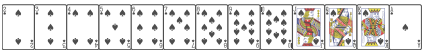
Which one is the best, the worst?



Three basic sorting algorithms

- bubble sort
- selection sort
- insertion sort

Which one is the best, the worst?

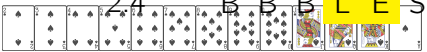


1.1	B	U	B	B	L	E	S	O	R	T
1.2	B	U	B	B	L	E	S	O	R	T
1.3	B	B	U	B	L	E	S	O	R	T
1.4	B	B	B	U	L	E	S	O	R	T
1.5	B	B	B	L	U	E	S	O	R	T
1.6	B	B	B	L	E	U	S	O	R	T
1.7	B	B	B	L	E	S	U	O	R	T
1.8	B	B	B	L	E	S	O	U	R	T
1.9	B	B	B	L	E	S	O	R	U	T
1.10	B	B	B	L	E	S	O	R	T	U

After the first iteration, the last element is the maximum. Why?

2.1	B	B	B	L	E	S	O	R	T	U
2.2	B	B	B	L	E	S	O	R	T	U
2.3	B	B	B	L	E	S	O	R	T	U
2.4	B	B	B	L	E	S	O	R	T	U

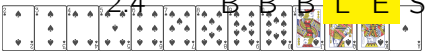
2nd iteration needs to start from the first pair, because they may not be the same pair.



1.1	B	U	B	B	L	E	S	O	R	T
1.2	B	U	B	B	L	E	S	O	R	T
1.3	B	B	U	B	L	E	S	O	R	T
1.4	B	B	B	U	L	E	S	O	R	T
1.5	B	B	B	L	U	E	S	O	R	T
1.6	B	B	B	L	E	U	S	O	R	T
1.7	B	B	B	L	E	S	U	O	R	T
1.8	B	B	B	L	E	S	O	U	R	T
1.9	B	B	B	L	E	S	O	R	U	T
1.10	B	B	B	L	E	S	O	R	T	U

2.1	B	B	B	L	E	S	O	R	T	U
2.2	B	B	B	L	E	S	O	R	T	U
2.3	B	B	B	L	E	S	O	R	T	U
2.4	B	B	B	L	E	S	O	R	T	U

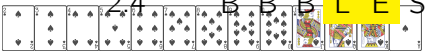
To save time, we can stop after an iteration when no swap happens.



1.1	B	U	B	B	L	E	S	O	R	T
1.2	B	U	B	B	L	E	S	O	R	T
1.3	B	B	U	B	L	E	S	O	R	T
1.4	B	B	B	U	L	E	S	O	R	T
1.5	B	B	B	L	U	E	S	O	R	T
1.6	B	B	B	L	E	U	S	O	R	T
1.7	B	B	B	L	E	S	U	O	R	T
1.8	B	B	B	L	E	S	O	U	R	T
1.9	B	B	B	L	E	S	O	R	U	T
1.10	B	B	B	L	E	S	O	R	T	U

2.1	B	B	B	L	E	S	O	R	T	U
2.2	B	B	B	L	E	S	O	R	T	U
2.3	B	B	B	L	E	S	O	R	T	U
2.4	B	B	B	L	E	S	O	R	T	U

What's the array after 2nd iteration?
Is there any swap in 3rd iteration?



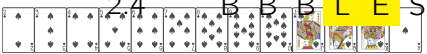
1.1	B	U	B	B	L	E	S	O	R	T
1.2	B	U	B	B	L	E	S	O	R	T
1.3	B	B	U	B	L	E	S	O	R	T
1.4	B	B	B	U	L	E	S	O	R	T
1.5	B	B	B	L	U	E	S	O	R	T
1.6	B	B	B	L	E	U	S	O	R	T
1.7	B	B	B	L	E	S	U	O	R	T
1.8	B	B	B	L	E	S	O	U	R	T
1.9	B	B	B	L	E	S	O	R	U	T
1.10	B	B	B	L	E	S	O	R	T	U
2.1	B	B	B	L	E	S	O	R	T	U
2.2	B	B	B	L	E	S	O	R	T	U
2.3	B	B	B	L	E	S	O	R	T	U
2.4	B	B	B	L	E	S	O	R	T	U

initially flag = false

flag = true

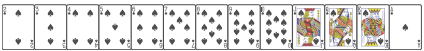
reset flag = false

flag = true



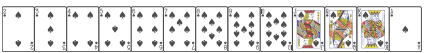
1	I	N	S	E	R	T	I	O	N	S	O	R	T
2	I	N	S	E	R	T	I	O	N	S	O	R	T
3	I	N	S	E	R	T	I	O	N	S	O	R	T
4	E	I	N	S	R	T	I	O	N	S	O	R	T
5	E	I	N	R	S	T	I	O	N	S	O	R	T
6	E	I	N	R	S	T	I	O	N	S	O	R	T
7	E	I	I	N	R	S	T	O	N	S	O	R	T
8	E	I	I	N	O	R	S	T	N	S	O	R	T
9	E	I	I	N	N	O	R	S	T	S	O	R	T
10	E	I	I	N	N	O	R	S	S	T	O	R	T
11	E	I	I	N	N	O	R	S	S	T	O	R	T
12	E	I	I	N	N	O	O	R	R	S	S	T	T
13	E	I	I	N	N	O	O	R	R	S	S	T	T

It maintains a sorted part, which is extended after each iteration.

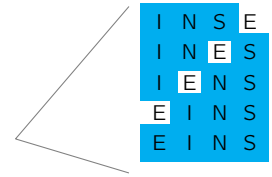


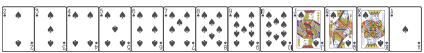
1	I	N	S	E	R	T	I	O	N	S	O	R	T
2	I	N	S	E	R	T	I	O	N	S	O	R	T
3	I	N	S	E	R	T	I	O	N	S	O	R	T
4	E	I	N	S	R	T	I	O	N	S	O	R	T
5	E	I	N	R	S	T	I	O	N	S	O	R	T
6	E	I	N	R	S	T	I	O	N	S	O	R	T
7	E	I	I	N	R	S	T	O	N	S	O	R	T
8	E	I	I	N	O	R	S	T	N	S	O	R	T
9	E	I	I	N	N	O	R	S	T	S	O	R	T
10	E	I	I	N	N	O	R	S	S	T	O	R	T
11	E	I	I	N	N	O	R	S	S	T	O	R	T
12	E	I	I	N	N	O	O	R	R	S	S	T	T
13	E	I	I	N	N	O	O	R	R	S	S	T	T

I	N	S	E
I	N	E	S
I	E	N	S
E	I	N	S
E	I	N	S

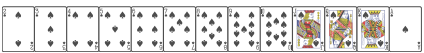


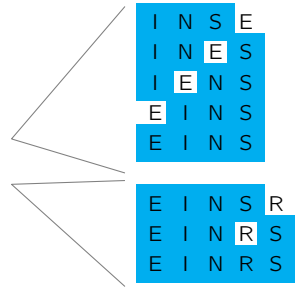
1	I	N	S	E	R	T	I	O	N	S	O	R	T
2	I	N	S	E	R	T	I	O	N	S	O	R	T
3	I	N	S	E	R	T	I	O	N	S	O	R	T
4	E	I	N	S	R	T	I	O	N	S	O	R	T
5	E	I	N	R	S	T	I	O	N	S	O	R	T
6	E	I	N	R	S	T	I	O	N	S	O	R	T
7	E	I	I	N	R	S	T	O	N	S	O	R	T
8	E	I	I	N	O	R	S	T	N	S	O	R	T
9	E	I	I	N	N	O	R	S	T	S	O	R	T
10	E	I	I	N	N	O	R	S	S	T	O	R	T
11	E	I	I	N	N	O	R	S	S	T	O	R	T
12	E	I	I	N	N	O	O	R	R	S	S	T	T
13	E	I	I	N	N	O	O	R	R	S	S	T	T



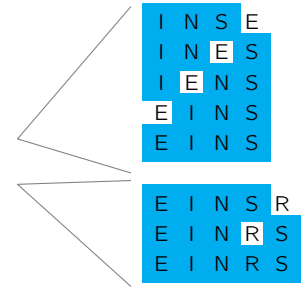


1	I	N	S	E	R	T	I	O	N	S	O	R	T
2	I	N	S	E	R	T	I	O	N	S	O	R	T
3	I	N	S	E	R	T	I	O	N	S	O	R	T
4	E	I	N	S	R	T	I	O	N	S	O	R	T
5	E	I	N	R	S	T	I	O	N	S	O	R	T
6	E	I	N	R	S	T	I	O	N	S	O	R	T
7	E	I	I	N	R	S	T	O	N	S	O	R	T
8	E	I	I	N	O	R	S	T	N	S	O	R	T
9	E	I	I	N	N	O	R	S	T	S	O	R	T
10	E	I	I	N	N	O	R	S	S	T	O	R	T
11	E	I	I	N	N	O	R	S	S	T	O	R	T
12	E	I	I	N	N	O	O	R	R	S	S	T	T
13	E	I	I	N	N	O	O	R	R	S	S	T	T

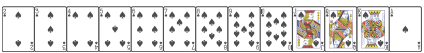




1	I	N	S	E	R	T	I	O	N	S	O	R	T
2	I	N	S	E	R	T	I	O	N	S	O	R	T
3	I	N	S	E	R	T	I	O	N	S	O	R	T
4	E	I	N	S	R	T	I	O	N	S	O	R	T
5	E	I	N	R	S	T	I	O	N	S	O	R	T
6	E	I	N	R	S	T	I	O	N	S	O	R	T
7	E	I	I	N	R	S	T	O	N	S	O	R	T
8	E	I	I	N	O	R	S	T	N	S	O	R	T
9	E	I	I	N	N	O	R	S	T	S	O	R	T
10	E	I	I	N	N	O	R	S	S	T	O	R	T
11	E	I	I	N	N	O	R	S	S	T	O	R	T
12	E	I	I	N	N	O	O	R	R	S	S	T	T
13	E	I	I	N	N	O	O	R	R	S	S	T	T



The last iteration cannot be omitted: it might be I.



1	S	E	L	E	C	T	I	O	N	S	O	R	T
2	C	E	L	E	S	T	I	O	N	S	O	R	T
3	C	E	L	E	S	T	I	O	N	S	O	R	T
4	C	E	E	L	S	T	I	O	N	S	O	R	T
5	C	E	E	I	S	T	L	O	N	S	O	R	T
6	C	E	E	I	L	T	S	O	N	S	O	R	T
7	C	E	E	I	L	N	S	O	T	S	O	R	T
8	C	E	E	I	L	N	O	S	T	S	O	R	T
9	C	E	E	I	L	N	O	O	T	S	S	R	T
10	C	E	E	I	L	N	O	O	R	S	S	T	T
11	C	E	E	I	L	N	O	O	R	S	S	T	T
12	C	E	E	I	L	N	O	O	R	S	S	T	T
13	C	E	E	I	L	N	O	O	R	S	S	T	T
14	C	E	E	I	L	N	O	O	R	S	S	T	T

```
void bubble(int[] a) {
  int i, j, n = a.length;
  for (i=1; i<n; i++)
    for (j=0; j<n-i; j++)
      if (a[j+1] < a[j])
        swap(a, j, j+1);
}
```

```
void insertion(int[] a) {
  int i, j, key, n = a.length;
  for (i = 1; i < n; i++) {
    key = a[i];
    for (j = i - 1; j >= 0; j--) {
      if (a[j] <= key) break;
      a[j + 1] = a[j];
    }
    a[j + 1] = key;
  }
}
```

```
void selection(int[] a) {
  int n = a.length;
  int min;
  for (int i=0; i<n-1; i++) {
    min = i;
    for (int j=i+1; j<n; j++)
      if (a[min] > a[j]) min = j;
    swap(a, min, i);
  }
}
```

I compress the codes to fit into one slide,
please do not do that in your programming.

Week 2: Analysis of Algorithms