Algorithms and Data Structures

DG200 team (contribution Loe Feijs)

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Content

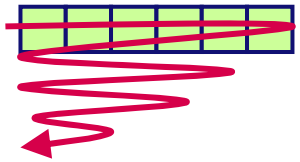
- General principles
- Searching
- Sorting
- Optimising (more algorithms and data structures)

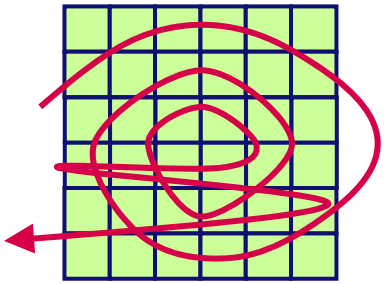
General principles

- Data structures
  - built from variables, arrays, objects refering to each other
- Algorithms
  - built from if, if-else, for, while, functions calling each other
- To be designed together
  - like the railway topology and the train schedule
  - you cannot design an efficient train schedule
  - unless you know the railway topology

General principles (continued)

- Searching
  - one-dimensional problems
  - with single-loop solutions
- Sorting
  - one-dimensional problems
  - with nested-loop solutions
  - useful to prepare for easy searching
- Optimising
  - two-dimensional problems
  - with complex nested-loop solutions
  - good example: shortest path problems
  - useful in route planners and internet packet routing





General principles (continued)

- Issues
  - correctness
  - efficiency
- Correctness
  - respecting array bounds
  - not using null pointers
  - initialising all variables
  - careful reasoning
- Efficiency
  - data structures for memoization
  - $\mathcal{O}(n)$ is better than $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$
  - sometimes one has to accept $\mathcal{O}(e^n)$
  - use existing classics

Searching

- Basic idea
  - data in one array
  - scan the entire array with one for-loop
- Basic code

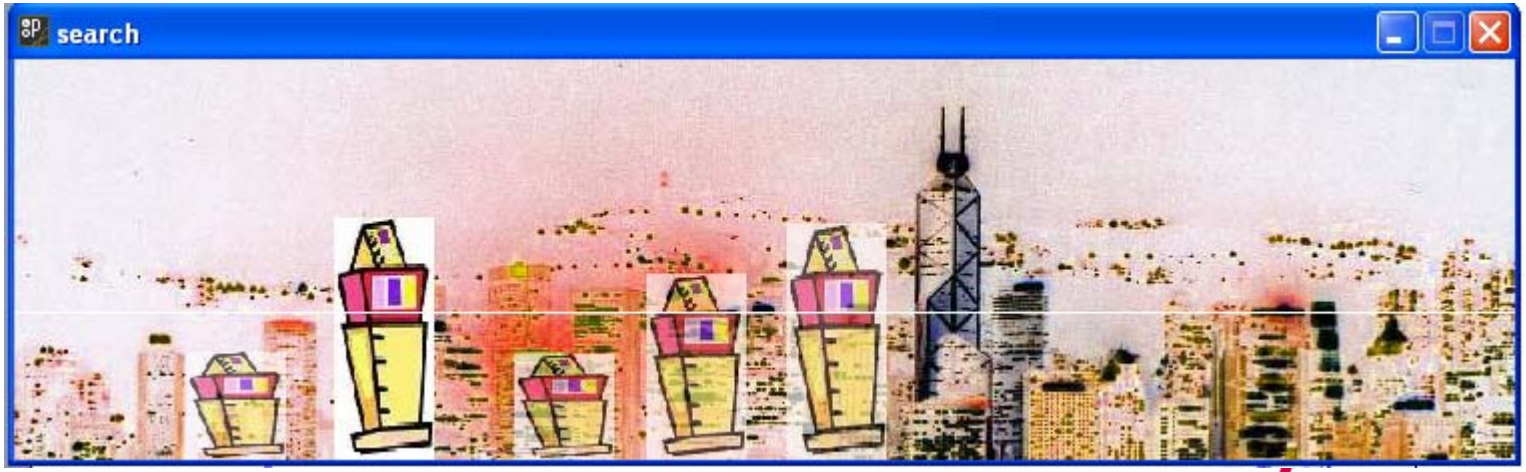
```
int[] a=new int[N];
// a is filled here
for (int i=0; i<N; i++) {
    if  (a[i] == x) {
        // aha, x found
    }
}
```

Searching

- Tricks of the trade
  - for an array of N elements
  - the indexes run from 0 to N-1
  - use a local loop counter named i, j or k
  - use a boolean found to stop once your x is found
  - use the pseudo value infinity to begin a min or max loop

Searching

- Demo application
  - edit an array of towers
  - random height levels
  - calculate average
  - find the tallest



Searching

- Interface of the demo

The construction truck is at horizontal position x

```
void keyPressed() {
  if(keyCode == LEFT)  { x=x-2; }
  if(keyCode == RIGHT) { x=x+2; }
  if(key == ' ') { // add tower
      if (towers<maxtowers){
      loc[towers]=x;
      lvl[towers]=floor(random(0,200));
      towers++;
    }
  }
  if(keyCode == ENTER) { done=true;}
}
```

For each tower index there is a location loc and a height level lvl

Searching

- Action of the demo

```
// in draw
if (done) {
      int i=tallest();
      tint(255,255);
      image(tower,loc[i],200-lvl[i],50,lvl[i]);
      int a=average();
      stroke(255);
      line(0, 200-a, 750, 200-a);
}
```

Finding a maximum

Calculating an average

Searching

- Finding a maximum

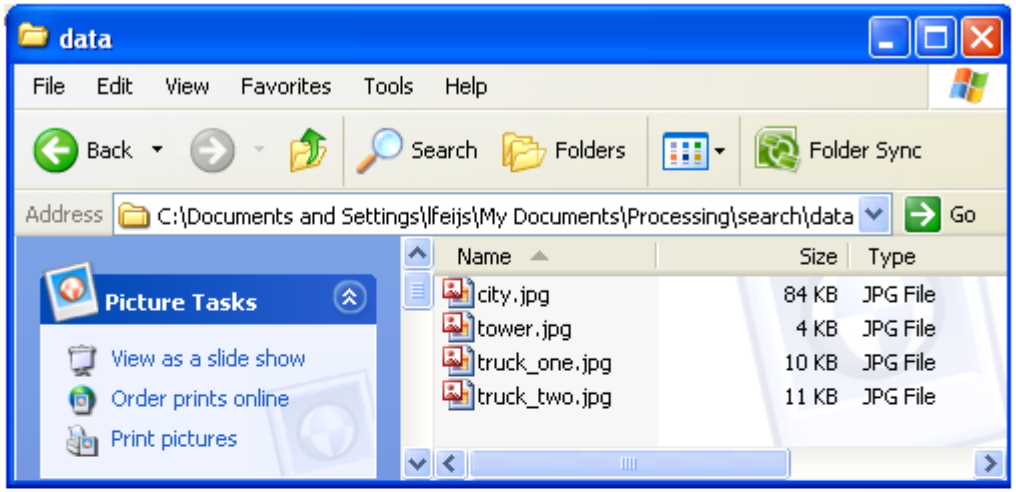
```
int tallest() {
    int m=-1; // max level found so far
    int i=-1; // index where it's found
    for (int j=0; j<towers; j++)
        if (lvl[j] > m) {
           m = lvl[j];
           i = j;
        }
    return i;
}
```

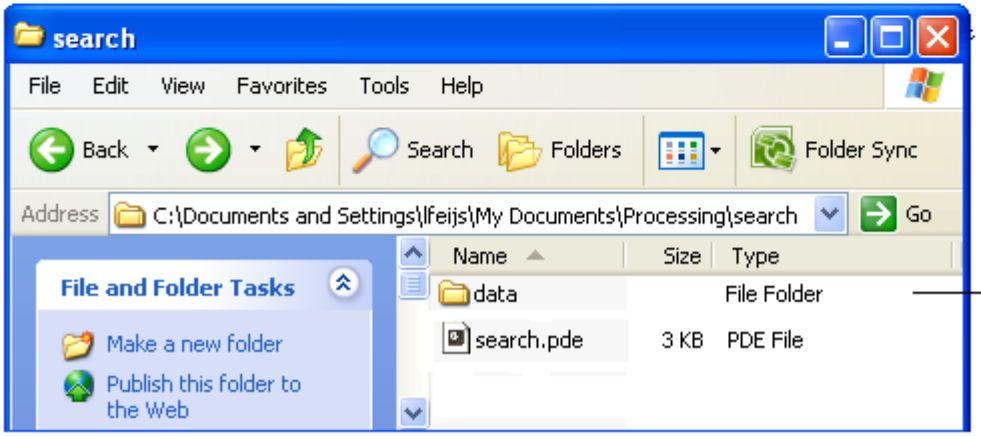
Searching

- Calculating an average

```
int average() {
    int s=0; // sum calculated so far
    for (int j=0; j<towers; j++)
        s = s+lvl[j];
    if (towers>0)
        return s / towers;
        else return 1;
}
```

Searching (more about the demo)





Exercises on searching

- Do it yourself
  - find the lowest tower
- Food for thought
  - can you find the median?

Sorting

- Demo application
  - edit an array of towers
  - three tower types
  - towers as objects
  - sort by level

Sorting

edit

sort

Technische Universiteit
Eindhoven
University of Technology

Sorting

- Selection sort
  - noted for its simplicity
  - see e.g. (Dutch) wikipedia, selection sort
  - more: heapsort, insertion sort, bubble sort, quicksort, Shellsort, …

```
// invoer = array met integers
for (int i = 0; i < array.length; i++) {

        // zoek kleinste in de rest van de array
        int minIndex = i;
        for (int j = i; j < array.length; j++) {
                if (array[j] < array[minIndex]) {
                        minIndex = j;
                }
        }

        // verwissel waarden
        int temp = array[i];
        array[i] = array[minIndex];
        array[minIndex] = temp;
}
```

Sorting

```
void selectionSort(){
     for (int i = 0; i < towers; i++) {
      // when here: i smallest values are sorted in 0..i-1
      int mi = i;
      for (int j = i; j < towers; j++) {
          // when here: mi is index of smallest in i..j-1
          if (twr[j].lvl < twr[mi].lvl)
              mi = j;
      }
      // swap towers at indexes i and mi
      Tower tmp = twr[i]; twr[i] = twr[mi]; twr[mi] = tmp;
   }
}
```

Sorting (oo)

- Object orientation
  - instead of 3 arrays of single tower-attributes

```
int[] loc = new int[maxtowers]; // location, i.e. x value
int[] lvl = new int[maxtowers]; // level, i.e. height
int[] typ = new int[maxtowers]; // type, either 0,1, or
```

  - define one array of tower objects with 3 attributes each

```
class Tower {
  int loc; // location, i.e. x value
  int lvl; // level, i.e. height
  int typ; // type, either 0,1, or 2
  // etc.
}

Tower[] twr = new Tower[maxtowers];
```

Exercises on sorting

- Study and run the sorting demo
- Do it yourself
  - bubble sort
  - e.g. read wikipedia: bubble sort sort
- Food for thought
  - now can you find the median?