COMP20230: Data Structures & Algorithms
Lecture 2: Complexity Analysis

Dr Andrew Hines

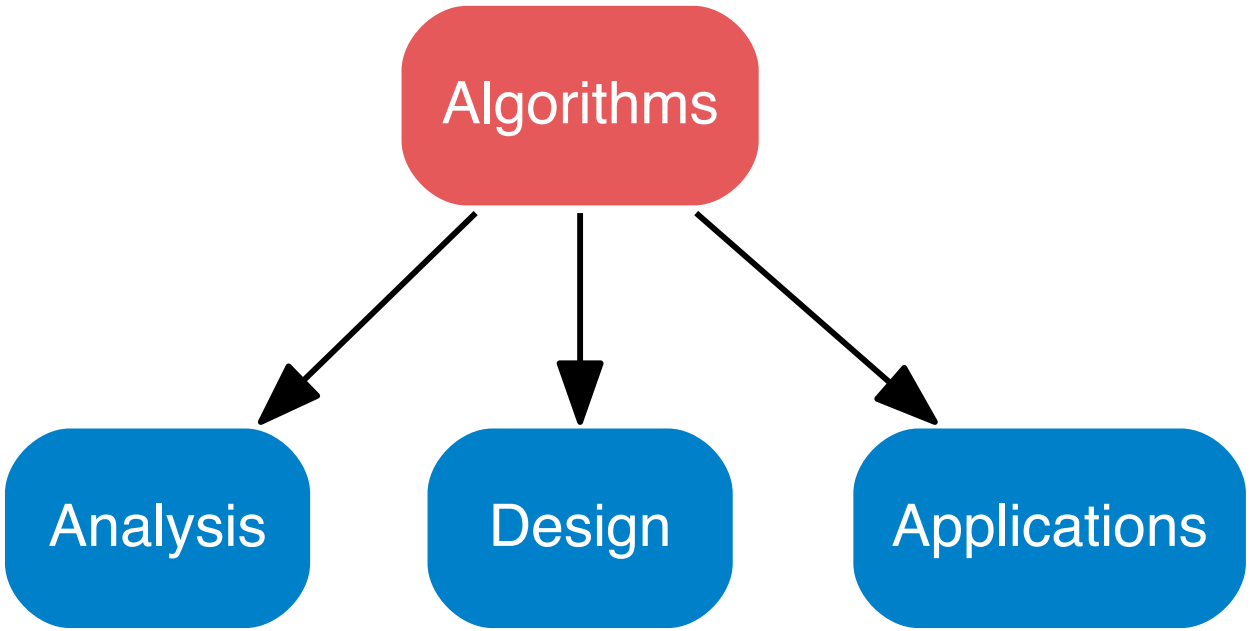
Office: E3.13 Science East
School of Computer Science
University College Dublin

andrew.hines@ucd.ie

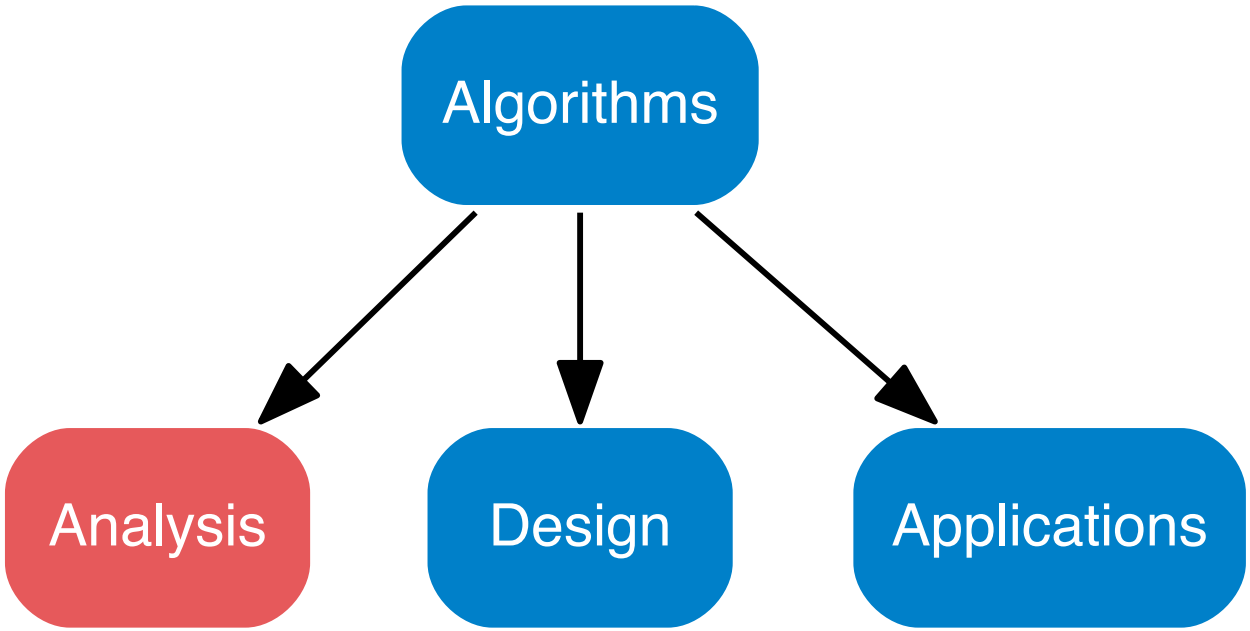
Algorithms

Recap: Lecture 1 covered

Housekeeping: Assignments, assessment, tutorials, moodle
Module: Learning outcomes, syllabus, course text, schedule
Moodle: Handout, Overleaf Latex Template, lecture notes groups



Analysis

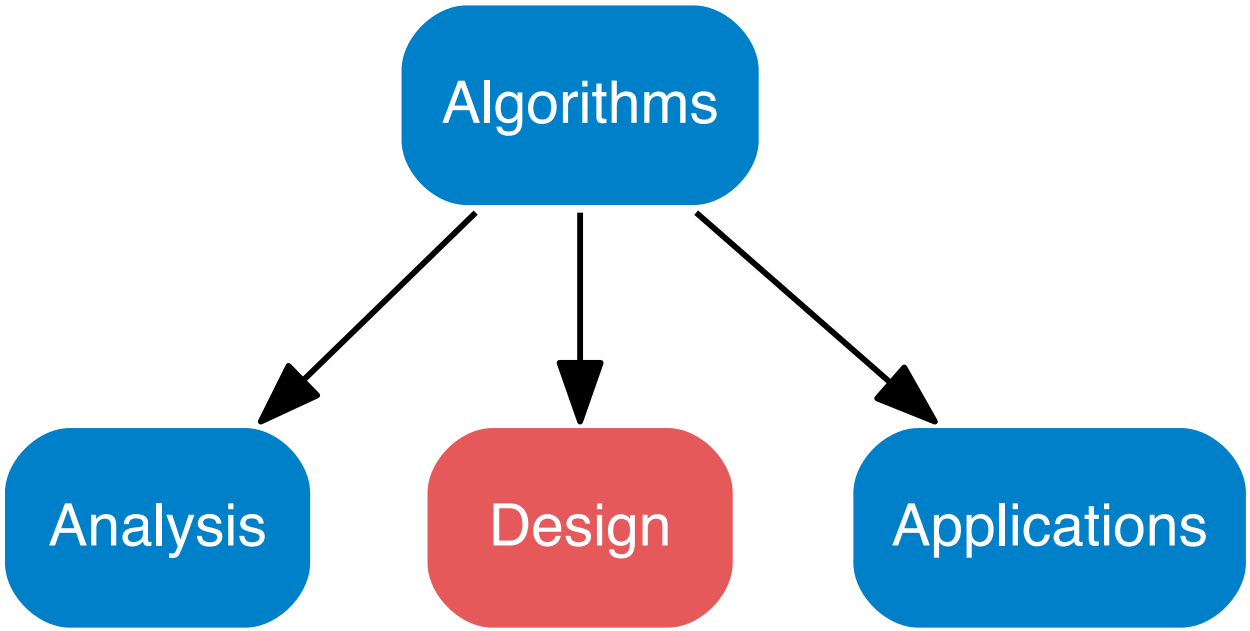


Attributes of a good algorithm

- **Correctness**
- **Speed**
- **Efficiency**
- Security
- Robustness
- Clarity
- Maintainability

"Does exactly what it says on the tin"

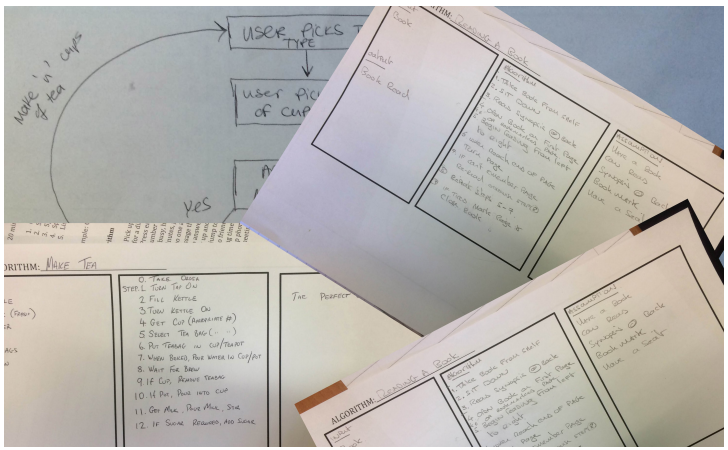
Design



Algorithm Design: Algorithm for an everyday task

20 Minute Goal

1. List any **assumptions**
2. Specify:
   1. **Inputs** (recipe ingredients)
   2. **Method** (recipe steps)
   3. **Outputs** (end result)



Before you begin...

Example: Making a telephone call

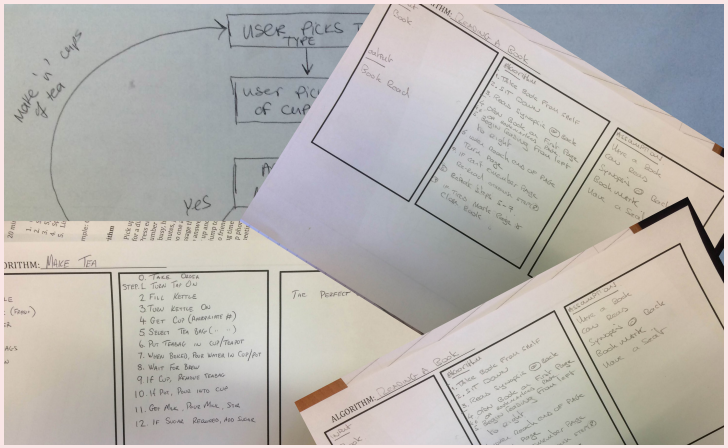
Everyday Tasks

- **Make tea**: multiple cups? preferences?
- **Brushing teeth**: how to make sure they were all equally cleaned?
- **Read a book**: how to pick up where you left off?
- **Pumping bike tyres**: how to check pressure is high enough?

Output: 1 page per group

Group Work Reflection

Review of Group Algorithms – (Photo Sheet Presentations)

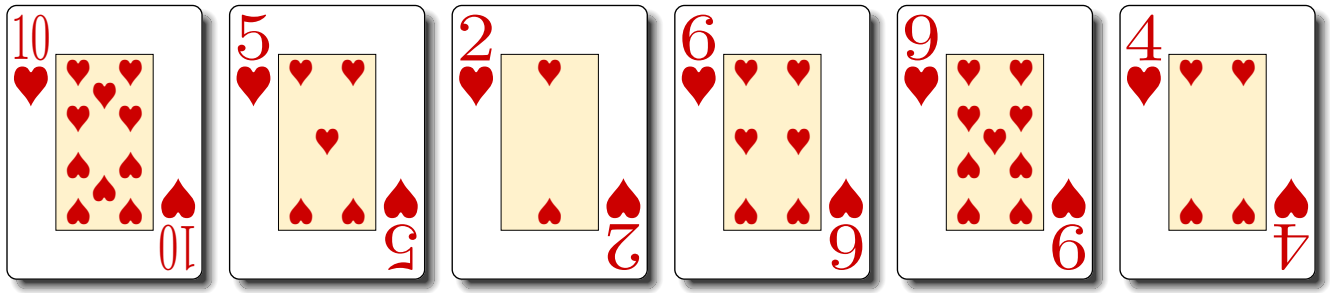


Individual Questions

- Were the algorithms precise?
- Which attributes of a good algorithm do they exhibit?
- Were they different?
- What might you do differently next time you try to write a program?

Algorithm: Simple Card Sort

1. Get a hand of unsorted cards (same suit)



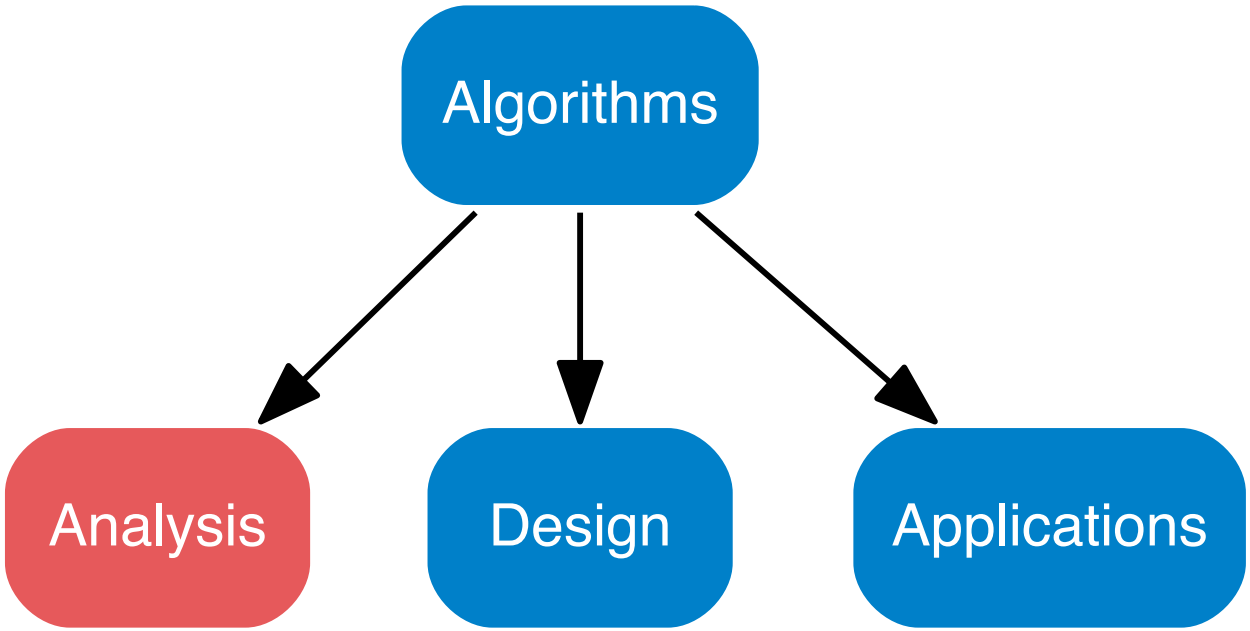
2. Repeat steps 3 through 5 until the unsorted hand is empty
3. Compare all unsorted cards
4. Select the smallest unsorted card (first is)
5. Move this card to the sorted hand
   Loop 1:
   Loop 2:
   Loop 3:
   Loop 4:
   Loop 5:
   Loop 6:
6. Stop

Pseudocode: Simple Card Sort

How do we convert this into pseudo code?

1. Try to write the pseudocode
2. Review the pseudocode tutorial handout on moodle
3. Did your group algorithm exercise follow pseudocode best practices?
4. Can you rewrite it so it does?

Analysis



Attributes of a good algorithm

- **Correctness**
- **Speed**
- **Efficiency**
- Security
- Robustness
- Clarity
- Maintainability

"Does exactly what it says on the tin"

Let's talk about another example algorithm

Question

Why are (were?) telephone directories are sorted alphabetically by last name.

Let's talk about another example algorithm

Question

Why are (were?) telephone directories are sorted alphabetically by last name.

Answer

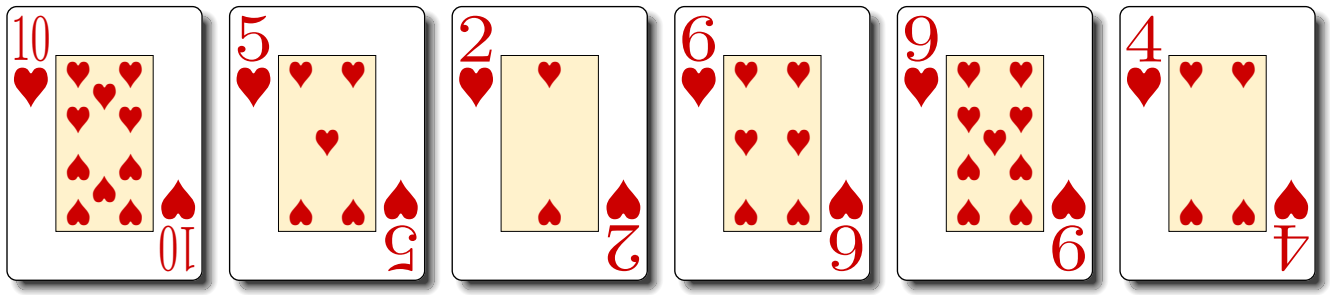
A sorted index can be searched quickly.

Sorting Algorithms

From: `http://www.sorting-algorithms.com`
Animated:
`https://www.toptal.com/developers/sorting-algorithms`

Bubble Sort Algorithm

1. Get a hand of unsorted cards (same suit)



2. Repeat steps 3 through 5 until nothing happens
3. For every couple/pair of neighbouring cards (left-right)
4. If the number on the left is bigger than the one on the right swap the cards
5. Stop

Bubble Sort

- Bubble sort sorts a sequence (ADT) of values
- Based on a structured pattern of **comparison-exchange (CE)** operations
- `comparison_exchange(i)`: Take value in two adjacent slots in the sequence and if the values are out of order (i.e. the larger before the smaller), then swap them around

Psuedocode: Bubble Sort

```
Algorithm bubble_sort
Input:  A an array of n elements
Output:  A is sorted
for s = 1 to n-1 do
    for current = 0 to n-2 do
        if A[current] > A[current + 1] then
          swap A[current] and A[current + 1]
        endif
    endfor
endfor
```

Python Bubble Sort

```
def bubblesort(alist):
    for passnum in range(len(alist)-1,0,-1):
        for i in range(passnum):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp
```

Python: Bubble Sort

Adding debug printouts to follow the code:

```
def bubblesort(alist):
    for passnum in range(len(alist)-1,0,-1):
        for i in range(passnum):
            print(i, ':  comparing: ', alist[i],alist[i+1])
            if alist[i]>alist[i+1]:
                print('     ', alist[i], '>', alist[i+1], ' => switch')
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp
                print("     New list order: ", alist)


def main():
    cards=[10, 5, 2, 6, 9, 4]
    print('before sorting:',cards)
    bubblesort(cards)
    print('after sorting:',cards)

if __name__ == '__main__':
    main()
```

Complexity Analysis: Next Lecture

```
Algorithm bubble_sort
Input:  A an array of n elements
Output:  A is sorted
for s = 1 to n-1 do # 1 op per loop
    for current = 0 to n-2 do # 1 op per loop per loop
        if A[current] > A[current + 1] then
          # 1 op per loop per loop
          swap A[current] and A[current + 1]
          # 1 op per loop per loop
        endif
    endfor
endfor
```

Running Time and Complexity

$T(n) = 3(n-1)^2 + n - 1$

Complexity: $\mathcal{O}(n^2)$

Summary

Today

Algorithm Analysis: attributes of a good algorithm
Algorithm Design Exercise: Assumptions, Pseudocode
Take home message: Pen and paper problem solving – use example data inputs/outputs and solve the problem on paper to get it clear in your head