Computational Thinking
Temasek Junior College
Davin Choo
cxjdavin+CTC@gmail.com
Who am I?

- Ex-TJC Student (CG24/07)
- IT Club Secretary
  (while it was under Mr Low Chang Hong)
- Currently in NUS
  - Computer Science and Mathematics DDP
Computational Thinking

- Logical Thinking
- Modelling
- Decomposition
- Pattern recognition
- Pattern generalisation
- Abstraction
- Algorithmic Thinking
- Efficiency
In-class Exercise

Travel Agency
Travel Agency

**Given**

A list of tourists, each with places they want to visit

**To do**

Charter bus rides for them so each tours get to see all of their places
Tourism Spots in Singapore

- Botanical Gardens
- Gardens by the Bay
- Marina Bay Sands
- Sentosa Island
- Jurong Bird Park
- Singapore Flyer
- Universal Studios Singapore
- Clarke Quay

http://commons.wikimedia.org/wiki/File:Merlion_Sentosa.jpg
Tourists

- Amy
- Ben
- Charlie
- Dominic

- Emma
- Felicia
- Ginna
- Harry
Question

• Do location names matter?
• Do tourist names matter?
• Do tourist genders matter?
Tourism Spots in Singapore

- Botanical Gardens [BG]
- Gardens by the Bay [GB]
- Marina Bay Sands [MBS]
- Sentosa Island [SI]
- Jurong Bird Park [JBP]
- Singapore Flyer [SF]
- Universal Studios Singapore [USS]
- Clarke Quay [CQ]
Tourists

- Amy [A]
- Ben [B]
- Charlie [C]
- Dominic [D]
- Emma [E]
- Felicia [F]
- Ginna [G]
- Harry [H]
# Tourists’ Plans

<table>
<thead>
<tr>
<th></th>
<th>BG</th>
<th>GB</th>
<th>MBS</th>
<th>SI</th>
<th>JBP</th>
<th>SF</th>
<th>USS</th>
<th>CQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>D</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>E</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>F</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>H</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Tourists’ Plans

<table>
<thead>
<tr>
<th></th>
<th>BG</th>
<th>GB</th>
<th>MBS</th>
<th>SI</th>
<th>JBP</th>
<th>SF</th>
<th>USS</th>
<th>CQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>C</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>D</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>E</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>F</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>G</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>H</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
# Tourists’ Plans

<table>
<thead>
<tr>
<th></th>
<th>BG</th>
<th>GB</th>
<th>MBS</th>
<th>SI</th>
<th>JBP</th>
<th>SF</th>
<th>USS</th>
<th>CQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>B</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>C</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>D</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>E</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>F</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>G</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>H</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Tourists’ Plans

<table>
<thead>
<tr>
<th></th>
<th>BG</th>
<th>GB</th>
<th>MBS</th>
<th>SI</th>
<th>JBP</th>
<th>SF</th>
<th>USS</th>
<th>CQ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>E</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>G</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>H</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Solution #1

- Singapore 1-Day Tour
  - Put all tourists in 1 bus
  - Visit all 8 places in 1 day

- Pros
  - Works - 1 bus, 1 day

- Cons
  - Tourists unhappy: Too rushed. No time to see anything!
Solution #2

- **Constraint 1**: Each tourist visits at most 1 place/day

- Singapore Buffet-style tour
  - Schedule 8 buses to each location every day
  - Tourists pick which bus to take each day

- **Pros**
  - Works - At most 3 days to complete all plans

- **Cons**
  - Boss unhappy: Wasteful! 24 bus trips.
Solution #3

- **Constraint 1**: Each tourist visits at most 1 place/day
- **Constraint 2**: Send at most 1 bus to each place

**Singapore 8-day tour**

- Schedule 1 bus to a different location each day
- Tourists pick which day to take the bus

**Pros**

- Works - At most 8 bus trips

**Cons**

- Tourists unhappy: 8 days needed. At least 5 wasted days
Moral of the Story

• Don’t work in a travel agency

• Just kidding

• Solutions to real world problems are affected by all stakeholders
Hands on

• **Constraint 1**: Each tourist visits at most 1 place/day
• **Constraint 2**: Send at most 1 bus to each place
• **Constraint 3**: Minimise number of days
Hands on

• How many days did you use?

• How did you come up with the solution?

• What if I increase the number of attractions to 100?

• What if I increase the number of tourists to 100?
Graph Model Approach

• What is a graph?

![Graph of sin(x) and cos(x) with x from -6 to 6]

• Nope
Graph Model Approach

- What is a model?

Devi and Minah have $520 altogether. If Devi spends \( \frac{2}{5} \) of her money and Minah spends $40, then they will have the same amount of money left. How much money does Devi have? (Kho, 1987)

8 units = $520 - $40 = $480
1 unit = $480 ÷ 8 = $60
Devi's money = 5 units
= 5 \times $60
= $300

Similar
Graph Model Approach

• A mathematical model $G = (V, E)$

• Nodes/Vertices ($V$)

• Edges ($E$)

• In our case:
  • $V = $ Attractions
  • $E = $ Conflicts
Hands on

Draw the constraint graph

<table>
<thead>
<tr>
<th></th>
<th>BG</th>
<th>GB</th>
<th>MBS</th>
<th>SI</th>
<th>JBP</th>
<th>SF</th>
<th>USS</th>
<th>CQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>C</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>D</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>E</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Constraint Graph

<table>
<thead>
<tr>
<th>BG</th>
<th>GB</th>
<th>MBS</th>
<th>SI</th>
<th>JBP</th>
<th>SF</th>
<th>USS</th>
<th>CQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Re-arranging...
Graph Model

- **Usage**
  - Easy to read constraint
  - (Or the lack of it)

- **Claim**
  - Colour the vertices. Adjacent vertices not same colour
  - # colours used = # days needed
  - Same color = Visit on same day
Possible solution

Day 1 - JBP, MBS
Day 2 - BG, SI
Day 3 - CQ, USS
Day 4 - GB, SF
Questions

- Does the ordering of the colours matter?
- How do we get the list of tourists on each bus?
Models

Actual Problem → Actual Solution

Model Problem → Model Solution

Known methods
Models

Bend Steel Bar → Bent Steel Bar

http://youtu.be/MkibO5PdAiI?t=29s
Models

Bend Steel Bar → Bent Steel Bar

???

Physics Model → Calculated heat and pressure points

Apply Heat and Pressure
Models

Bus Scheduling → Bus Schedule

Constraint Graph

Colour graph

Vertex Colouring
Exam Scheduling

**Given**

A list of students, each with subjects that they take

**To do**

Plan the exam times and dates for the school
Fighting fish

**Given**

A list of fish, each with a list of other fish which they will fight with

**To do**

Use as little bowls to hold all the fish
Models

Exam Scheduling
Fighting Fish

Exam Schedule
Fish Bowls

Constraint
Graph

Colour graph

Vertex Colouring
Math to Wolfram

Conrad Wolfram
Conrad Wolfram is a British technologist and businessman known for his work in information technology and its application. Wikipedia
Born: June 10, 1970 (age 43), Oxford, United Kingdom
Siblings: Stephen Wolfram
Education: University of Cambridge, Eton College, Dragon School

What is math?

1. Posing the right questions
2. Real world $\rightarrow$ math formulation
3. Computation
4. Math formulation $\rightarrow$ real world, verification

http://www.wolframalpha.com
Wolfram’s TED talk

19min 19sec
Lessons Learnt

• Abstraction
  • Remove useless information that don’t help in solving the problem (e.g. names)

• Simplify if possible

• Real world problems are usually subject to many constraints

• Models are helpful
  • Grants you access to well-known problems/solutions
Palindromes

• What are palindromes?
  • It is the same thing when you read it both forwards and backwards

• Are these strings palindromes?
  • 121
  • ASDF
  • ASDFDSDA
  • 1
Homework

1) Palindromes

Given a string,
How do you check if it is a palindrome?

2) Read this and tell me the moral of the story:

http://www.comp.nus.edu.sg/~leonghw/Courses/cattywampus.html
If you are interested...

• Topics covered:
  • Abstraction
  • Modelling
  • Graph Theory
  • Graph colouring
Computational Thinking
Temasek Junior College
Davin Choo
cxjdavin+CTC@gmail.com
Homework

2) What is the moral of the Cattywampus story?

1) Palindromes

• It is the same thing when you read it both forwards and backwards

Given a string,
How do you check if it is a palindrome?
Discussion

• Are the steps provided by your classmates unambiguous/clear?

• Can you use their steps to check if something is a palindrome? Let’s try…
Next few sessions *might* feel like this

The Karate Kid (2010)
4min 19sec

https://www.youtube.com/watch?v=hkORWh_wbcY
But there are **good** reasons…

The Karate Kid (2010)
1min 53sec

https://www.youtube.com/watch?v=T10ycFr770g
Goal of next few sessions

• Teach you the basics and foundations (Important!)

• A step into understanding the world of Computer Science

• The core of Computer Science is “Algorithms”
Algorithms

• Google’s definition

  algorithm

  /ˈæl.gəˌrɪðəm/

  noun
  noun: algorithm; plural noun: algorithms

  1. a process or set of rules to be followed in calculations or other problem-solving operations, esp. by a computer.

• Properties

  • Well defined (i.e. not ambiguous)

  • Finite/Fixed number of steps
Algorithm Examples

• Cooking

1. Preheat oven to 350°C

2. Sift together flour, cocoa, baking soda and 1 tsp salt

3. Beat in eggs and vanilla

4. ...
Algorithm Examples

• Giving directions to a tourist (or Google Maps)

  1. Go straight until you reach “XXX Drive”

  2. Make a left turn and walk straight to bus stop #55423

  3. Take bus 42 for 9 stops

  4. ...
What’s “wrong” with the examples?

• Too verbose

• Different people have different “levels of understanding”

• We need a model of how things (in our case, computers) work before we can provide an appropriate algorithm at a “suitable level”
Dijkstra's algorithm

(Can you understand this?)

```
1 function Dijkstra(Graph, source):
2     for each vertex v in Graph:
3         dist[v] := infinity
4         previous[v] := undefined
5     end for
6     dist[source] := 0
7     Q := the set of all nodes in Graph
8     while Q is not empty:
9         u := vertex in Q with smallest distance in dist[]
10        remove u from Q
11        if dist[u] = infinity:
12            break
13        end if
14        for each neighbor v of u:
15            alt := dist[u] + dist_between(u, v)
16            if alt < dist[v]:
17                dist[v] := alt
18                previous[v] := u
19                decrease-key v in Q
20            end if
21        end for
22    end while
23    return dist;
24 endfunction
```

http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
Pseudocode

- “Language” of algorithms

- Used to describe an algorithm in “fairly standardised method” (there are variants)

- Similar to programming code but is programming language agnostic
Pseudocode

- Simple building blocks
  1. Numbering
  2. Comments
  3. Assignment
  4. Print/Return
  5. Conditionals (if-else)
  6. Repetition (while-loops)
  7. Function calls
1. Numbering

ALGORITHM_A (<Inputs>)

1. Do A
2. Do B
3. Do C

END
2. Comments

ALGORITHM_A (<Inputs>)

1. Do A // This helps others
2. Do B // to understand
3. Do C // what you wrote

END
3. Assignment

ALGORITHM_A (<Inputs>)

1. $X \leftarrow 3$
2. Do B
3. Do C

END
4. Print/Return

PLUS_ONE (A)

1. \( X \leftarrow 1 + A \)

2. PRINT “Hello”

3. RETURN X

END

Example:
PLUS_ONE(2)
Shows: “Hello”
Returned: 3

Question:
PLUS_ONE(PLUS_ONE(2)) returns ___?
5. Conditionals

If \textcolor{orange}{\textbf{Condition}}, then \textcolor{green}{\textbf{Condition is met}}

Otherwise, \textcolor{red}{\textbf{Condition not met}}
5. Conditionals

If \( \text{it rains today} \), then \( \text{Natalie will bring umbrella} \)

Otherwise, \( \text{Natalie will wear sunglasses} \)
5. Conditionals

Yun Feng is hungry
If ___________, then ____________

Otherwise, ____________

Yun Feng will eat
<Blank>
5. Conditionals

IS_EVEN (N)

1. IF (A % 2 == 0)

2. RETURN TRUE

3. ELSE

4. RETURN FALSE

5. END IF

END

Condition

X == Y
returns TRUE, if X is equal to Y
returns FALSE, otherwise

Convention (Why == instead of =)
In programming, = denotes ←

Example:
IS_EVEN (5) returns ___?
IS_EVEN (4) returns ___?
5. Conditionals

Nested Conditionals

If (A and B), then ____.

If (A and ¬B), then ____.

Otherwise (i.e. ¬A), ____.
NESTED_IF_ALGO ()

1. IF (A)
   2. IF (B)
   3. _____
   4. ELSE
   5. _____
   6. END IF
   }

   Check Conditional 2

7. ELSE
8. _____
9. END IF
END
6. Repetition

SAY_HI_N_TIMES (N)

1. WHILE (N != 0)
2. PRINT “HI”
3. N ← N - 1
4. END WHILE

END

Condition
X != Y returns TRUE, if X is not equal to Y returns FALSE, otherwise

Convention
! represents “not”/negation

Example:
SAY_HI_N_TIMES (1) returns ___?
SAY_HI_N_TIMES (5) returns ___?
SAY_HI_N_TIMES (-1) returns ___?
6. Repetition

SAY_HI_N_TIMES (N)

1. WHILE (N > 0)
2. PRINT “HI”
3. N ← N - 1
4. END WHILE

END

Example:
SAY_HI_N_TIMES (1) returns ___?
SAY_HI_N_TIMES (5) returns ___?
SAY_HI_N_TIMES (-1) returns ___?

Condition

Choice of condition MATTERS!
6. Repetition

SAY_HI_N_TIMES (N)

1. WHILE (N ≥ 1)
2. PRINT “HI”
3. N ← N - 1
4. END WHILE

Example:
SAY_HI_N_TIMES (1) returns ___?
SAY_HI_N_TIMES (5) returns ___?
SAY_HI_N_TIMES (-1) returns ___?

Condition
(N != 0) vs. (N > 0) vs. (N ≥ 1)

Assumption
MATTERS!
Math Joke

• An empty kettle is on the stove, how do you boil water?

• An empty kettle is on the floor, how do you boil water?
7. Function Calls

MAKE_EVEN (N)
1. IF (!IS_EVEN(N))
2. RETURN (N + 1)
3. ELSE
4. RETURN N
5. END IF
END

MAKE_EVEN (N)
1. IF (!IS_EVEN(N))
2. RETURN PLUS_ONE(N)
3. ELSE
4. RETURN N
5. END IF
END
Learning Points

• Pick your conditions carefully

• Be lazy. Re-use previous solutions!
  • Don’t repeat work
  • Don’t reinvent the wheel
  • Unless your wheel is better (Prove it)

• Decompose/Break down large problems into smaller, more manageable parts
Dijkstra’s algorithm
(How about now?)

function Dijkstra(Graph, source):
    for each vertex v in Graph:
        \[ \text{dist}[v] := \text{infinity} ; \]
    previous[v] := undefined ;
end for

dist[source] := 0 ;
Q := the set of all nodes in Graph ;

while Q is not empty:
    u := vertex in Q with smallest distance in dist[] ;
    remove u from Q ;
    if dist[u] = infinity:
        break ;
    end if

    for each neighbor v of u:
        alt := dist[u] + dist_between(u, v) ;
        if alt < dist[v]:
            dist[v] := alt ;
            previous[v] := u ;
            decrease-key v in Q ;
        end if
    end for
end while
return dist;
end function

// Initializations
// Unknown distance function from
// source to v
// Previous node in optimal path
// from source

// Distance from source to source
// All nodes in the graph are
// unoptimized – thus are in Q
// The main loop
// Source node in first case

// all remaining vertices are
// inaccessible from source

// where v has not yet been
// removed from Q.

// Relax (u,v,a)

// Reorder v in the Queue
Factorial

• Definition:
  
  • \( n! = n \times (n-1) \times (n-2) \times \ldots \times 1 \)

• Is there a pattern here?
  Can we break down the definition into smaller, easier parts?
  
  • \(0! = 1, 1! = 1\) \hspace{2cm} \text{Base case}
  
  • \(n! = n \times (n-1)!\) \hspace{2cm} \text{Recursive case}
Hands on

- How would you implement Factorial in Pseudocode?

`FACTORIAL(n)`

- Example:
  - `FACTORIAL(0) = 1`
  - `FACTORIAL(1) = 1`
  - `FACTORIAL(3) = 6`

Recall:

\[ n! = n \times (n-1) \times (n-2) \times \ldots \times 1 \]
\[ n! = n \times (n-1)! \]
Factorial

**Factorial**

```plaintext
FACTORIAL(n)

1. result ← 1
2. WHILE (n ≥ 1)
3. result ← result * n
4. n ← n - 1
5. END WHILE
6. RETURN result
END
```

**Question:**
What if I change to > ?
Will it work?

Which definition is this using?
Can we use the other one?

**Recall:**

\[ n! = n \times (n-1) \times (n-2) \times \ldots \times 1 \]
\[ n! = n \times (n-1)! \]
Factorial

FACTORIAL(n)

1. IF (n == 0 OR n == 1)
   2. RETURN 1

3. ELSE

4. RETURN n * FACTORIAL(n-1)

5. END IF

END

Question:
What if I change to “AND”? Will it work?

Recall:

\[ n! = n \times (n-1) \times (n-2) \times \ldots \times 1 \]
\[ n! = n \times (n-1)! \]
Model of Computer

• No “intermediate” memory
  • Use variables (boxes) to hold data
  • A single variable can hold 1 value
    • Overwritten when asked to store another value
  • Instructions are executed sequentially
    (i.e. step by step, unless loop or function call)
• Index numbering start from 0 (why: Binary numbers)
Abstraction!

Circuitry

Bits

Assembly language

Human readable programming languages

Pseudocode

Current flows, resistors, NAND gates

0’s and 1’s (like in the Matrix)

MIPS / ARM architecture (PUSH, MOV, etc)

Java, C++, Python, JS, etc

What we are currently doing
We want to swap the values of 2 variables, how do we do it?
Sequences / Arrays

• A contiguous (joined-up) chain of variables (boxes)

• Values are referenced by 1 name (chain’s name) and the index (position in the chain)

• Example:

```plaintext
LEN(SEQ) = 6 // Length of SEQ
SEQ[0] = 4
SEQ[1] = 3
SEQ[2] = 2
SEQ[3] = 7
SEQ[4] = 3
SEQ[5] = 6
```
Hands on

• How would you implement Swap in an array in Pseudocode?

\[ \text{SWAP}(\text{SEQ}, i, j) \]

• Example:

\[
\begin{align*}
\text{SEQ} &= [A, B, C, D, E] \\
\text{SWAP}(\text{SEQ}, 1, 4) \\
\text{SEQ} &= [A, E, C, D, B]
\end{align*}
\]
SWAP

SWAP (SEQ, i, j)

1. SEQ[i] ← SEQ[j]  

2. SEQ[j] ← SEQ[i]  

END

Does this work?

Why / Why not?
SWAP (SEQ, i, j)

1. temp ← SEQ[i]
2. SEQ[i] ← SEQ[j]
3. SEQ[j] ← temp

END
Discussion

• SWAP(SEQ, i, j) == SWAP(SEQ, j, i) ?

• Symmetric in 2\(^{nd}\) and 3\(^{rd}\) input parameters

• What other kinds of operations/functions are symmetric?
  
  • ADD(i, j), MULTIPLY(i, j)
Hopefully, you will be like…

Kung Fu Panda (2008)
4min 12sec

https://www.youtube.com/watch?v=84VtdVK2u0A
One last thing for today
FIND_MAX

• Idea: Find largest value in an array of numbers

• Example:

SEQ

<table>
<thead>
<tr>
<th>4</th>
<th>3</th>
<th>2</th>
<th>7</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
</table>

Largest value = 7
Largest index = 3

SEQ

<table>
<thead>
<tr>
<th>-4</th>
<th>3</th>
<th>2</th>
<th>-7</th>
<th>3</th>
<th>-6</th>
</tr>
</thead>
</table>

Largest value = 3
Largest index = 1 or 4?
FIND_MAX

- Should FIND_MAX return:
  - "largest value", or
  - "index of largest value"?
- Why?
Homework

1) Find recipe of favourite food (as per the survey!)

Write algorithm so that a computer/automated chef can understand it.
(Use repetitions and function calls where suitable!)

2) Write pseudocode of FIND_MAX(SEQ),
which returns the first index of the largest value

FIND_MAX(SEQ)

1. ...

END
If you are interested...

• Topics covered:
  • Pseudocode Structure
  • Working model of a computer
  • Arrays
  • Factorial, Swap, Find_max
Computational Thinking
Temasek Junior College
Davin Choo
cxjdavin+CTC@gmail.com
Homework

Present your pseudocode for \textsc{multiply}(A, B) \textit{using only additions}

May assume that A and B are both \textit{natural numbers}
Multiply

• Definition:

  • \( A \times B = A + A + \ldots + A \) (\( B \) times)

• Is there a pattern here? Can we break down the definition into smaller, easier parts?

  • \( A \times 1 = A \)  
    
    Base case

  • \( A \times B = A + (A \times (B-1)) \)  
    
    Recursive case
Multiply

MULTIPLY(A, B)

1. result ← 0
2. WHILE (B ≥ 1)
   3. result ← result + A
   4. B ← B - 1
3. END WHILE
4. RETURN result
5. END

Question:
What if I change to >? Will it work?

Which definition is this using?
Can we use the other one?

Recall:
A*B = A + A + … + A
A*B = A + (A * (B-1))
Multiply

MULTIPLY(A, B)

1. IF (B == 1)
2. RETURN A
3. ELSE
4. RETURN A + MULTIPLY(A, B-1)
5. END IF

END

Recall:
A*B = A + A + … + A
A*B = A + (A * (B-1))
Continuing from last week…

Let’s finish off the material from last week, and then we’ll start on Javascript!

(if time permits)
FIND_MAX

• Idea: Find largest value in an array of numbers

• Example:

  SEQ

  4 3 2 7 3 6

  Largest value = 7
  Largest index = 3

SEQ

-4 3 2 -7 3 -6

Largest value = 3
Largest index = 1 or 4?
**Find Max**

FIND_MAX(SEQ)

1. currentMax ← -∞; maxIndex ← -1; i ← 0
2. WHILE (i < LEN(SEQ))
3. IF (SEQ[i] > currentMax)
4. currentMax ← SEQ[i]; maxIndex ← i
5. END IF
6. i ← i+1
7. END WHILE
8. RETURN maxIndex

END

**Question:**
What if I change to \(\leq\)? Will it work?

**Question:**
What if I want the last index? What if I want all indices?

<table>
<thead>
<tr>
<th>SEQ</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>7</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ</td>
<td>-4</td>
<td>3</td>
<td>2</td>
<td>-7</td>
<td>3</td>
<td>-6</td>
</tr>
</tbody>
</table>
Computational Thinking
Temasek Junior College
Davin Choo
cxjdavin+CTC@gmail.com
Why Javascript (JS)

- Every *modern* browser comes with a javascript compiler
- Supports multiple programming paradigms
- You can use it to make interesting web applications or make your websites more interesting (together with HTML and CSS)
Working environments

- Chrome developer console
- Collabedit (http://collabedit.com)
- Notepad / Any text editors
Chrome Developer Console
Collabedit

Online Code Editor

Collabedit is an online code editor that lets people collaborate in real-time. It works in your web browser so no installation is needed. Try it now, no account necessary, just click here.

Features

- text editor
- chat
- document history
- syntax highlighting for programming languages

Useful For

Technical Phone Interviews  Collaborative Coding  Teach Programming

Here's what people are saying
Collabedit
Collabedit
Collabedit
Syntax vs. Semantics

• Syntax
  • “Grammar structure”

• Semantics
  • Meaning
Syntax vs. Semantics

• John ate a hotdog

• Syntax
  • John [Noun], ate [verb], hotdog [Noun]
  • Sentence Form: <Noun> <Verb> <Noun>

• Semantics
Syntax vs. Semantics

• “Store the value 3 into the variable (box) X”

• Syntax
  • $X \gets 3$ (Pseudocode)
  • `var X = 3;` (Javascript)

• Semantics
Basic Javascript Syntax

- Simple building blocks
  1. Numbering
  2. Comments
  3. Assignment
  4. Print/Return
  5. Conditionals (if-else)
  6. Repetition (while-loops)
  7. Function calls

From Pseudocode slides
2. Comments
3. Assignments

```javascript
> var X = 3; // Assigns the value 3 to the variable X
undefined
> X
3
> 
```
4. Print/Return

Print

```
3
```

Returned

```
undefined
```

Print

```
2
```

Returned

```
undefined
```

Print

```
1
```
5. Conditionals

**Pseudocode**

```
IS_EVEN (N)
1. IF (N % 2 == 0)
2. RETURN TRUE
3. ELSE
4. RETURN FALSE
5. END IF
END
```

**Javascript**

```javascript
function isEven(N) {
  if (N % 2 == 0) {
    return true;
  } else {
    return false;
  }
}

undefined
> isEven(5)
false
> isEven(4)
true
```
6. Repetition

SAY_HI_N TIMES (N)
1. WHILE (N != 0)
2. PRINT “HI”
3. N ← N - 1
4. END WHILE
END

Pseudocode

Javascript

function sayHiNTimes(N) {
  while (N != 0) {
    console.log("Hi");
    N = N-1; // We can also write N = N - 1
  }
}

undefined
> sayHiNTimes(1)
Hi
undefined
> sayHiNTimes(5)
5 Hi
undefined
> sayHiNTimes(-1)
65472 Hi

Caused my Chrome to hang

Returns nothing

Prints 5 times
7. Function Calls

**Pseudocode**

```
MAKE EVEN (N)
1. IF (!IS EVEN(N))
2. RETURN PLUS ONE(N)
3. ELSE
4. RETURN N
5. END IF
END
```

**Javascript**

```javascript
> function plus_one(N) {
>     return N + 1;
> }
> function makeEven(N) {
>     if (!isEven(N)) {
>         return plus_one(N);
>     } else {
>         return N;
>     }
> }
> makeEven(1)
> 2
> makeEven(4)
> 4
```
MULTIPLY(A, B) in JS

Pseudocode:

MULTIPLY(A, B)

1. result ← 0
2. WHILE (B ≥ 1)
3. result ← result + A
4. B ← B - 1
5. END WHILE
6. RETURN result

END

Javascript (Collabedit):

```javascript
function multiply(A, B) {
    var result = 0;
    while (B >= 1) {
        result = result + A;
        B = B - 1;
    }
    return result;
}
```

Javascript (Chrome):

```javascript
> function multiply(A, B) {
>     var result = 0;
>     while (B >= 1) {
>         result = result + A;
>         B = B - 1;
>     }
>     return result;
> }

> multiply(2, 3)
6
> multiply(43, 123)
5289
```
MULTIPLY(A, B) in JS

Pseudocode

1. IF (B == 1)
2. RETURN A
3. ELSE
4. RETURN A + MULTIPLY(A, B-1)
5. END IF
END

Javascript (Collabedit)

```
function multiply(A, B) {
    if (B == 1) {
        return A;
    } else {
        return A + multiply(A, B-1);
    }
}
```

Javascript (Chrome)

```javascript
> function multiply(A, B) {
    if (B == 1) {
        return A;
    } else {
        return A + multiply(A, B-1);
    }
}
> multiply(3,2)
6
> multiply(123,43)
5289
```

Testing...
Sequences/Arrays in JS

Pseudocode

<table>
<thead>
<tr>
<th>SEQ</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>7</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEN(SEQ) = 6 // Length of SEQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ[0] = 4   SEQ[3] = 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"Out of bounds"

Javascript

```
var SEQ = [4, 3, 2, 7, 3, 6]
undefined
> SEQ[0]
4
> SEQ[1]
3
> SEQ[2]
2
> SEQ[3]
7
> SEQ[4]
3
> SEQ[5]
6
> SEQ[6]
undefined
> SEQ[-1]
undefined
```
Homework

Write Javascript code for:

**FACTORIAL(N)** and **SWAP(SEQ, i, j)**

- Both versions of Factorial, so 3 codes in total
- Make sure it works
  - Try on some simple test cases
  - Check output by hand
- Email me your code in “.txt” format by **Sunday 23rd Feb 2014**
If you are interested...

- Topics covered:
  - Syntax vs. Semantics
  - Basic Javascript syntax
Computational Thinking
Temasek Junior College

Davin Choo
cxjdavin+CTC@gmail.com
Plans for today

• Recap some basic Javascript

• Practice, practice, practice!

• Learn Binary Search

• Update on interesting problem set:
  • Currently still in talks to with Professor Martin Henz and his team on getting hold of it
Homework

Write Javascript code for:

\textsc{factorial}(N) \text{ and } \textsc{swap}(\textsc{seq}, i, j)

• Both versions of Factorial, so 3 codes in total

• Make sure it works
  • Try on some simple test cases
  • Check output by hand
Questions asked

- How to type Javascript
  - Chrome console vs. Collabedit vs. Notepad
  - Syntax (Review in a bit)
- How to test
Method 1 (Collabedit)

1. Type in **Collabedit**
2. Copy the entire function
3. Open Chrome console
4. Paste into Chrome console
Method 2 (Notepad)

1. Type in **Notepad**
2. Copy the entire function
3. Open Chrome console
4. Paste into Chrome console
Method 3 (Console)

1. Open Chrome console
2. Type in Chrome console
3. Use Shift+Enter to go to the next line of your code
Things to remember

• Use “1 tab” or “4 spaces” to indent your code
  • Improve readability
• Curly braces “{“ and “}”
• “var”
• Capitalisation makes a difference
• Don’t type the numberings in JS!
  • That’s for line referencing purposes
Basic Javascript Syntax

- Simple building blocks
  1. Numbering
  2. Comments
  3. Assignment
  4. Print/Return
  5. Conditionals (if-else)
  6. Repetition (while-loops)
  7. Function calls

From Pseudocode slides
2. Comments
3. Assignments

```javascript
> var X = 3; // Assigns the value 3 to the variable X
undefined
> X
3
> 
```
4. Print/Return

Print

```
> console.log(X)
3
< undefined
> console.log(1+1)
2
< undefined
> 1
```

← Print
← Returned
← Print
← Returned
5. Conditionals

**Pseudocode**

```
IS_EVEN (N)
1. IF (N % 2 == 0)
2. RETURN TRUE
3. ELSE
4. RETURN FALSE
5. END IF
END
```

**Javascript**

```
function isEven(N) {
  if (N % 2 == 0) {
    return true;
  } else {
    return false;
  }
}
```

```
> isEven(5)
false
```

```
> isEven(4)
ture
```

Returns nothing
6. Repetition

Pseudocode

```
SAY_HI_N_TIMES (N)
1. WHILE (N != 0)
2. PRINT “HI”
3. N ← N - 1
4. END WHILE
END
```

Javascript

```
function sayHiNTimes(N) {
    while (N != 0) {
        console.log("Hi");
        N = N-1; // We can also write N -- = 1
    }
}
```

- Returns nothing
- Prints 5 times
- Caused my Chrome to hang
7. Function Calls

**Pseudocode**

```
MAKE_EVEN (N)
1. IF (!IS_EVEN(N))
2. RETURN PLUS_ONE(N)
3. ELSE
4. RETURN N
5. END IF
END
```

**Javascript**

```
function plus_one(N) {
    return N+1;
}
function makeEven(N) {
    if (!isEven(N)) {
        return plus_one(N);
    } else {
        return N;
    }
}
```

```javascript
> makeEven(1)
2
> makeEven(4)
4
```
Hands on

• Pair up and code up:
  • Factorial (Iterative version)
  • Factorial (Recursive version)
  • Swap (Given SEQ, i, j)

• Pairs
  • Wee Teck, Taha
  • Leng Ze, Natalie
  • Nicolas, Yun Fen

Those who did, guide those who didn’t.
Don’t just type for them
How to test

• Suppose you coded up function `fun()`

• Create “test cases”
  • Pick `<{inputs}, output>`. Check output by hand.

• Run `fun(inputs)` in console

• Does it return `output`?
  • If yes, it passed this test case. But not necessarily correct.
  • If not, it is definitely wrong.
Example

- \( \text{factorial}(3) = 6 \)
  - \(<\{\text{inputs}\}, \text{output}\> = <\{3\}, 6>\)

- \( \text{swap}([1, 2, 3], 0, 1) \) returns \([2, 1, 3]\)
  - \(<\{\text{inputs}\}, \text{output}\> = <\{[1, 2, 3], 0, 1\}, [2, 1, 3]>\)
Binary Search

- Recall what is an array/sequence
- Assume it is sorted in some fixed order
  - “Ascending”, based on some measure
- **Question:**
  - Is “x” in the array?
  - If yes, give me its index
  - If no, tell me it is not in the array.
Binary Search

• Examples
  • Searching in a dictionary
  • Searching in a telephone book
  • Searching in a list of names in a class roster
  • Etc…
Hands on

• **Question:**
  Is “x” in the array?
  If yes, give me its index
  If no, tell me it is not in the array.

• Find the following words in the dictionary:
  1. Pseudocode
  2. Algorithm
  3. Binary
Ideas?

1. Flip from front to end
   - Very slow. Worst case: Need to check all entries

2. Flip to some kind of pre-partitioned index and search within that section
   - Need to pre-process before hand
   - Still bad if partitions are huge

3. Binary search
**Binary Search (Idea)**

- If array is only length 1, check directly if x is there
- Otherwise:
  - Look at middle of array, is x there?
  - If yes, done
  - If no, ask whether x should be in left or right half?
  - Consider searching in that half
Hands on

Formulate the pseudocode of Binary Search
Binary Search

BINARY_SEARCH (arr, L, R, x)
1. IF (L > R)  // Typical index value to denote failure
2. RETURN -1
3. END IF
4. middle ← ⌈(L + R) / 2⌉  // Calculate the middle page (round up)
5. IF (arr[middle] == x)  // Suppose every page has only 1 name
6. RETURN middle  // Return page number
7. ELSE IF (arr[middle] > x)  // If middle page is “too large”
8. RETURN BINARY_SEARCH (arr, L, middle-1, x)  // Recurse on left half
9. ELSE  // If middle page is “too small”
10. RETURN BINARY_SEARCH (arr, middle+1, R, x)  // Recurse on right half
11. END IF
END
Homework

Write Javascript code for:
BINARY_SEARCH (arr, L, R, x)

• 1 code in total

• Make sure it works
  • Start early!

• Specify and show at least 3 test cases

• Email me your code in “.txt” format by Sunday 2\textsuperscript{nd} Mar 2014
If you are interested...

- Topics covered:
  - Revision of “Basic Javascript syntax”
  - Basic “test case” testing technique
  - Binary Search
Computational Thinking
Temasek Junior College
Davin Choo
cxjdavin+CTC@gmail.com
Mini-test next Wed
12\textsuperscript{th} March

- 45 minutes, at the start, before we begin on Runes

- Takes only \~10 minutes if you know your stuff

- Review what you have learnt so far

- How to read & write Pseudocode

- How to read & write Javascript code / Translate from Pseudocode

- Recursion vs. Iteration
Plans for today

• Binary search. Everyone practice.
• Recursive solutions
• Iterative solutions
Binary Search (Idea)

• If array is only length 1, check directly if x is there

• Otherwise:
  • Look at middle of array, is x there?
  • If yes, done
  • If no, ask whether x should be in left or right half?
  • Consider searching in that half
Binary Search

BINARY_SEARCH (arr, L, R, x)

1. IF (L > R)  // Typical index value to denote failure
2. RETURN -1
3. END IF
4. middle ← ⌈(L + R) / 2⌉  // Calculate the middle page (round up)
5. IF (arr[middle] == x)  // Suppose every page has only 1 name
6. RETURN middle  // Return page number
7. ELSE IF (arr[middle] > x)  // If middle page is “too large”
8. RETURN BINARY_SEARCH (arr, L, middle-1, x)  // Recurse on left half
9. ELSE  // If middle page is “too small”
10. RETURN BINARY_SEARCH (arr, middle+1, R, x)  // Recurse on right half
11. END IF

END
Recursive solutions

- Idea: Solve smaller parts, combine to form solution
- Components
  - Base case (BC)
  - Recursive case (RC)
- Identify the BC and RC in the following examples
Recursion Examples

Recursion

Recursion is the process of repeating items in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested images that occur are a form of infinite recursion.

Related topics

Recursive data structures can dynamically grow to a theoretically infinite size in response to runtime requirements; in contrast, the size of a static array must be set at compile time.

Explore: Data structure

In mathematics, a recurrence relation is an equation that recursively defines a sequence, once one or more initial terms are given; each further term of the sequence is defined as a function of the preceding terms.

Explore: Recurrence relation

Dynamic programming is an approach to optimization that restates a multiperiod or multistep optimization problem in recursive form.

Explore: Dynamic programming
Recursion Examples

Canteen queue
Recursion Examples

Sesame Street - Russian Dolls (1-10)

http://upload.wikimedia.org/wikipedia/commons/7/7f/Floral_matryoshka_set_1.JPG

https://www.youtube.com/watch?v=aNQV45Wichw

http://upload.wikimedia.org/wikipedia/commons/7/7f/Floral_matryoshka_set_1.JPG
Recursion Examples

```
MULTIPLY(A, B)
1. IF (B == 1)
2. RETURN A
3. ELSE
4. RETURN A + MULTIPLY(A, B-1)
5. END IF
END
```

Pseudocode for Multiply(A,B)
Recursion Examples

BINARY_SEARCH (arr, L, R, x)
1. IF (L > R) // Typical index value to denote failure
2. RETURN -1
3. END IF
4. middle ← ⌈(L + R) / 2⌉ // Calculate the middle page (round up)
5. IF (arr[middle] == x) // Suppose every page has only 1 name
6. RETURN middle // Return page number
7. ELSE IF (arr[middle] > x) // If middle page is “too large”
8. RETURN BINARY_SEARCH (arr, L, middle-1, x) // Recurse on left half
9. ELSE // If middle page is “too small”
10. RETURN BINARY_SEARCH (arr, middle+1, R, x) // Recurse on right half
11. END IF
END
Iterative solutions

• Idea: Repeat step by step until termination

• Components
  • Iterating variable(s)
  • Terminating condition

• Identify the components in the following examples
Iteration Examples

Doing a worksheet of N problems
Iteration Examples

MULTIPLY(A, B)
1. result ← 0
2. WHILE (B ≥ 1)
3. result ← result + A
4. B ← B - 1
5. END WHILE
6. RETURN result

END

Pseudocode for Multiply(A,B)
Iteration Examples

Canteen queue of N people
Recursion vs. Iteration

• Recursion is performed iteratively in a computer

• This means:
  Anything written in recursion form can be re-written in an equivalent iteration form

• Recursion form may be more intuitive, natural and/or easier to understand than it's iterative form
Hands on

- Do it together on screen
- Both recursive and iterative solutions
- Remember testing
Hands on

- addOne(x)
- addition(A,B) using addOne(x). result = A + B
- subtractOne(x) using addOne(x). result = A + B
- subtract(A,B) using subtractOne(x). result = A - B
- multiply(A,B) using addition(A,B). result = A * B
- Challenge: divide(A,B,n). result = A/B (where n is number of digits of answer)
Power

- Also called exponentiation
- $\text{power}(A, B) = A^B$
- E.g.
  - $\text{power}(2, 5) = 32$
  - $\text{power}(10, 3) = 1000$
Power

- power(A,B) = $A^B$
- $A*A*...*A$ (B times)
- A * power(A, B-1)

- Which is recursive, which is iterative? What are the components?
Homework

Write Javascript code for: **POWER (A, B)**

- **Both** versions of power, so 2 codes in total
- Make sure it works
  - *Start early!*
- Specify and show at least 3 test cases
- Email me your code in “.txt” format by **Sunday 9th Mar 2014**
Mini-test
12th March

45 minutes, at the start, before we begin on Runes

Takes only ~10 minutes if you know your stuff
Runes Problem Set

- Adapted from NUS Course: CS1101S Programming Methodology
  - First semester CS course that selected few go through
- Javascript implementation of Runes (Credit: Professor Martin Henz and his team)
- Modified slides from CS1101S (Credit: Professor Ben Leong)
- Quite a fair bit of things to cover. Please be more responsive and active in hands on.
Function

- A.k.a. “Procedure”

- Computers follow instructions exactly

- Associates a name to a sequence of operations

- e.g.
  - function addOne(x) { return x+1; }
  - function ASDF(x) { return x+1; }

- These 2 functions are given different names but perform the exact same operations
Functional Abstraction

• Treat functions as “black box”

• You *only need to know* **what** it does

• You *don’t have to know* **how** it does it
Abstract Environment

- Previous weeks:
  - Numbers
  - Mathematical functions

- Next few sessions:
  - Pictures! ("Runes")
Elements of Programming

• Primitives
• Combination
• Abstraction
Primitives

show(rcross_bb)
Primitives

show(sail_bb)
Primitives

\[ \text{show(corners_bb)} \]
Primitives

show(nova_bb)
Primitives

show(heart_bb)
Primitives

show(circle_bb)
Primitives

show(ribbon_bb)
Primitives

show(black_bb)
Primitives

\texttt{show(\text{blank\_bb})}
Primitives

show(pentagram_bb)
Wait...

- What does show do?
  - Take in an image and display it on Firefox
- How is show implemented?
  - We don’t care. We don’t need to know
  - “Functional abstraction”
How to get “clear” the box?

clear_all()
Rotating 90° clockwise

\[ \text{show(quarter\_turn\_right(sail\_bb))} \]

Image \[\text{quarter\_turn\_right}\] Image (Rotated 90° clockwise)

Original Result
Rotating 180°?

```javascript
function rotate180(img) {
    return quarter_turn_right(quarter_turn_right(img));
}

show(rotate180(sail_bb));
```

Original | Result
Rotating 90° left?

• Do we need a new primitive function?
• What can we make use of?

function \texttt{quarter\_turn\_left}(\texttt{img}) { 
    \texttt{return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;} 
}
Combining using \textit{stack}

\texttt{\textbf{Op} Rune 1 Rune 2}

\texttt{show(stack(sail\_bb, nova\_bb))}

\begin{itemize}
  \item \texttt{sail\_bb}
  \item \texttt{ nova\_bb}
  \item \texttt{\textbf{Result}}
\end{itemize}
Hands on

How do we do this?
(Put one rune beside another)

sail_bb  nova_bb  Result

Use quarter_turn_left, quarter_turn_right, stack!
function **beside**(pic1, pic2) {
    return quarter_turn_left(stack(quarter_turn_right(pic1),
                                 quarter_turn_right(pic2)));
}
show(beside(pic1, pic2));
Multiple Stacking

\( \text{show} \left( \text{stack} \left( \text{sail}\_bb, \text{stack} \left( \text{heart}\_bb, \text{nova}\_bb \right) \right) \right) \)

\( \text{Result} \)

\( \text{Op} \hspace{1cm} \text{Rune 1} \hspace{1cm} \text{Rune 2} \)

\( \text{sail}\_bb \)

\( \text{heart}\_bb \)

\( \text{nova}\_bb \)
Multiple Beside

Same as multiple stack!
New Pattern

function make_cross(pic) {
    return stack(beside(quarter_turn_right(pic),
                        rotate180(pic)),
               beside(pic,
                       quarter_turn_left(pic)));
}
New Pattern

show(make_cross(rcross_bb))

show(make_cross(nova_bb))
Repeating Patterns

```
show(make_cross(make_cross(nova_bb)))
```

OR

```
var myPic = make_cross(nova_bb)
show(make_cross(myPic))
```
Repeating Patterns

What if we want more than just repeating once?

Recursion / Iteration!
Hands on

• `repeat_pattern(n, pat, rune)`
  
  • `n = Number of times to apply pattern`
  
  • `pat = pattern to repeat`
  
  • `rune = image to apply pattern on`
Repeating Patterns

Recursive solution

function repeat_pattern(n, pat, rune) {
    if (n == 0) {
        return rune;
    } else {
        return pat(repeat_pattern(n-1, pat, rune));
    }
}
Iterative solution

function repeat_pattern(n, pat, pic) {
    var result = pic;
    while (n > 0) {
        result = pat(result);
        n = n-1;
    }
    return result;
}
Recall...

\[
\text{show}(\text{stack}(\text{sail}_{bb}, \text{stack}(\text{heart}_{bb}, \text{nova}_{bb})))
\]

We want equal splitting for all rows!

New primitive: \textit{stack\_frac}
stack_frac

show(stack_frac(1/4, sail_bb, nova_bb))
Stack 3 rows evenly

\[
\text{show}(\text{stack}_\frac{1}{3}, \ \text{rcross}_\text{bb}, 
\text{stack}((\text{rcross}_\text{bb}, \ \text{rcross}_\text{bb})))
\]
Hands on

• stackn(n, rune)

  • n = Number of times to stack
  • rune = image to stack

• Each rune height is 1/n of the entire height
Stack n rows evenly

Recursive solution

function stackn(n, rune) {
    if (n == 1) {
        return rune;
    } else {
        return stack_frac(1/n,
                         rune,
                         stackn(n-1, rune));
    }
}
Stack n rows evenly

Iterative solution

function stackn(n, rune) {
    var result = rune;
    var current_frac = 1;
    while (current_frac <= n) {
        result = stack_frac(1/current_frac, rune, result);
        current_frac = current_frac + 1;
    }
    return result;
}
Functional Abstraction

No idea how a picture is represented
Functional Abstraction

No idea how the operations do their work
Functional Abstraction

Yet we can build complex pictures
That’s it?

Nope!

More cool stuff in next session
Homework

- Refer to handout for quick reference

- Download the Runes zipped folder from Dropbox (refer to slides behind for the following functions)
  - \texttt{mosaic(rune1, rune2, rune3, rune4)}
  - \texttt{simple\_fractal(rune)}
  - \texttt{fractal(rune, n)}

- Email me your code in “.txt” format by \textbf{Sunday 15\textsuperscript{th} Mar 2014}
Instructions on doing HW

• Don’t touch anything except ctc.js

• Fill in the functions in ctc.js (Using notepad)

• Save your changes

• Open ctc.html using Firefox / Refresh after changes

• Open Firefox console

• Test!
Reminder

No session next week (19\textsuperscript{th} March)

Next session will be 26\textsuperscript{th} March (Wed)
mosaic(rune1, rune2, rune3, rune4)
mosaic(rcross_bb, sail_bb, corner_bb, nova_bb)
mosaic(rcross_bb, sail_bb, corner_bb, nova_bb)
simple_fractal(rune)
simple_fractal(make_cross(rcross_bb))
simple_fractal(make_cross(rcross_bb))
fractal(rune, 3)
fractal(rune, 3)

rune

1/2

1/(2^2)

fractal(rune, 2)

fractal(rune, 2)
fractal(rune, 4)
fractal(rune, n)

fractal(rune, n-1)

1/2

fractal(rune, n-1)

rune

1/2

1/2

1/2

1/2
fractal(make_cross(rcross_bb), 2)

Rune

fractal(rune, 2)

n

simple_fractal(rune) = fractal(rune, 2)
Rune

fractal(make_cross(rcross_bb), 3)
Computational Thinking

Temasek Junior College

Davin Choo
cxjdavin+CTC@gmail.com
Welcome back!

- Hope you enjoyed your holidays!
  - Any interesting stories to share with the class? ;)
- What have we done so far?
- What will we be doing next?
Modelling

- Basic graph model
- Modelling real life problems into numbers
Problem solving techniques

- Iteration
- Recursion
- Abstraction / Decomposition of large problems
Technical knowledge

• How to read and write
  • Pseudocode
  • Basic Javascript
• What’s “testing”, and how to do it
• Data structure: Array
Specific algorithms

• Factorial
• Swap
• Find max
• Binary Search
Recap Learning Objectives

• When given a problem, know how to get started

• “So what did you learn?”

• Life is complex

  • Manage complexity!

• Basic programming skills to implement solutions

• *Programming is the language of the future*
Ready for next term?

• Compulsory
  • Basic sorting

• Topics by voting
  • Vote later today
Today’s Plans

• Finish up on Runes
• Share some technical interview questions
• Vote on topics
  • Starts after sorting
Homework

• Refer to handout for quick reference

• Download the Runes zipped folder from Dropbox (refer to slides behind for the following functions)
  
  • mosaic(rune1, rune2, rune3, rune4)
  
  • simple_fractal(rune)
  
  • fractal(rune, n)

• Email me your code in “.txt” format by Sunday 15\textsuperscript{th} Mar 2014
mosaic(rcross_bb, sail_bb, corner_bb, nova_bb)
fractal(rune, n)

rune

fractal(rune, n-1)
fractal(rune, n-1)

1/2
1/2
1/2
1/2
Persian rug / carpets

Persian carpet

From Wikipedia, the free encyclopedia

The Persian carpet or Persian rug (Middle Persian: بُد, [1] Persian: فرش, meaning "to spread"; sometimes فرش qalîh) [2] is an essential part of Persian art and culture. Carpet-weaving is undoubtedly one of the most distinguished manifestations of Persian culture and art, and dates back to ancient Persia. In 2008, Iran's exports of hand-woven carpets was $420 million or 30% of the world's market. [3][4] There is an estimated population of 1.2 million weavers in Iran producing carpets for domestic markets and international export. [5] Iran exports carpets to more than 100 countries, as hand-woven rugs are one of its main non-oil export items. The country produces about five million square metres of carpets annually—80 percent of which are sold in international markets. [6] In recent times Iranian carpets have come under fierce competition from other countries producing reproductions of the original Iranian designs as well as cheaper substitutes. [6]

The designs of Persian carpets are copied by weavers from other countries as well. Iran is also the world's largest producer and exporter of handmade carpets, producing three quarters of the world's total output. [7][8][9] Though in recent times, this ancient tradition has come under stiff competition from machine-made products. [10] Iran is also the maker of the largest handmade carpet in history, measuring 60,546 square feet (5,624.0 square meters). [11][12][13]

Persian carpets can be divided into three groups; Farsh / Qali (sized anything greater than 6x4 feet), Qalicheh (قلمچه, meaning "small rug", sized 6x4 feet and smaller), and nomadic carpets known as Gelim (گلیم) including Sizil, meaning "rough carpet". [2] In this use, Gelim includes both pile rugs and flat weaves (such as kilim and soumak).

Contents

1 History
  1.1 Early History
  1.2 Islamic period
  1.3 Modern period
2 Materials
3 Designs, motifs, and patterns
4 Design
  4.1 Layout
  4.2 Motifs
5 Techniques and structures

The Rothchild Small Silk Medallion Carpet, mid-16th century, Museum of Islamic Art, Doha (enlarge image to see detail)
How do we build this?

Apply what you have learnt!

persian(rune, n)

n = 5

Break it down into smaller parts that you know how to solve!
What small part do you know how to solve?

make_cross

stacking: stack_frac, stackn

rotating: quarter_turn_left, quarter_turn_right
Decomposition the problem

How do we split this?
Decomposition #1

How do we split this?

What do we use here?

How do we split this?
Decomposition #2

How do we split this?

What do we use here?

How do we split this?
Creating 3D Objects

- Use greyscale to represent depth
- Surface = Black
  Maximum depth = White
- Closer to you = Blacker
  Further from you = Whiter
Creating 3D Objects

means

1/2
Creating 3D Objects

• overlay(sail_bb, heart_bb)
Of course there’s overlay_frac
Creating 3D Objects

- overlay_frac(1/3, sail_bb, heart_bb)
Other cool stuff you can do

• **anaglyph** E.g. anaglyph(sail_bb)

• **stereogram** E.g. stereogram(sail_bb)

• **hollusion** E.g. hollusion(sail_bb)
Challenge

• What cool pictures can you make using the available runes and functions?

• Refer to CS1101S AY2013/2014 Rune Contest slides
Technical interviews

• Interview where tech companies test interviewees on how “zai” they are. E.g. Google, Microsoft, etc.

• Quite interesting sometimes. Like brainteasers.

• Usually can learn some interesting stuff from the questions

• Now let’s look at some actual technical interview questions
Swapping without temp

• Recall your swapping code

```javascript
swap(a,b) {
    var temp = a;
    a = b;
    b = temp;
}
```

• How to swap without creating/using a temporary variable?
Swapping without temp

• Solution #1 (Addition and subtraction)

```plaintext
swap(a,b) {
    a = a+b; // a now holds a+b
    b = a-b; // b now holds a
    a = a-b; // a now holds b
}
```
Swapping without temp

- Solution #2 (XOR)

- What is XOR?
  - Exclusive-OR
  - 0 XOR 1 = 1 XOR 0 = 1
  - 0 XOR 0 = 1 XOR 1 = 0
Swapping without temp

• Solution #2 (XOR)

```c
swap(a, b) {
    a = a XOR b; // a now holds a XOR b
    b = a XOR b; // b now holds a
    a = a XOR b; // a now holds b
}
```

• Same idea, replace +, - with XOR
Quicksort

- “Write quicksort” on a whiteboard (cannot test)
- Actual interview question from Microsoft
  - One of my professor used to work in Microsoft
  - This was one often common interview questions
- What’s this “quicksort”?
  - A sorting algorithm
    (not basic, but it’s quick. Can guess from name right?)
Find median in 2 sorted arrays

• Recall binary search

• Given sorted array is sorted, find element X

• Median (vs. Mean vs. Mode)

  • “Center” element. Left side size = Right side size

• Now:

  • Given 2 sorted arrays find middle element

3 types of “average”
Find median in 2 sorted arrays

• Given 2 sorted arrays find middle element of all elements

• Easy (but slow) solution
  • Combine both array into new array `newArr`
  • Sort
  • Return `newArr[size/2]`

• Apply binary search simultaneously on both arrays!
Why am I sharing these?

• Simple mathematical properties like XOR are surprisingly powerful

• Cryptography uses a lot of XORs…

• You are actually learning actual useful stuff here

• Know how to make use and combine things you have learnt (This applies to everything in life!)
Basic Sorting Topics

• Bubble sort
• Insertion sort
• Selection sort
• Merge sort
Vote on Topics

More general stuff

• Advanced sorting
• Data structures
• Algorithm analysis
• Graph algorithms
• Dynamic programming

More specific things:

• Basic cryptography
• Minimax algorithm
• Project Euler
• Computer Organization
• Build a program/game

Suggestions?
Advanced sorting

- Heap sort
- Quick sort
- Counting sort
- Radix sort
- Bucket sort

Needs a little data structure

Heaps

Hash tables
Data structures

• Heaps (implementation in arrays)
• Trees (balanced trees? augmented trees?)
• Hash tables
• Linked lists
• Stack/Queue
• Priority Queues
Algorithm analysis

• Remember how we said “quicksort is quick” and easy approach earlier was “slow”?

• How to measure how fast/good an algorithm is?

• Space complexity, Time complexity

• Big-O notation
Graph algorithms

• Examples you know of:
  Pancake flipping, shortest path algorithm

• Other examples: BFS, DFS, Topological Sort, etc.

• Applications (in Artificial Intelligence):
  A* Search (general form of Dijkstra’s)
  Minimax algorithm
Dynamic Programming

- Another class of useful program solving skill
  - Besides recursion/divide-and-conquer
  - In fact, one of the most powerful ones…
- But not trivial to understand
- Idea: Similar to recursion but don’t repeat computation
- Example: Fibonacci
Basic cryptography

• Shift ciphers
• Affine ciphers
• Block ciphers (?)
• RSA (?)
• Program your own encoding/decoding code

Good way to show you why programming is useful (no one do these by hand)
Minimax algorithm

- Best way to play a game / decide action
- Used in basic A.I. programs
- Example: Tic-tac-toe
Project Euler

- http://projecteuler.net

About Project Euler

What is Project Euler?

Project Euler is a series of challenging mathematical/computer programming problems that will require more than just mathematical insights to solve. Although mathematics will help you arrive at elegant and efficient methods, the use of a computer and programming skills will be required to solve most problems.

The motivation for starting Project Euler, and its continuation, is to provide a platform for the inquiring mind to delve into unfamiliar areas and learn new concepts in a fun and recreational context.

Who are the problems aimed at?

The intended audience include students for whom the basic curriculum is not feeding their hunger to learn, adults whose background was not primarily mathematics but had an interest in things mathematical, and professionals who want to keep their problem solving and mathematics on the edge.

- We’ll work on the questions together one by one
Computer Organisation

- Curious about internal workings of a computer?
- How do computers represent numbers?
- How do computers do calculations?
- What are logic gates?
- Etc etc…
Build a program/game

• Build something (Ever had a cool idea?)

• Learn skills along the way to accomplish goal

• I can suggest topics or you can propose

• Can be group work
Vote on Topics

More general stuff
• Advanced sorting
• Data structures
• Algorithm analysis
• Graph algorithms
• Dynamic programming

More specific things:
• Basic cryptography
• Minimax algorithm
• Project Euler
• Computer Organisation
• Build a program/game

Suggestions?
Computational Thinking
Temasek Junior College
Davin Choo
cxjdavin+CTC@gmail.com
Vote on Topics

More general stuff
- Advanced sorting
- Data structures
- Algorithm analysis
- Graph algorithms
- Dynamic programming

More specific things:
- Basic cryptography
- Minimax algorithm
- Project Euler
- Computer Organisation
- Build a program/game

4/5 votes!

Suggestions?
Pacman
But first...

Sorts
Sorts

Bubble sort
Insertion sort
Selection sort
Merge sort
Seen this before?

Uniform Group (NPCC) Sizing

https://www.youtube.com/watch?v=gdVPxQJJKo
NPCC Sizing

• To make contingent look “nicer”

• How it works
  • Line up in descending order
  • Alternate numbering between 1 and 2
  • “1” step forward; “2” step backwards
  • March and form up
Sorting

• Line up in descending order

• What if we have 10000 people?

• Abstract into a model
  • Input: A sequence of numbers
  • Output: Sequence in descending order
Models

UNO Cards

NPCC Sizing

Sorted Hand

Sequence of Numbers

People in order

Descending Sequence

Sort

???
All of you stand up and gather to the front

Arrange in descending height order

Arrange in earliest birthday order

Arrange in alphabetical name order
Hands on

• How many people did you compare with? (Roughly)

• That means ____ comparisons in total…

• Lower # comparisons better (Algorithm analysis)
UNO Cards

Let us use UNO cards as an abstraction/example

Consider [4, 5, 1, 2, 9, 6, 8]
Bubble sort

Idea: Iterative!

1. Look from left to right
   - If it is larger than the card on its left, swap
   - Stop when we reach last card

2. If we made a swap, repeat Step 1.

We already know how!
Bubble sort

• Why does it work?

• If already sorted?
  • ___ comparisons
  • ___ swaps

• If reversed order initially?
  • ___ comparisons
  • ___ swaps
Insertion sort

Idea: Iterative!

1. Look from left to right
   - If it is larger than the card on it’s left, swap
   - Keep swapping until left card is larger or same
   - Stop when we reach last card

We already know how!
Insertion sort

• Why does it work?

• If already sorted?
  • ___ comparisons
  • ___ swaps

• If reversed order initially?
  • ___ comparisons
  • ___ swaps
Selection sort

Idea: Iterative!

1. Look at cards, find maximum
2. Put maximum at left most side
3. Look at the rest of the cards (Repeat 1)

We already know how!
By swapping
Selection sort

- Why does it work?
- If already sorted?
  - ___ comparisons
  - ___ swaps
- If reversed order initially?
  - ___ comparisons
  - ___ swaps
Merge sort

Idea: Recursive!

1. Split cards into K portions (usually K=2)

2. Sort each portion By using any sort

3. Combine sorted portions ("Merge step")

How?
Merge

Sorted

R > B

Sorted
Merge

Sorted

Sorted

R > B
Merge

Sorted

R < B

Sorted
Merge

Sorted

R < B

Sorted
Merge sort

• Why does it work?

• If already sorted?
  • ___ comparisons
  • ___ swaps

• If reversed order initially?
  • ___ comparisons
  • ___ swaps
Loop invariant

• A property that is guaranteed before and after every iteration of a loop

• Example:

  • Terminating conditions in a loop
Loop invariant

- Bubble sort
  - Every time we are done from L to R, cards are 1 position closer to correct position

- Insertion sort
  - Left portion of cards always in sorted order

- Selection sort
  - After K steps, K largest cards are in correct place
Parallelism

Idea

- Multi-core / multiple processors
- Split workload and combine results
- Can apply to sorts we learnt so far?
Parallelism

• Bubble sort
  • Check alternate cards at same time
• Insertion sort & Selection sort
  • Can’t. Why?
• Merge sort
  • Perfect! One processor sort 1 portion
Performance

- Algorithmic analysis
  - Worst case performance
  - Best case performance
- In practice…
  - Small N: Insertion sort
  - Large N: Merge sort, until small N, then insertion
Computational Thinking
Temasek Junior College
Davin Choo
cxjdavin+CTC@gmail.com
What are these?

- HTML5
  - Hyper Text Markup Language
  - Standard markup language used to create web pages
- CSS
  - Cascading Style Sheets
  - Used for describing look and formatting in markup languages
- JS
  - Javascript
Goals for today

• Basic HTML5 syntax
• How to draw using HTML5 <canvas>
• How to capture keyboard inputs
• How to simulate motion
• Some game logic
Basic HTML5

- Tags
- Canvas
- Demo (basic.html)
- Demo (canvas_demo.html)
Events

- Capturing keyboard inputs
- Demo (keycapture.html)
JS Timing Events

- setInterval()
- setTimeout()
- Demo (motion.html)
Other stuff

• A lot of other HTML5 tags
• Alerts
• Nameless functions
• for-loop
• random()
Basic Game Logic

- Game state
  - Maintain
  - Draw

- Visual
  - See

- User input
  - Process and Update
  - Interact

Game

User
Basic Game Logic

• Maintain
  • Update self-moving things / A.I. motion

• Draw
  • Remember to clear canvas before re-drawing

• Process and update
  • Process keyboard presses / button clicks
2-player catching

- Maintain (Do nothing, until keyboard input detected)
- Draw
  - Clear canvas. Draw player images at player positions.
- Process and update
  - Process keyboard presses
  - Update players’ position accordingly
  - If collide, add point to catcher
2048

- Maintain (Do nothing, until keyboard input detected)
- Draw
  - Clear canvas. Draw cells.
- Process and update
  - Process keyboard presses
  - Update array values (Shift/Combine)
  - Spawn new cell (Either 2 or 4) at random location
Useful links

Tutorials:

http://www.w3schools.com/html/html5_intro.asp

http://www.w3schools.com/html/html5_canvas.asp

http://www.w3schools.com/js/js_timing.asp

Key codes (to capture keyboard input):

http://www.cambiaresearch.com/articles/15/javascript-char-codes-key-codes