CS16 Section 1

Week 2
Ice Breaker!

Penguins are sea creatures?
Welcome to Section and CS16

- Section is a great place to learn and reinforce material!
  - Some weeks we will have labs instead of sections
  - Most weeks, we have a few different options of what to talk about, lecture review, project preparation, or course logistics, and will try to go over the things that y’all want to go over

- Expectations:
  - Students will NC the course if they miss more than 3 sections
  - To switch, email TAs **before** your section time (at least 24 hours)
  - Bring mini assignments to all sections
  - Participate and help us help you
Other resources and Mentorship

- Mentorship meetings - meet one-on-one with TAs
  - This year, we’re opening up the option for students to find mentors that they might find more closely related to their interests and or identities.
    https://docs.google.com/forms/d/e/1FAIpQLSelbHf-xtl4-dVRxelBAviWvJ1nxMHVWyufAWeK-miV1zcA/viewform?usp=sf_link
    - Due: 05/28/21 at 11:59PM EST
  - **Pros for defaulting: We highly suggest defaulting to your section TAs.** Students in the past have told us they felt more comfortable developing a relationship with their section TAs, whereas meeting a TA with whom they have no experience with can be very impersonal. If you choose to go with one of the section TAs, you don’t need to fill out this form!
  - **Pros for opting-in:** If you have a very specific interest or question about CS in the dept. that you believe can only be answered by certain TAs, this might be better for you.
  - **FORM DUE DATE: UPCOMING FRIDAY AT 11:59PM.**

- Debugging Hours
- Ed Discussion
- Clinic hours - work together with other students on problems
- Conceptual hours - quicker group TA hours to learn concepts
Community Guideline

● Some Ideas:
  ○ Don’t interrupt people
  ○ Be mindful of how your participation (speaking, body language) might affect others in the section
    ■ If a TA asks to move on, please be respectful of this!
  ○ Be respectful of your TAs and fellow students
  ○ Feel free to reach out to us with any concerns or problems privately

● Any Suggestions:
  ○
Mini Assignment

Modulo problems:

1. $60 \% 6 = 0$

2. $368 \% 13 = 4$

3. $4901 \% 172 = 85$
Mini Assignment

Python

1. Parentheses and semicolon
2. Prints you lost
   a. Need decimal division
3. 34550
   a. Make them ints instead of strings
4. 
   a. Indentation
   b. for else doesn’t exist
   c. need to use range
   d. use “None” not “null”
Topics for today

- **Testing**
- **Gradescope**
- **Seamcarve**
- **Python**
- **Big - O**
Testing

- What are some edge cases for each scenario?
  - You have an algorithm that takes in an array of numbers and outputs a new number
  - You have an algorithm that compare if two arrays are equal
  - You have a sorting algorithm
Gradescope

Now we will go through an example of submitting your code and test suite on Gradescope!
Seamcarve - findLowestCostSeam ()

- You have to implement the method to find the lowest cost seam
  - 1. Calculate the importance value for each pixel
      - Support Code Methods: getPicHeight(), getPicWidth(), getPixelColor(row, col)
  - 2. Find the lowest cost of each vertical seam
     - Dynamic Programming: How can we break the problem into smaller problems? (Lecture topic for 5/25)
     - In what order are we iterating through the image?
     - What 2 things do we need to track for every pixel?
Seamcarve - findLowestCostSeam ()

- You have to implement the method to find the lowest cost seam
  - 1. Calculate the importance value for each pixel
     - Support Code Methods: getPicHeight(), getPicWidth(), getPixelColor(row, col)
  - 2. Find the lowest cost of each vertical seam
     - Dynamic Programming: How can we break the problem into smaller problems?
     - In what order are we iterating through the image
     - What 2 things do we need to track for every pixel

<table>
<thead>
<tr>
<th>Importance:</th>
<th>Cost:</th>
<th>Dir</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 6 5 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 4 3 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 8 2 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Seamcarve - findLowestCostSeam ()

- You have to implement the method to find the lowest cost seam
  - 1. Calculate the importance value for each pixel
     - Support Code Methods: getPicHeight(), getPicWidth(), getPixelColor(row, col)
  - 2. Find the lowest cost of each vertical seam
     - Dynamic Programming: How can we break the problem into smaller problems?
     - In what order are we iterating through the image
     - What 2 things do we need to track for every pixel

<table>
<thead>
<tr>
<th>Importance</th>
<th>Cost</th>
<th>Dir</th>
</tr>
</thead>
<tbody>
<tr>
<td>3  6  5  3</td>
<td>9 11 8  6</td>
<td>1  1  1  0</td>
</tr>
<tr>
<td>1  4  3  1</td>
<td>7  6  5  3</td>
<td>0  1  0  -1</td>
</tr>
<tr>
<td>6  8  2  4</td>
<td>6  8  2  4</td>
<td></td>
</tr>
</tbody>
</table>
Seamcarve - findLowestCostSeam ()

- You have to implement the method to find the lowest cost seam
  - 3. Find the lowest cost Seam
    - Where do we store the final costs?
    - How can we follow the path?

<table>
<thead>
<tr>
<th>Importance:</th>
<th>Cost:</th>
<th>Dir</th>
<th>Lowest Seam</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 6 5 3</td>
<td>9 11 8 6</td>
<td>1 1 1 0</td>
<td>[3, 3, 2]</td>
</tr>
<tr>
<td>1 4 3 1</td>
<td>7 6 5 3</td>
<td>0 1 0 -1</td>
<td></td>
</tr>
<tr>
<td>6 8 2 4</td>
<td>6 8 2 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Seam carve Pseudocode

function find_least_important_seam(vals):
dirs = 2D array with same dimensions as vals
costs = 2D array with same dimensions as vals
costs[height-1] = vals[height-1] // initialize bottom row of costs

for row from height-2 to 0:
    for col from 0 to width-1:
        costs[row][col] = vals[row][col] +
        min(costs[row+1][col-1], costs[row+1][col], costs[row+1][col+1])
        dirs[row][col] = -1, 0, or 1 // depending on min

// Find least important start pixel
min_col = argmin(costs[0]) // Returns index of min in top row

// Create vertical seam of size ‘height’ by tracing from top
seam = []
seam[0] = min_col
for row from 0 to height-2:
    seam[row+1] = seam[row] + dirs[row][seam[row]]
return seam
Python

Java

```java
public void sayHello(String name) {
    System.out.println("Hello " + name + ":");
}
```

Python

```python
def say_hello(name):
    print("Hello " + name + ":")
```
More Python

Notable syntax changes:

- Indentation is crucial to nest code, instead of brackets
- You do not need to specify the type of variables/parameters/etc.
- `or`, `and`, `not` instead of `||`, `&&`, `!`
- `'''` for block comments instead of `/* */`
- `#` instead of `//` for inline comments

Python has many useful idioms-- you will slowly learn ‘Pythonic’ ways of doing things with built-in functions like `list()`, `range()`, `dictionaries`, and doing complicated things all in one line!

Refer to Python Lab Part 1 for a helpful table of syntax translations!
Finding Big-O Runtime

Problem 1: What is the runtime of this function?

```python
run_race(x, y, z):
    for i in range 0 to x:
        for j in range 0 to y:
            print "LAP COMPLETED"
    return 3*z
```

Problem 2

write a function `sum_list` that takes in a list and returned the sum of all elements in the list. What is the runtime of this function.
Big-O Proof

Prove that \( f(n) = n^2 + 5n + 7 \) is \( O(n^2) \)

**Definition (Big-O):** \( T_A(n) \) is \( O(T_B(n)) \) if there exists positive constants \( c \) and \( n_0 \) such that:
\[
T_A(n) \leq c \cdot T_B(n)
\]
for all \( n \geq n_0 \)