Graphics
Chris Piech and Mehran Sahami
CS106A, Stanford University
Learn by Doing

Assignment 2

Number of Students

Time Spent (hours)

Piech + Sahami, CS106A, Stanford University
Task

Almost a hundred thousand unique solutions
US K-12 Students

= 500,000 learners
Code.org Students

= 500,000 learners
Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.
Each node is a unique partial solution

Each edge is what a teacher suggested

Pink dots are students.

Solution

Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.
Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.
Desirable Path Algorithm

Poisson Common Path

\[ \gamma(s) = \arg \min_{p \in Z(s)} \sum_{x \in p} \frac{1}{\lambda x} \]

- Predicted next partial solution
- Paths to solution
- Partial solutions in the path
- Path Cost
- Submission count of partial solution
Deep Learning Algorithms

1. 

![Diagram](image1.png)

2. 

![Diagram](image2.png)

Benchmark AUC

- Marginal
- BKT
- BKT*
- DKT

Program → \( \mathbb{R}^n \)

Recall vs. Precision

- Neural Network
- Poisson Common Path
- AST
- Edit Distance
Deep Learning on Trajectories

Research in collaboration with Lisa Wang
Predicts Future Success

- Problem A: $y = 3.0x + 61.2$, $R^2 = 0.98$
- Problem B: $y = 3.8x + 58.1$, $R^2 = 0.97$
Predicts Future

Recall on Prediction of Future

- Baseline
- Logistic Regression
- Deep Learning

Piech + Sahami, CS106A, Stanford University
Highly Rates Grit

1. Two compound errors
2. Solves first error
3. Starts reasonable attempt

4. Completes attempt
5. Backtracks
6. Finds solution
Review
What are the basic operations needed to create a sepia tone? My reference point is the perl imagemagick library, so I can easily use any basic operation. I've tried to quantize (making it grayscale), colorize, and then enhance the image but it's still a bit blurry.

The algorithm comes from [this page](https://stackoverflow.com/questions/1061093/how-is-a-sepia-tone-created), each input pixel color is transformed in the following way:

```
outputRed = (inputRed = .393) + (inputGreen = .769) + (inputBlue = .189)
outputGreen = (inputRed = .349) + (inputGreen = .686) + (inputBlue = .168)
outputBlue = (inputRed = .272) + (inputGreen = .534) + (inputBlue = .114)
```

If any of these output values is greater than 255, you simply set it to 255. These specific values are the values for sepia tone that are recommended by Microsoft.
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for pixel in image:
        sephia_pixel(pixel)
    image.show()

def sephia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for y in range(image.height):
        for x in range(image.width):
            pixel = image.get_pixel(x, y)
            sepia_pixel(pixel)
    image.show()

def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for y in range(image.height):
        for x in range(image.width):
            pixel = image.get_pixel(x, y)
            sepia_pixel(pixel)
    image.show()

def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
def main():

    for y in range(600):
        for x in range(800):
            print(x, y)
End Review
Today’s Goal

1. How do I draw shapes?
Graphics Programs
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```
the following `main` method displays a blue square

def main():
    canvas = make_canvas(800, 200, 'Hello Rect')

800 pixels wide
the following `main` method displays a blue square

def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```

**Draw a Rectangle**
the following **main** method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```
Draw a Rectangle

the following **main** method displays a blue square

def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```

**Aside: Named Arguments**
This argument is named as `filled`. It allows functions to have arguments which you can ignore if you want a default value.
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100)
```

Aside: Named Arguments
This argument is named as `filled`. It allows functions to have arguments which you can ignore if you want a default value.
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```

**Aside: Named Arguments**
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.
the following `main` method displays a blue square

```python
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```
TK Natural Graphics
Rectangles, Ovals, Text

Programming is Awesome!
- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text()`
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

The first point of the line is \((x1, y1)\).
• `canvas.create_line(x1, y1, x2, y2)`

• `canvas.create_oval()`

• `canvas.create_text()`

The second point of the line is \((x2, y2)\)
- `canvas.create_line(x1, y1, x2, y2)`

- `canvas.create_oval()`

- `canvas.create_text()`
• `canvas.create_line()`

• `canvas.create_oval(x1, y1, x2, y2)`

• `canvas.create_text()`
- `canvas.create_line()`

- `canvas.create_oval(x1, y1, x2, y2)`

- `canvas.create_text()` (x1, y1)

- `(x2, y2)`
- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text(x, y, text='hi')`
- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text(x, y, text='hi', anchor='w')`
Pedagogy
Goal
Milestone 2
Teaser for next week...
def draw_square(canvas, row, col):

    Hold up! If you get a copy when you pass a parameter. Does this copy the canvas??!!

Large variables are stored using something like a URL. The URL gets copied
How do you share google docs?

https://docs.google.com/document/d/1eBtnEill3KHe
fFS-kSAOpXqeSXpbfTTMImOgj6l9dvk/
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
```python
def main():
    canvas = make_canvas(...)”
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
```python
def main()
    canvas = make_canvas(...)  
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

Diagram:
- **Stack**:
  - `main`
    - `canvas` with a `42` value
  - `draw_square`
    - `canvas` with a `42` value

- **Heap**:
  - Window with a square and a `42` value
```python
def main():
    canvas = make_canvas(...)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

![Diagram of stack and heap]
```python
def main():
    canvas = make_canvas(
        draw_square(canvas)
    )

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

---

**Stack**

- **main**
  - **canvas**
    - 42

**Heap**

- 42
  - 42

---

Piech + Sahami, CS106A, Stanford University
Large variable types are stored as memory addresses

(which are like memory URLs)