COMP 204
Review and final exam preparation

Yue Li
Outline

Seed filling algorithm (revisit)

Final exam (April 30 6:30 PM)

In-class practice questions
front: [(0,0)]
front: [(0,1), (1,0), (1,1)]

Filled color
Bckg color
Edge color
(0,0)(0,1)
(1,0)(1,1)
front: [(0,1), (1,0), (1,1)]
front: \[(1,0), (1,1), (0,2)\]
front: [(1,1), (0,2), (2,0), (2,1)]
front: [(0,2), (2,0), (2,1)]
front: \[ (2,0), (2,1), (0,3) \]
front: [(10,14)]
- We just came back from the seed fill function!
- The “cell” size is the number of the pixels outside of the cell (too big as a “cell”)
- The next pixel to go: $i = 0$, $j = 1$
- But pixel $(0, 1)$ is already filled with color (i.e., no longer background)
- We keep going until we see another background pixel: $(2, 3)$
front: \([(2,3)]\)
front: [(2,4), (3,3), (3,4)]
front: [(3,3), (3,4), (2,5), (3,5)]
front: [(3, 4), (2, 5), (3, 5), (4, 2), (4, 3), (4, 4)]
def seedfill(im, seed_row, seed_col, fill_color, bckg):
    
    """im: The image on which to perform the seedfill algorithm
    seed_row and seed_col: position of the seed pixel
    fill_color: Color for the fill
    bckg: Color of the background, to be filled
    Returns: Nothing
    Behavior: Modifies image by performing seedfill
    """

    size = 0  # keep track of patch size
    n_row, n_col, foo = im.shape
    front = [(seed_row, seed_col)]  # initial front

    while len(front) > 0:
        r, c = front.pop(0)  # remove 1st element of front
        if np.array_equal(im[r, c, :], bckg):
            im[r, c] = fill_color  # color pixel
            size += 1
        # look at all neighbors
        for i in range(max(0, r - 1), min(n_row, r + 2)):
            for j in range(max(0, c - 1), min(n_col, c + 2)):
                if np.array_equal(im[i, j, :], bckg) and (i, j) not in front:
                    front.append((i, j))

    return size
Seeding from all possible starting pixel...

```python
min_cell_size=100  # based on prior knowledge
max_cell_size=300  # based on prior knowledge
n_cells=0

# look for a black pixel to seed the filling
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if np.array_equal(edge[i,j,:],(0,0,0)):
            rand_color = (random.randrange(255),
                           random.randrange(255),
                           random.randrange(255))
            size=seedfill_with_animation(edge, i ,j,
                                         rand_color, (0,0,0) )
            if size>= min_cell_size and
               size<max_cell_size:
                n_cells+=1
        print("Number of cells:",n_cells)
```

17 / 36
Outline

Seed filling algorithm (revisit)

Final exam (April 30 6:30 PM)

In-class practice questions
Final exam info

- Date: April 30, 6:30-9:30 PM; Location: TBD
- Weight: 35% of your final grade (or 55% if better than midterm grade for students who opted the second non-programming midterm assignment option)
- Closed book but 8.5 x 11 double-sided crib sheet allowed.
- Questions:
  - 9 multiple choice questions (total 27%). Answer on Scantron (not on exam). Follow instructions for each questions: For some questions you need to indicate the only ONE correct answer. For other questions you need to indicate ALL correct answers.
  - Answer the rest of the questions directly on exam
  - 8 short answer questions (7 questions each worth 4 points and 1 question worth 5 points) (total: 33%).
  - 1 bonus short answer question worth 5 point.
  - 4 long answer questions (10 point per question; total: 40%).
Final exam content

Main materials that are covered in the final exam include:

- Basics: functions, loops, variables, data types (string, list, tuple, dictionary, sets), difference between pass by copy and pass by memory addresses
- Algorithms: Searching (linear and binary search) and sorting (insertion and selection sort)
- Pattern searching by string indexing and regular expression (simple ones)
- Object oriented programming: class, attributes, class inheritance, class methods
- BioPython sequence handling covered in class (I will remind you what the methods are in the exam)
- Machine learning: know what supervised, unsupervised, reinforcement learning are, problems they can solve, TPR, FPR, overfitting, cross-validation, ROC, decision trees
- Image processing: basic understanding of going from a pixel in the image to numpy ndarray
- What to memorize? Nothing. Use cribsheet to note the
Preparing for the final exam

How best to prepare for the exam:

▶ Practice, practice, practice.
▶ Review all lecture notes, assignment solutions, midterm solutions
▶ Practice on the problems we’ve posted on MyCourses-Content
▶ Attend CSUS review session
▶ Come to my office hours:
  ▶ Wednesday: 11:30-12:30
Outline

Seed filling algorithm (revisit)

Final exam (April 30 6:30 PM)

In-class practice questions
What prints out?

```python
def myfun(x, y):
    x = x + 1
    y = y + 1
    return x + y

x = 0
y = 1
z = myfun( myfun(x,y), x)
print(z)
```
Functions (pass by memory address)

What prints out?

```python
def myfun(x, y):
    x[0] = x[0] + 1
    y[0] = y[0] + 1
    return [x[0] + y[0]]

x = [0]
y = [1]
z = myfun( myfun(x,y), x)
print(z)
```
Linear and binary search

How to search number 9 in this list by linear search and binary search? [2,5,7,9,10]
Insertion and selection sort

How to sort the following list by insertion sort and selection sort?
[2,10,5,9,7]
Sequence alignment (A2)

Given match score +3, mismatch score -2, gap score -1. What’s the similarity score between sequence GGC with sequence GTC?
List comprehension

Convert the following for loop into list comprehension with one line of code:

```python
x = []
for i in range(5):
    x.append(-2*i)
```
Object oriented programming: attributes

What are attributes in MyClass

class MyBus:
    def __init__(self, stationID, passengers):
        self.s = stationID
        self.p = passengers
        terminal = 0
Object oriented programming: methods

What prints out?

class Animal:
    def __init__(self, age):
        self.age = 0
        self.claws = 0
    def grow(self):
        self.age += 1
        claws = self.claws + 1

animal = Animal()
animal.grow()
print(animal.age)
print(animal.claws)
class Animal():
    def __init__(self, age):
        self.age = 0
        self.claws = 0
    def grow(self):
        self.age += 1
        claws = self.claws + 1

class Predator(Animal):
    def __init__(self):
        Animal.__init__(self, 0)
        self.horns = 0
        self.eyes = 0
    def grow(self):
        Animal.grow(self)
        self.horns += 1
        eyes = self.eyes + 1

pred = Predator()
pred.grow()
print(pred.claws, pred.age, pred.horns, pred.eyes)
Central dogma

Every three DNA letters (i.e., codon) code for an amino acid
Transcription

Given a DNA string as the template strand say 5’-AGATCAT-3’, write a function called `transcribe(dna)` that returns the transcribed RNA sequence (i.e., AUGAUCU)

```python
def transcribe(dna):
    return rna
```
### Translation: Codon Table

<table>
<thead>
<tr>
<th>1st base</th>
<th>TTT</th>
<th>TTC</th>
<th>TTA</th>
<th>TTG[^A]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Phenylalanine)</td>
<td>TCT</td>
<td>TCC</td>
<td>TCA</td>
<td>TCG</td>
</tr>
<tr>
<td>(Serine)</td>
<td>TAT</td>
<td>TAC</td>
<td>TAA</td>
<td>TAG</td>
</tr>
<tr>
<td>(Alanine)</td>
<td>TAC</td>
<td>TCG</td>
<td>TGA</td>
<td>TGG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2nd base</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>(Tyr/Y) Tyrosine</td>
<td>(His/H) Histidine</td>
<td>(Gln/Q) Glutamine</td>
<td>(Arg/R) Arginine</td>
</tr>
<tr>
<td>C</td>
<td>(Stop/Ochre)[^3]</td>
<td>(Stop/Ampur)[^3]</td>
<td>(Trp/W) Tryptophan</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3rd base</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Cys/C) Cysteine</td>
<td>(Cys/C) Cysteine</td>
<td>(Cys/C) Cysteine</td>
<td>(Cys/C) Cysteine</td>
<td>(Cys/C) Cysteine</td>
</tr>
<tr>
<td>T</td>
<td>(Cys/C) Cysteine</td>
<td>(Cys/C) Cysteine</td>
<td>(Cys/C) Cysteine</td>
<td>(Cys/C) Cysteine</td>
</tr>
</tbody>
</table>

---

**Not all mutation leads to a different amino acid**

e.g., GCT and GCC both code for Alanine
Translation

Assume the codon table is provided to you as a dictionary $ct$ with key as the 3-letter DNA string and value as the amino acid, write a function that translates an RNA into the amino acid sequence

```python
def translate(rna, ct):
    return aa
```
Suppose we obtain a collection of unknown cells from a patient. Each cell is a Cell object. We are provided with a function called `cancer_cell_score(cell)` that give a cancer score to the unknown cell. Write a function that return the highest scoring cell.

```python
def get_most_similar_cell(cancer_cell, unknown_cells):
    
    return ccc  # candidate cancer cell
```