CS1134 Final Exam

Lee;Cindy

TOTAL POINTS

133 / 150

QUESTION 1

1 Runtime 10 / 10

✓ **- 0 pts Correct**

- 5 pts (i) Runtime should be O(n^2)

- 5 pts Code in (ii) is not faster than quadratic time

- 2.5 pts Code does not return the correct string

- 5 pts (ii) Wrong/blank

- 10 pts Skip

- 2 pts Your runtime analysis in (iii) is wrong

- 0 pts Click here to replace this description.

QUESTION 2

2 2-D Array 10 / 10

✓ **- 0 pts Correct**

- 10 pts Wrong: the answer is 3

QUESTION 3

3 Linked List 5 / 10

- 0 pts Correct

- 10 pts Wrong. Answer is self._head._next._next._next._prev=self._head._next

✓ **- 5 pts Partially wrong. Answer is self._head._next._next._next._prev=self._head._next**

- 7.5 pts Partially wrong. Answer is self._head._next._next._next._prev=self._head._next

- 10 pts Blank

- 0 pts Click here to replace this description.

QUESTION 4

4 Heap 10 / 10

✓ **- 0 pts Correct**

- 5 pts Insert is wrong

- 5 pts Extract-min is wrong

- 10 pts Blank

QUESTION 5

5 Balanced BST 10 / 10

✓ **- 0 pts Correct**

- 10 pts Wrong

- 5 pts AVL Rebalance wrong

QUESTION 6

6 Longest common subsequence 10 / 10

✓ **- 0 pts Correct**

- 10 pts Wrong

- 10 pts Skip

- 10 pts Blank

QUESTION 7

7 Selection 10 / 10

✓ **- 0 pts Correct**

- 10 pts Wrong.

- 5 pts If you are looking for the an item that is greater than the pivot, you need to adjust the item you are looking for when you recurse.

- 5 pts You don't show both parameters

- 10 pts Skip

QUESTION 8

8 Suffix Trie 10 / 10

✓ **- 0 pts Correct**

- 5 pts Regular Trie Wrong

- 5 pts Compressed Trie Wrong

- 2.5 pts Regular trie partly wrong

- 2.5 pts Compressed trie partly wrong

- 10 pts Blank

- 10 pts Skip

QUESTION 9

9 Hashing - Linear Probing 10 / 10

✓ **- 0 pts Correct**

- 2.5 pts 1 item wrong

- **- 5 pts** Two items wrong
- **- 7.5 pts** Three items wrong
- **- 10 pts** Wrong
- **- 10 pts** Blank
- **- 4 pts** Search Wrong
- **- 10 pts** You did not do linear probing

QUESTION 10

10 Hashing - Chaining 10 / 10

✓ **- 0 pts Correct**
- **- 5 pts** Insert Wrong
- **- 5 pts** Search wrong

QUESTION 11

11 Graphs 10 / 10

✓ **- 0 pts Correct**
- **- 5 pts** You must pass two vertexes to insert_edge
- **- 5 pts** You do not make the correct graph
- **- 1 pts** The vertexes should not have labels
- **- 10 pts** Wrong
- **- 3 pts** Error
- **- 10 pts** Skip
- **- 3 pts** Code is slower than O(n^2)
- **- 3 pts** Runtime wrong

QUESTION 12

12 Trees: Aunt / Uncle 10 / 10

✓ **- 0 pts Correct**
- **- 10 pts** Wrong
- **- 3 pts** You check to make sure you dont return yourself, when you should make sure you dont return your parent
- **- 4 pts** Error
- **- 10 pts** Blank
- **- 10 pts** Does not return an iterator over the right items
- **- 5 pts** You don't exclude the parent

QUESTION 13

13 Mode 10 / 10

✓ **- 0 pts Correct**
- **- 3 pts** Runtime is wrong
- **- 3 pts** max computation not right
- **- 10 pts** Skip
- **- 6 pts** Error
- **- 3 pts** Error
- **- 10 pts** Wrong, blank, many mistakes or does not make sense
- **- 2 pts** Minor mix-up of data types
- **- 4 pts** Slow

QUESTION 14

14 Calendar 8 / 10

- **- 0 pts** Correct
- **- 3 pts** Runtime not indicated
- **- 10 pts** Blank/Almost blank
- **- 10 pts** Skip
- **- 10 pts** Does not work. For example, if you call C.block(5,10), C.is_blocked(7) should return True, but it returns False.

✓ **- 2 pts Slow! O(log n) time is possible**
- **- 5 pts** isBlocked does not work
- **- 10 pts** Wrong
- **- 4 pts** Time could be any float, e.g. 1.5
- **- 5 pts** block does not work
- **- 2.5 pts** Error in block
- **- 2.5 pts** Error in isBlocked

QUESTION 15

15 Wildcard searching 0 / 10

- **- 0 pts** Correct

✓ **- 10 pts Skip**
- **- 10 pts** Blank
- **- 10 pts** You must treat ? differently from other characters.
- **- 5 pts** Does not work
- **- 10 pts** Handing of ? is wrong
- **- 10 pts** Wrong

The Final Exam

Data Structures—CS1134—Spring 2017—John Iacono

Instructions

- Read the instructions
- **No questions! None!** (except for "May I please use the bathroom?") If something is wrong or not clear, state any assumptions or changes you make to make it clear or to be solvable. Do not waste you time raising your hand, the proctors have been told to not answer anything.
- No error checking needed. Assume inputs are as described.
- You may use any of the code and classes from the class/book/Python/the next few pages.
- **This exam will be scanned! Do not detach or mutilate the pages. If you need more space go to the end.** If you need alot more space, you are doing something wrong.
- The exam is closed book and notes. No calculators or electronic devices of any kind.
- Six pages, double sided, of cheat sheets allowed
- Slow algorithms that are correct are woth more than fast ones which are incorrect.
- All questions are worth the same number of points.
- Note: I am not giving you the code for the AVL tree, but you can assume you have a class called AVLtree with the same methods of BST class in the provided code, but implements an AVL tree instead, and has the runtimes of an AVL tree.
- You must skip one question. You must write SKIP as the answer to the question. If you do not write SKIP as the answer to a question, I will not grade your answer to the last question.
- Do not cheat.

[∞ POINTS] What is your name, as it appears on your ID card? Please write your first name first and your last name last.

Cindy Lee

[∞ POINTS] Did you read the instructions? There is some important stuff there.

Positional list code:

```
class PList:
    class _Node:
        __slots__='_data','_prev','_next'
        def __init__(self,data,prev,next):
            self._data=data
            self._prev=prev
            self._next=next
    class Position:
        def __init__(self,plist,node):
            self._plist=plist
            self._node=node
        def data(self):
            return self._node._data
        def __eq__(self,other):
            return type(other) is type(self) and other._node is self._node
        def __ne__(self,other):
            return not (self == other)
    def _validate(self,p):
        #Code omitted
    def _make_position(self,node):
        if node is self._head or node is self._tail:
            return None
        else:
            return self.Position(self,node)
    def __init__(self):
        self._head=self._Node(None,None,None)
        self._head._next=self._tail=self._Node(None,self._head,None)
        self._size=0
    def __len__(self):
        return self._size
    def is_empty(self):
        return self._size==0
    def first(self):
        return self._make_position(self._head._next)
    def last(self):
        return self._make_position(self._tail._prev)
    def before(self,p):
        node=self._validate(p)
        return self._make_position(node._prev)
    def after(self,p):
        node=self._validate(p)
        return self._make_position(node._next)
    def __iter__(self):
        pos = self.first()
        while pos:
            yield pos.data()
            pos=self.after(pos)
    def _insert_after(self,data,node):
        newNode=self._Node(data,node,node._next)
        node._next._prev=newNode
        node._next=newNode
        self._size+=1
        return self._make_position(newNode)
    def add_first(self,data):
        return self._insert_after(data,self._head)
    def add_last(self,data):
        return self._insert_after(data,self._tail._prev)
    def add_before(self,p,data):
        node=self._validate(p)
        return self._insert_after(data,node._prev)
    def add_after(self,p,data):
        node=self._validate(p)
        return self._insert_after(data,node)
    def delete(self,p):
        node=self._validate(p)
        data=node._data
        node._prev._next=node._next
        node._next._prev=node._prev
        node._prev=node._next=node._data=None
        self._size-=1
        return data
    def replace(self,p,data):
        node=self._valdiate(p)
        olddata=node._data
        node._data=data
        return olddata
    def rev_itr(self):
        #Code omitted
```

Code for BST class from class:

```
class BST:
    class _Node:
        def __init__(self,parent,left,right,data):
            self._left=left
            self._right=right
            self._parent=parent
            self._data=data
    def __init__(self):
        self._root=None
    def insert(self,x):
        if self._root == None:
            self._root=BST._Node(None,None,None,x)
        else:
            self._rec_insert(self._root,x)
```

```
    def _rec_insert(self,n,x):
        if x<n._data:
            if n._left == None:
                n._left=BST._Node(n,None,None,x)
            else:
                self._rec_insert(n._left,x)
        else:
            if n._right == None:
                n._right=BST._Node(n,None,None,x)
            else:
                self._rec_insert(n._right,x)
    def search_le(self,x):
        if self._root==None:
            return None
        else:
            return self._rec_search_le(self._root,x)
    def _rec_search_le(self,n,x):
        if x<n._data:
            if n._left:
                return self._rec_search_le(n._left,x)
            else:
                return None
        else:
            if n._right:
                rv=self._rec_search_le(n._right,x)
                if rv:
                    return rv
                else:
                    return n._data
            else:
                return n._data
    def delete(self,x):
         #code omitted
```

Tree class from book (code inside the methods omitted):

```
class Tree:
  class Position:
    def element(self):
      """Return the element stored at this Position."""
    def __eq__(self, other):
      """Return True if other Position represents the same location."""
    def __ne__(self, other):
      return not (self == other)           # opposite of __eq__
  def root(self):
    """Return Position representing the tree's root (or None if empty)."""
  def parent(self, p):
    """Return Position representing p's parent (or None if p is root)."""
  def num_children(self, p):
    """Return the number of children that Position p has."""
    raise NotImplementedError('must be implemented by subclass')
  def children(self, p):
    """Generate an iteration of Positions representing p's children."""
  def __len__(self):
    """Return the total number of elements in the tree."""
    raise NotImplementedError('must be implemented by subclass')
  def is_root(self, p):
    """Return True if Position p represents the root of the tree."""
  def is_leaf(self, p):
    """Return True if Position p does not have any children."""
  def is_empty(self):
    """Return True if the tree is empty."""
  def __iter__(self):
    """Generate an iteration of the tree's elements."""
  def positions(self):
    """Generate an iteration of the tree's positions."""
```

Trie code from class/sample exam

```
class Trie:
    class _Node:
        def __init__(self):
            self.children={}
    def __init__(self):
        self._root=self._Node()
    def insert(self,s):
        n=self._root
        for c in s:
            if c not in n.children:
                n.children[c]=self._Node()
            n=n.children[c]
```

Graph class

```
class Graph:
  class Vertex:
    """Lightweight vertex structure for a graph."""
    __slots__ = '_element'
    def __init__(self, x):
      """Do not call constructor directly. Use Graph's insert_vertex(x)."""
      self._element = x
    def element(self):
      """Return element associated with this vertex."""
      return self._element
    def __hash__(self):         # will allow vertex to be a map/set key
```

```
    return hash(id(self))
class Edge:
  """Lightweight edge structure for a graph."""
  __slots__ = '_origin', '_destination', '_element'
  def __init__(self, u, v, x):
    """Do not call constructor directly. Use Graph's insert_edge(u,v,x)."""
    self._origin = u
    self._destination = v
    self._element = x
  def endpoints(self):
    """Return (u,v) tuple for vertices u and v."""
    return (self._origin, self._destination)
  def opposite(self, v):
    """Return the vertex that is opposite v on this edge."""
    if not isinstance(v, Graph.Vertex):
      raise TypeError('v must be a Vertex')
    return self._destination if v is self._origin else self._origin
    raise ValueError('v not incident to edge')
  def element(self):
    """Return element associated with this edge."""
    return self._element
  def __hash__(self):         # will allow edge to be a map/set key
    return hash( (self._origin, self._destination) )
#---------------------- Graph methods ----------------------
def __init__(self, directed=False):
  """Create an empty graph (undirected, by default)
  Graph is directed if optional paramter is set to True.
  """
  self._outgoing = {}
  # only create second map for directed graph; use alias for undirected
  self._incoming = {} if directed else self._outgoing
def _validate_vertex(self, v):
  """Verify that v is a Vertex of this graph."""
  if not isinstance(v, self.Vertex):
    raise TypeError('Vertex expected')
  if v not in self._outgoing:
    raise ValueError('Vertex does not belong to this graph.')
def is_directed(self):
  """Return True if this is a directed graph; False if undirected.
  Property is based on the original declaration of the graph, not its contents.
  """
  return self._incoming is not self._outgoing # directed if maps are distinct
def vertex_count(self):
  """Return the number of vertices in the graph."""
  return len(self._outgoing)
def vertices(self):
  """Return an iteration of all vertices of the graph."""
  return self._outgoing.keys()
def edge_count(self):
  """Return the number of edges in the graph."""
  total = sum(len(self._outgoing[v]) for v in self._outgoing)
  # for undirected graphs, make sure not to double-count edges
  return total if self.is_directed() else total // 2
def edges(self):
  """Return a set of all edges of the graph."""
  result = set()       # avoid double-reporting edges of undirected graph
  for secondary_map in self._outgoing.values():
    result.update(secondary_map.values())    # add edges to resulting set
  return result
def get_edge(self, u, v):
  """Return the edge from u to v, or None if not adjacent."""
  self._validate_vertex(u)
  self._validate_vertex(v)
  return self._outgoing[u].get(v)        # returns None if v not adjacent
def degree(self, v, outgoing=True):
  """Return number of (outgoing) edges incident to vertex v in the graph.
  If graph is directed, optional parameter used to count incoming edges.
  """
  self._validate_vertex(v)
  adj = self._outgoing if outgoing else self._incoming
  return len(adj[v])
def incident_edges(self, v, outgoing=True):
  """Return all (outgoing) edges incident to vertex v in the graph.
  If graph is directed, optional parameter used to request incoming edges.
  """
  self._validate_vertex(v)
  adj = self._outgoing if outgoing else self._incoming
  for edge in adj[v].values():
    yield edge
def insert_vertex(self, x=None):
  """Insert and return a new Vertex with element x."""
  v = self.Vertex(x)
  self._outgoing[v] = {}
  if self.is_directed():
    self._incoming[v] = {}        # need distinct map for incoming edges
  return v
def insert_edge(self, u, v, x=None):
  """Insert and return a new Edge from u to v with auxiliary element x.
  Raise a ValueError if u and v are not vertices of the graph.
  Raise a ValueError if u and v are already adjacent.
  """
  if self.get_edge(u, v) is not None:      # includes error checking
    raise ValueError('u and v are already adjacent')
  e = self.Edge(u, v, x)
  self._outgoing[u][v] = e
  self._incoming[v][u] = e
```

Cindy Lee

1. Consider the following code:

```
def f(A):
    rv=""
    for x in A:
        if x % 2 == 0:
            rv=rv+"E"
        else:
            rv=rv+"O"
    return rv
```

If I call `print(f([10,2,4,3,12]))` it will print `EEEOE`. Logically, this is because 10, 2, 4, and 12 are even and 3 is odd.

(i) What is the runtime of your code if it is called with A of size n. Answer using big O.

$O(n^2)$

1+2+3...

(ii) Write code with the same functionality but is faster. (no points if not faster or if it does not compute and return the string correctly)

```
def f(A):
    rv = []
    for x in A
        if x % 2 == 0:
            rv.append("E")
        else:
            rv.append("O")
    return "".join(rv)
```

(iii) What is the runtime of this code if it is called with A of size n. Answer using big O. No points if (ii) is wrong.

$O(n)$

What is your name?

Cindy Lee

2. What does this print?

```
def g(A,B):
    A.append(B)
    B=A

A=[[1,2],[3,4,5]]
B=[6,7,8,9,10,11]
g(A,B)
print(len(A))
```

A.append(B)

[[1,2], [3,45], [6,7,8,9,10,11]]

3

Cindy Lee

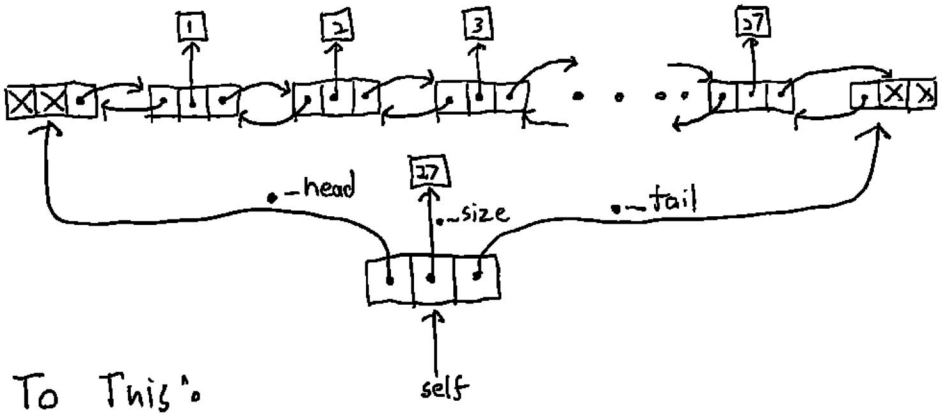
3. Suppose you have a PList drawn below and you are in the middle of coding a method inside the Plist class. Suppose you want to change the single arrow that changes in this figure (the arrow going left out of the node that has element 3). What line (or lines) of code would you write?

node3 represents node with element 3

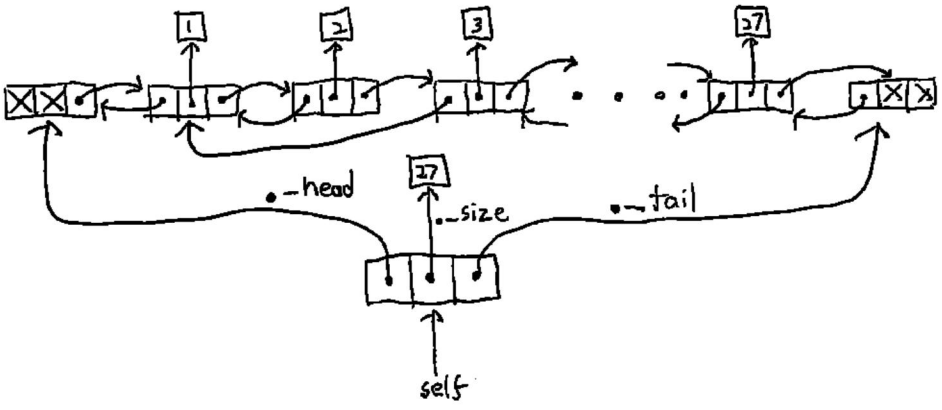
prev = node3._prev

node3._prev = prev._prev

Change This:



To This:

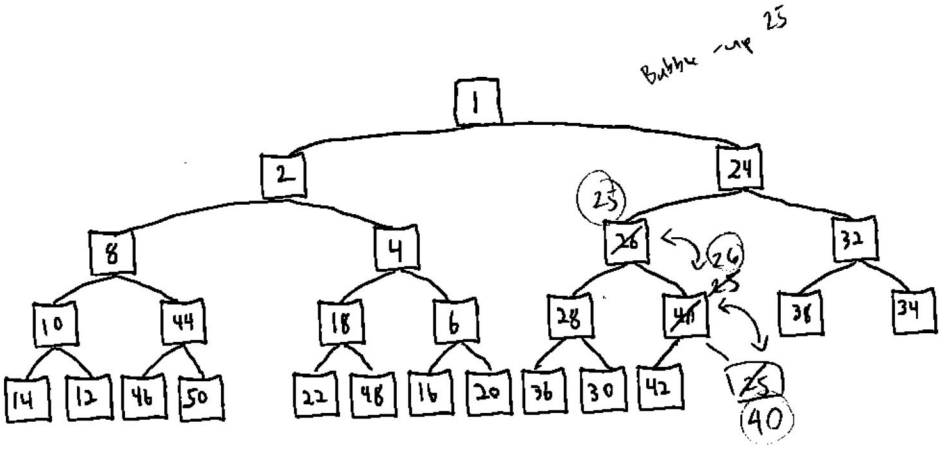


What is your name?

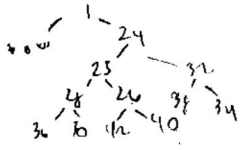
Cindy Lee

4. Heaps.

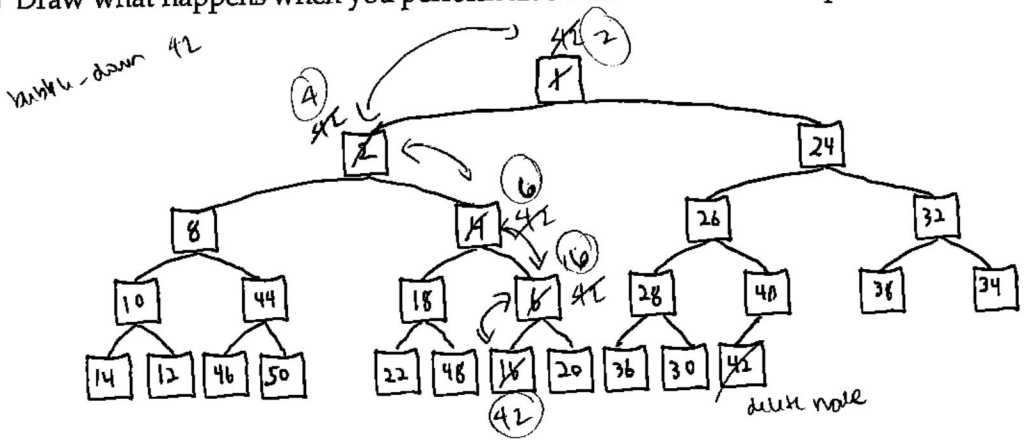
(i) Draw what happens when you insert 25 into this heap



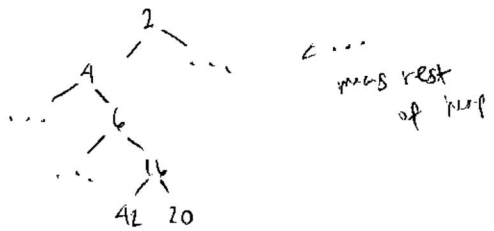
result of right side when bubble-up occurred:



(ii) Draw what happens when you perform an extract-min on this heap:

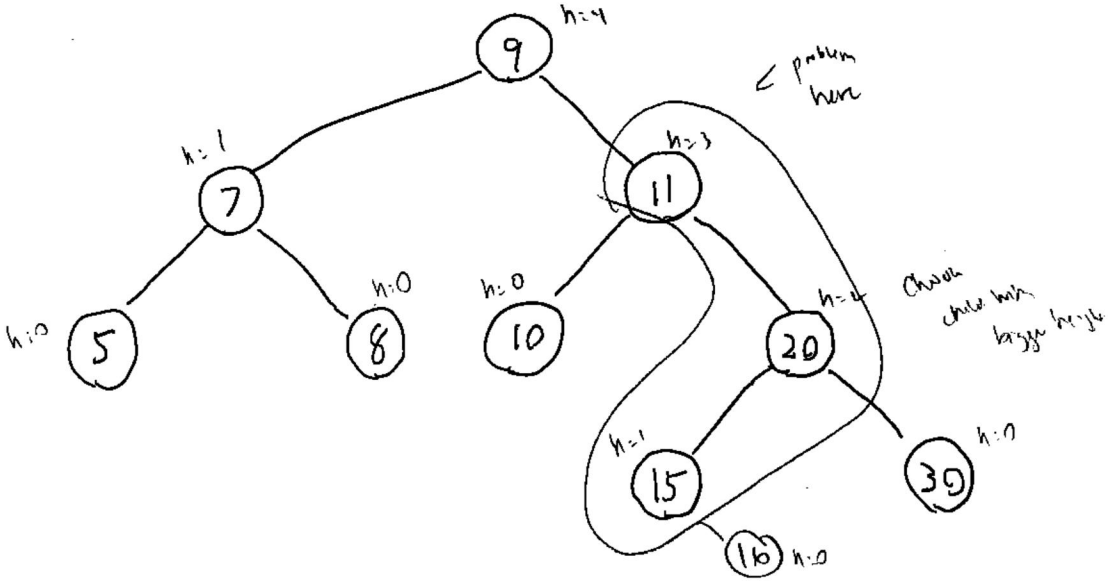


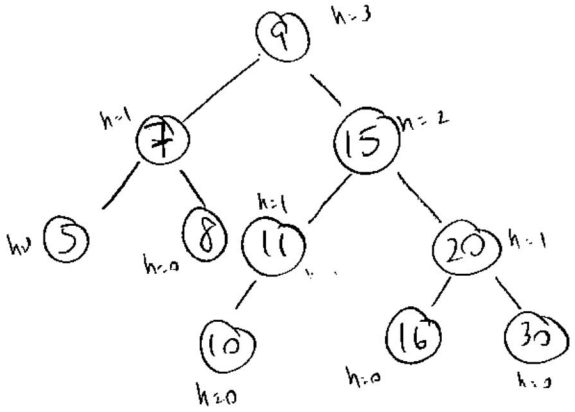
result of bubble down



Cindy Lee

5. Consider the following AVL tree. Draw the structure after you insert 16. I have not drawn the information stored in each node needed to keep the AVL tree balanced.





What is your name?

Cindy Lee

6. Consider the Longest Common Subsequence (LCS) code from class. If one ran it to compute the LCS of "which question will" and "you skip" how many top-level recursive calls are made, and on what pairs of strings? (The quotes are not part of the strings)

"which question will"
"you skip"
↓
start here

w != y

call on w, o

call on h, y

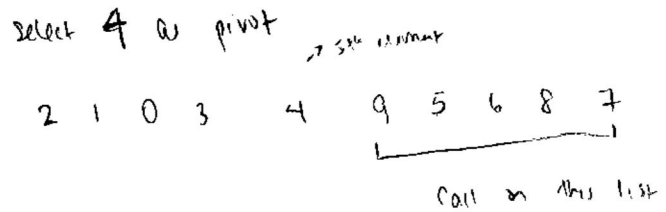
2 recursive calls on "hich question will", "you skip"

and "which question will", "ou skip"

(if using dynamic programming with indexes,
2 recursive calls on "w", "o", and "h", "y")

Cindy Lee

7. In the quick sort-based selection algorithm, suppose you want to find the 8th largest item in [4,2,9,1,5,6,8,0,7,3]. How many recursive calls are made, and with what parameters (i.e what list and looking for the ith largest, for what i). I don't care if you view the min as the 0th largest or the 1st largest.



1 recursive call made on [9, 5, 6, 8, 7]

for 3rd largest

parameters: [9, 5, 6, 8, 7], 3

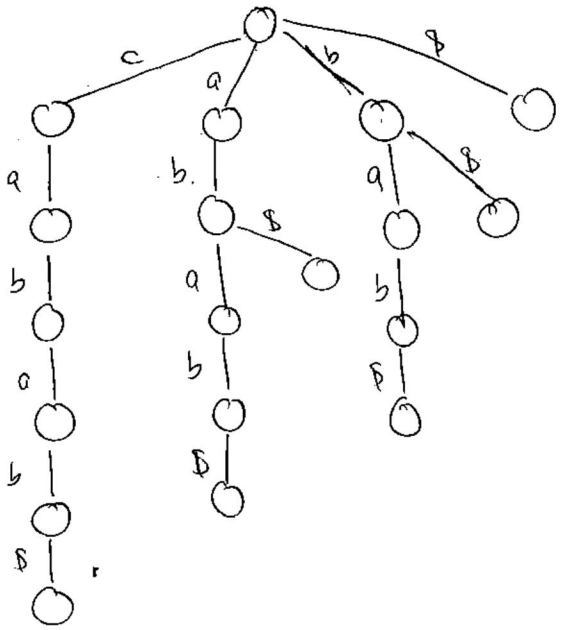
What is your name?

Cindy Lee

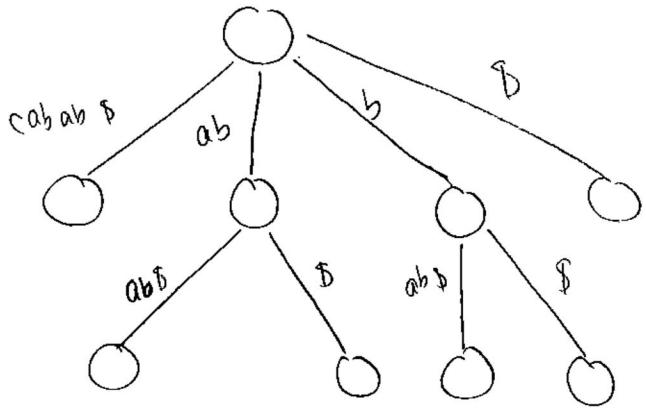
8. Tries.

(i) Draw the regular suffix trie for the string *cabab$*

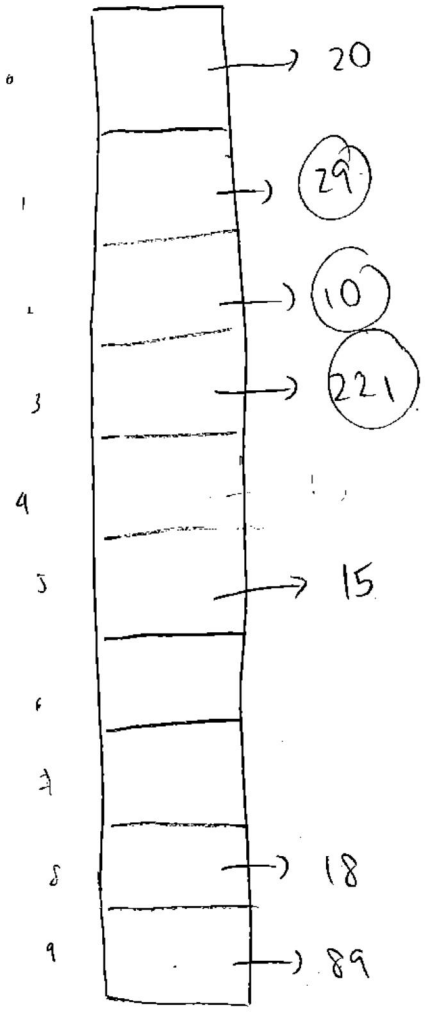
cabab$
abab$
bab$
ab$
b$
$



(ii) Draw the compressed suffix trie for the string *cabab$*



9. Suppose you have a hash table of size 10 using linear probing for collision resolution. Suppose you are storing integers in it, and the hash function is just the number mod 10. Suppose you insert 15, 20, 89, 18, 29, 10, 221. Draw what the hash table looks like. Suppose you search for 51 (which is not in the table), circle the elements of the hash table that you will look at when you execute this search. $h(k) = k \bmod 10$

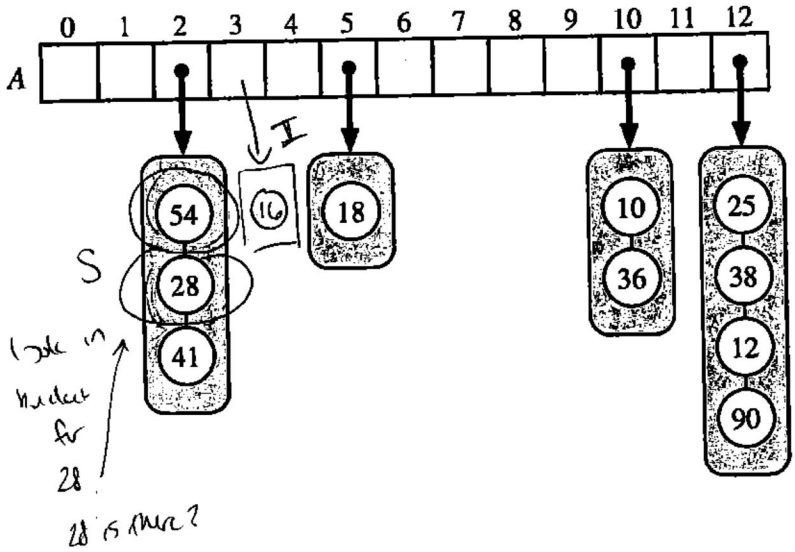


51
↓
$h(51) = 51 \bmod 10 = 1$
look at index 1 first

What is your name?

Cindy Lee

10. Consider the following figure from your book, illustrating a hash table, where the hash function used is $k(k) = k \bmod 13$:



(i) [5 Points] Illustrate in the diagram what will happen if you insert 16. Write "I" next to any changes.

(ii) [5 Points] Suppose you search for 28 in the has table; it is not there. What items in the hash table will you look when searching for 28? Circle them and write "S" next to your circles.

???

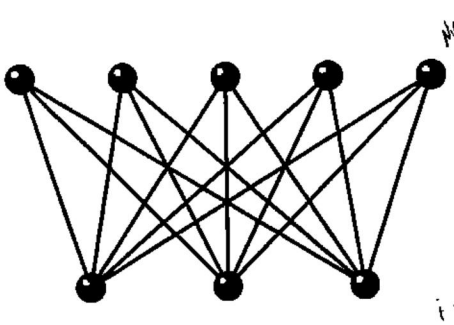
16 mod 13 = 3

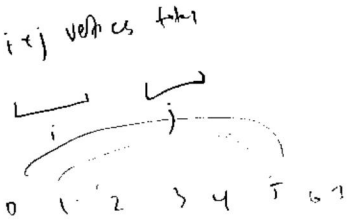
28 mod 13 = 2

28 is there???

Cindy Lee

11. The complete bipartite graph $K_{i,j}$ consists of $i + j$ vertices in two groups of size i and j, respectively. There is an edge connecting every vertex in the first group with every vertex in the second group. Write code for the the stand alone function `K(i,j)` that creates and returns the graph $K_{i,j}$. For example $K(3,5)$ should return the graph drawn below. Note that neither the edges nor the vertices have labels in this problem.






```
def K(i,j):
    g = Graph()
    lst = [ ]
    for x in range(i+j):
        lst.append(G.insert_vertex())
    for y in range(i):
        for z in range(i, i+j):
            G.insert_edge(lst[y], lst[z])
```

What is the runtime of your code if one calls `K(n,n)`?

create $n+n$ nodes, $\rightarrow 2n$

create n^2 edges $\quad O(n^2)$

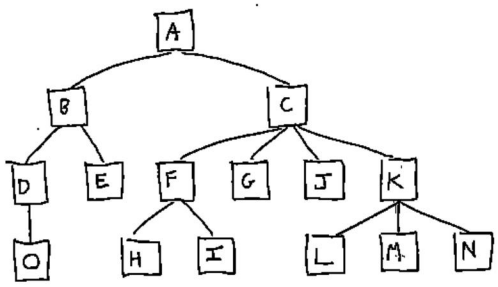
$n^2+n \rightarrow$

n nodes $\rightarrow n$ edges,

What is your name?

Cindy Lee

12. Given the tree class from the book (included at the beginning), write a method auntUncle(self,p), that will return an iterator over all of the positions representing the siblings of the parent of the node p represents (the parent of the node that p represents is NOT to be returned). Note the tree is not a binary tree, it could have many children, and that you can only use the methods in the tree class that are given. For example, if T is the tree below, and p is a position to M, then `print([i.element() for i in T.auntUncle(p)]` would print `[F,G,J]` (in any order).




```
def auntUncle(self, p)
    if not self.is_root(p):
        if self.parent(self.parent(p)):
            parent = self.parent(p)
            grandparent = self.parent(self.parent(p))
            for x in self.children(grandparent):
                if x != parent:
                    yield x
```

Cindy Lee

13. Write a function `mode(A)` that given a list of numbers (not necessarily sorted), returns the mode of the set. The *mode* is the number that appears most frequently. For example if $A = [5,9,2,9,0,5,9,7]$, the mode is 9, since that appears more than any other number. You can assume the mode is unique. What is the runtime of your function? Your grade here will very much depend on the speed of your code.

```
def mode(A):
    d = {}
    if len(A) == 0
        return None
    mode_num = A[0]
    count = 1
    d[A[0]] = 1
    for x in range(1, len(A))
        if A[x] not in d:
            d[A[x]] = 1
        else:
            d[A[x]] += 1
        if d[A[x]] > count
            mode_num = A[x]
            count = d[A[x]]
    return mode_num
```

O(n) if n numbers in A

What is your name?

Cindy Lee

14. Suppose you want to create a very simple calendar class, that keeps track of time as either being blocked or free. Initially, all time is free. For the sake of simplicity, time is represented by a float. It must support two methods:

    - `block(x,y)` blocks the time from just after x up to and including y. You can assume that block will only be called on intervals of time that are completely free.
    - `isBlocked(x)` returns a boolean indicating whether or not the given time was blocked.

    For each method please indicate the runtime in big-O, assuming n is the number of intervals blocked so far. Your grade on this question will depend very much on picking the best data structure that works. You can assume that isBlocked is not called on the endpoints of any period of time that has been blocked, this allows you not to worry about boundary cases.

```
class Calendar:
    def __init__(self):
        self._d = {}

    def block(self, x,y):
        self._d [(x,y)] = True
        # True means is blocked
        # O(1)

    def isBlocked(self, x):
        for k in self._d:
            start = k[0]
            end = k[1]
            if x > start and x <= end:
                return True
        return False
        # O(n)
```

Cindy Lee

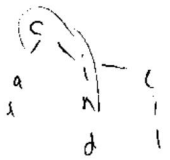
15. Code a function `inWild(T,P)` that given a suffix trie T of some string S, returns a boolean which indicates whether P appears as a substring in S. Additionally, P may contain one or more question marks, which can match any single character. For example, if S="mississippi" and P="ss?s" the function would return True since if you replace the ? with a "i", there is a match. If P="m??p" the function would return False as there is nothing you can replace the ??'s with to get a match.

??

n

```
def inWild(T,P,i=0,n=None):
    if n==None:
        n=T.root
    if i==len(P):
        return True
    if n.children:
        if
```

SKIP



What is your name?

Extra space. If you use it please write "answer on page 20" on the appropriate question.

THE END
Have a great summer!