Ph.D. Comprehensive Examination

Computer Science Department
University of Miami

August 19, 2022

Student Name: Alexander Clemmer

Student Number: C12041265

Problem number	Points (10 max)
1	5
2	0
3	10
4	9
5	10
6	10
7	5
8	8
9	10
10	10
Total:	77

DONE

1. **Data organization; Algorithms and complexity**

Each item x in a set S has a unique key $key[x]$. We need to implement the following operations.

(a) Search (S, key)
(b) Insert (S, x)
(c) Successor (S, x)
(d) Predecessor (S, x)

Give the 4 running times as an O() for the following implementations:

(a) Ordered (sorted) array,
(b) Ordered doubly linked list,
(c) Min-Heap, and
(d) Hash table

5 points

ordering is on x, not key?
these search estimates might go to $O(\log n)$ (binary search)

a) Ordered array

a) Search should be $O(n)$ ⇒ worst case, it's at the end

b) Insert should be $O(n)$ ⇒ worst case, needs to go at "end" (is biggest)

c) $O(1)$
d) $O(1)$ (go to next/previous element in ordered list)

b) Ordered doubly-linked list:

a) Search is $O(n)$ ⇒ linear scan
b) insert is $O(n)$ ⇒ same as search for order maintenance
c) successor
d) predecessor ⇒ easy: doubly linked ⇒ next, previous ⇒ $O(1)$

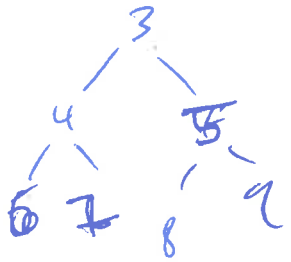
d) Hash table

a/b) Search & insert should be $O(1)$ (hash fn)
c/d) I think $O(n)$? Need to check all; hash fn might map next biggest/smallest somewhere else?

c) min-heap: I really don't know what this is.

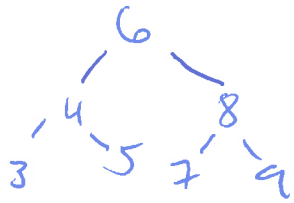
Maybe just a "higher = smaller" tree? or something?

3456789 ⇒



⇒ is it not ordered? like BST?

vs.



Search: if tree, maybe $O(n \log n)$?

addition: $O(n)$? needs to make sure is smallest ⇒ what it needs to add at the top? Bad ⇒ lots of swaps. Maybe only $O(\log n)$ though ⇒ based on depth of tree.

predecessor } successor } $O(?)$ just go up/down (linked list implementation?)

2. **Program control and structure; Programming language and notations**

Suppose that procedure *swap* is declared as follows:

```
procedure swap( x, y: integer);
    procedure f(): integer;
        var z: integer;
        begin // f
            z = x; x = y; return z;
        end // f
    begin // swap
        y = f();
    end // swap
```

ONE

Please see for reverse for full answers

Describe the effect of the procedure call *swap*(i, A[i]) under each of the following parameter passing methods:

(a) Call-by-value

(b) Call-by-reference

(c) Call-by-value-result

↳ I'm not sure what these are (never seen these formal defs before). However, I think the ideas are:

reference value

$\downarrow x = \downarrow 3 \Rightarrow$ pointers?

so 3 is 3

i = 3 ⇒ value = 3

reference = i (address)

A[i] ⇒ value = address of thing @ A[i]

reference = address of A[i]

value-result = value of thing @ address pointed to by A[i]

thought process

example

A_add → A = [$i_1, i_2, \ldots, i_n$]

A[i_k] → i_k → V_k

ref (↑ i_k) value (↑ V_k) result

→ flip for my best explanation in this paradigm

Okay, let's now formally define this.

Swap(i, A[i]) ⇒
- i (address) = 3456
- i (value) = 4
- A[i] (address) = 789
- A[i] (value) = address at ith position of A = 10112
- A[i] (value-result) = value @ address @ ith position of A = 6

I'm constructing this to make my life easier trying to write this all out. Could also use print statements in actual program.

---

b) in address mode: Swap(3456, 789) (add) (add):

Swap { f { z = 3456; 3456 = 789; return 3456; 789 = 3456 } } pointer swap? so 789 now points to 4 & 3456 now points to 10112.

---

c) in value mode: swap(4, 10112): seg fault (access address 4)
(val) (val)

Swap { f { z = 4; 10112 = 4; return 10112; 4 = 10112 } } this is the Sad Thing™: you literally want to swap what's in the array, not the address to the value in position 3 of the array.

---

c) in value-result mode: swap(4, 6)

swap { f { z = 6; 6 = 6; return 6; 4 = 6 } }

actual value swapping.

I think the goal of this question (regardless of if I got the details correct) was to assess/explain the difference between references/pointers & values.

DONE

3. **Software engineering**

From the software engineering point of view, any software development process can be divided into several sub-disciplines:

(a) Requirement Analysis
(b) Functional Specification
(c) Architectural Design
(d) Implementation
(e) Testing and Evaluation
(f) Maintenance

Choose three sub-disciplines or tasks within these sub-disciplines that involve a mathematical approach, and illustrative them with examples.

OK 10 points

not entirely sure what this entails

(e) Testing & Evaluation

- an example of this might be unit testing / test-driven-development: create a (quantitative) test a given unit of code should pass, then write the code such that it passes the test
- ex. unit test for an adding function (in Python)

```
def test_add(add_fn):
    x = 2
    y = 3
    assert add_fn(x, y) == 5
    # Maybe also requires commutativity?
    assert add_fn(y, x) == 5
```

↑ this is the callable adding fn to be designed

(b) Functional specification (or maybe (a) Requirement Analysis?)
- whichever is covered by the SRS document (software requirement specifications), so I guess (a)?)

I remember prioritizing requirements (both functional & nonfunctional) numerically on a scale when I wrote mine. i.e. 0 is absolute highest priority, 5 is probably not that important

flip for last

(f) Maintenance

- one important aspect of maintenance is version control (i.e. tracking what version of the code is currently deployed).
- classically, this has been done w/ version numbers (i.e. v1.0), etc., with major releases perhaps incrementing the version # (v2.0, v3.0) & minor patches/alterations incrementing subsequent values (v1.1, 1.2, 1.3
- also relevant in this discussion are version control systems (such as git and subversion). These allow for (numerically & logically) organized collaboration across machines, users, and versions.

4. **Systems**

DONE

(a) _____ linked libraries can support shared library code, allowing one copy of a library routine to be used by several different processes.

absolute **relative** **static** **dynamic** **none of these is correct**

(b) When it is not known at compile time where a process will reside in memory, _____ code must be generated.

logical **physical** **absolute** **relocatable**

(c) A UNIX process calls *fork()* to create a child process as shown: *pid = fork()*;

i. What value will be assigned to pid in the parent process by the call to *fork()*?

the parent's process id **the child's process id** **zero none of these**

ii. What value will be assigned to pid in the child process by the call to *fork()*?

the parent's process id **the child's process id** **zero none of these**

(I don't think a value is assigned)

(d) The Banker's algorithm is used for deadlock prevention.

denial **prevention** **avoidance** **recovery**

deadlock = thread 2 waiting on B, resides A & thread 2 waiting on A, resides B.

(e) Belady's anomaly can affect the performance of the _____ page replacement algorithm.

FIFO **LRU** **optimal** **SJF**

"recovery" and "denial" don't make sense?

(f) _____ access files are made of fixed length records that allow programs to read and write records in no particular order.

sequential **direct** **logical** **none of these is correct**

(g) When an I/O request is being handled for a user's process, which term refers to the policy of returning control to the user process before the I/O is completed?

synchronous I/O **asynchronous I/O** **delayed I/O** **none of these**

(h) Which multithreading model requires that a new kernel thread be created for each new user thread?

many-to-one **one-to-one** **many-to-many** **none of these is correct**

(i) A process that does not affect, and is not affected by, another process is referred to as:

static **independent** **cooperating** **dynamic** **unbounded**

5. **Software, Programming Techniques**

Given that

$B(x)$ means "x is a bear"
$F(x)$ means "x is a fish", and
$E(x, y)$ means "x eats y",

what is the best English translation of
$\forall x[F(x) \rightarrow \forall y(E(y, x) \rightarrow B(y))]$?

(a) All fish eat bears. X
(b) Every fish is eaten by some bear. X
(c) Bears only eat fish. F(x) →
(d) Every bear eats fish.
(e) Only bears eat fish.

OK 10points

DONE

for all things x
if x is a fish
for all things y
if x is eaten by y
y is a bear
all fish are eaten by bears

for everything:
if it is a fish, it is eaten
by a bear

thought process

for all fish
if they are eaten by something
the thing is a bear
all fish are eaten by bears?
↓
closest to B?
but B has "some" ⇏ ∃ not ∀?
↳ Anything that eats a fish is a bear?

This is it. It's E.

6. **Networking and Communications** DONE

(a) Draw a diagram showing layers of the Internet Protocol Stack and briefly discuss role of each layer.

(b) Describe functions of each layer when a file is transferred from a source to destination using (*file transfer protocol* (FTP).

OK, 4 or 5 layers

a) 5 layers are: physical, link, network, transport, application

low layer

physical: bit stream, BUT determines several things
- rate (how fast data is sent)
- mode of transport: simplex (only sends or receives), half-duplex (either sends or receives but can do both), full-duplex (sends and receives, can be simultaneous)

link: encapsulates data from higher layers w/ head & tail, making a frame
- checks for errors (detection not correction)
- (local) access control w/ media access control (MAC) address
- allows hop-to-hop ⇒ set from one machine to the next

network: gets from start destination to end destination w/ intermediates
- internet protocol (IP) address (analogous to area code in phone #)
- ~~IP is Best (Transmission Rate)~~
- ~~router~~ uses address resolution protocol (ARP) table to resolve local addresses on arrival

higher layer

low layer

transport: packaging of larger files into segments (fragmentation)

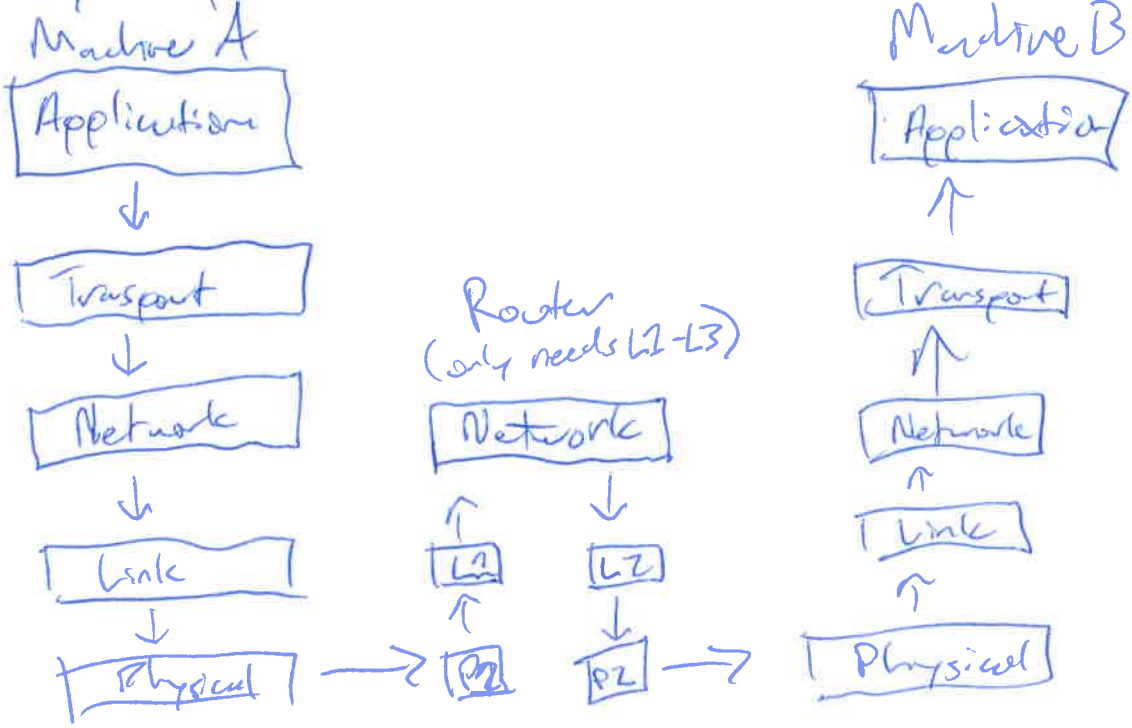
- max frame size in link layer means multiple transport layer packets might be required for one large file
- TCP (transmission control protocol) is a transport layer protocol

application: highest layer, control, dictates what to send

- FTP (file transfer protocol) is an application layer protocol

highest layer

Diagram:



b) FTP establishes 2 TCP connections on ports 20 & 21. One is a control, one is for data transmission. (This is somewhat analogous to buses in comp. architecture). TCP uses 3-way handshake to open stream (e.g. 1) "Hi" 2) "Cool, ready." 3) "ready!"). Breaks application layer data (i.e. file in FTP) into packets, sends these over physical layer in link layer frames by network layer IP address.

DONE

7. **Algorithms and complexity**

Describe an algorithm that takes two input lists of integers $A = a_1, ..., a_n$ and $B = b_1, ..., b_m$ and delivers the list of all the elements that belong to A but not to B. A and B do not contain redundant elements, however, the elements of A and B might have a large range.

The algorithm should run in $O(n \log m + m \log m)$ time.

5 points

Okay: general approach:

- sort A & sort B, can sort in each $O(n \log n)$
- now can just go through each list simultaneously ⇒ linear search/scan ⇒ $O(n)$ or $O(m)$, whichever is longer
- $O(n \log n)$ dominates runtime; will be $O(n \log n)$ (for A) + $O(m \log m)$ for (B)

⇓

$O(n \log n + m \log m)$

≡

↳ not $O(n \log m + m \log m)$

I assumed difference, not only A;

Only diff. is not saving out B-list values

① Something like merge sort of A & B to get ordered A&B
- merge sort is $O(n \log n)$ with input size n and scales well with large arrays

② Linear comparison approach ⇒ take the smaller of the two start values (if different). If same ⇒ move to next element in list. If we run out of one list: the rest of the other is unique, add it; we're done.

↳ (see reverse for implementation)

Recursive approach: THIS IS CORRECT

```
scan_return_difference (A (ordered), B (ordered), C):
    if A & B are empty: return C
    if A is empty & B is not:
        return C
    if B is empty & C is not:
        return C + A (array concat)

    if A[0] == B[0]:
        return scan_return_dif (A[1:], B[1:], C)

    if A[0] < B[0]:
        add A[0] to C  (C.append(A[0]))
        return scan_return_dif (A[1:], B, C)

    if A[0] > B[0]:
        return scan_return_dif (A, B[1:], C)
```

end condition

this can start as an empty array (C = [] in python)

case #1: first element is the same in both

case #2: first ele in A is smaller ⇒ unique ⇒ save it, keep going

Case #3: same as #2 but B is smaller

↳ "the rest of A", etc.

You could do a wrapper like:

```
array_dif (A (unsorted), B (unsorted)):
    A_sort = merge_sort (A)
    B_sort = merge_sort (B)
    return scan_return_diff (A_sort, B_sort, [])
```

8. **Automata and language theory**

Consider the following grammar:

8 points

$G \longrightarrow S\ \$\$$
$S \longrightarrow A\ M$
$M \longrightarrow S \mid \epsilon$
$A \longrightarrow a\ E \mid b\ A\ A$
$E \longrightarrow a\ B \mid b\ A \mid \epsilon$
$B \longrightarrow b\ E \mid a\ B\ B$

(a) Describe the language that the grammar generates in English.

(b) Show a parse tree for the string `a b a a`.

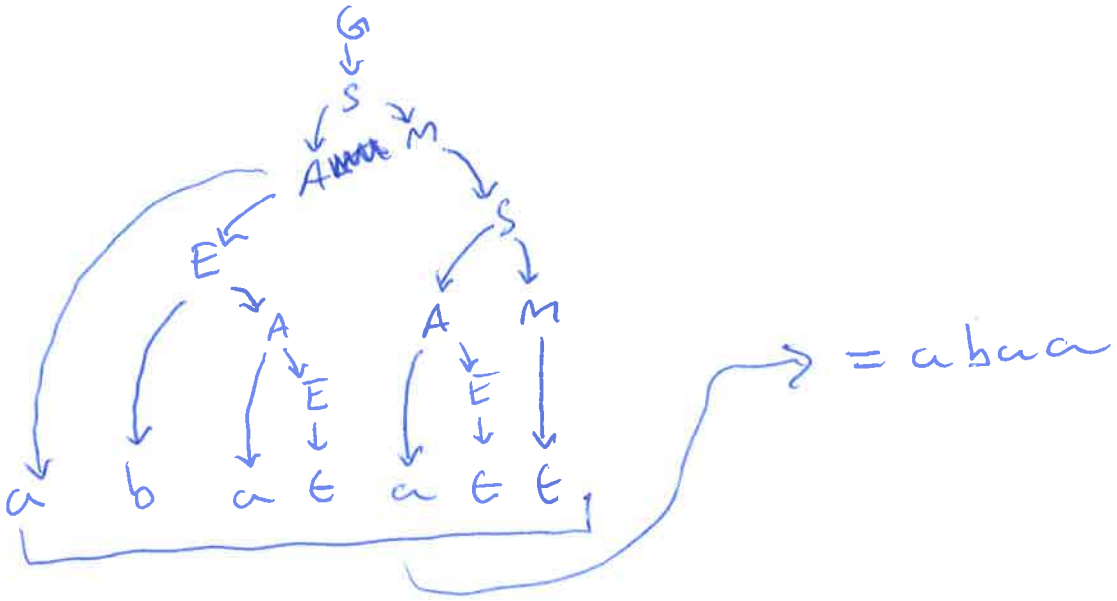
(c) Is the grammar LL(1)? If so, show the parse table; if not, identify a prediction conflict.

Please see reverse for final answers

a) I think this grammar describes any string such that the alphabet is [a, b]. NOPE: "b" is not in the language. Hmmm. Well, no string in the language starts with b. Maybe $L = \{ w \text{ s.t. } w[0] != b \mid [a,b] \}$?

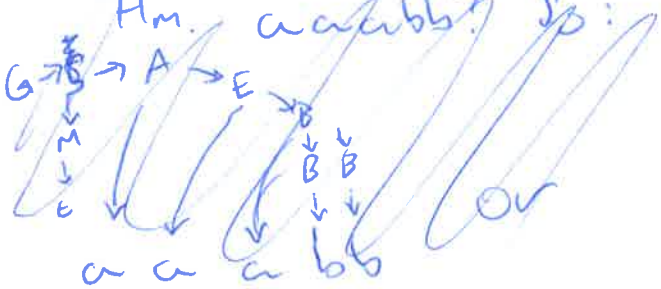
I think this is it. Could get opposite by replacing $S = AM$ with $S = BM$. Nope(bAA) ⇒ (any string in language must have an a in it.)

b) I don't formally remember rules for parse trees, but here is my best effort. The key is abca + a (≈ A + M ⇒ A)



c) Okay, again, don't remember exactly what LL(1) is. BUT ⇒ "prediction conflict" ⇒ I'm guessing this means "does there exist a string in the language that can be parsed multiple ways?" If yes ⇒ not LL(1), if no ⇒ LL(1).

Hm. aaabbb? So:



9. **Discrete Structures** DONE

Recall that the Hamiltonian Cycle Problem is the problem of deciding, on input graph G, whether G has a cycle that visits all the nodes exactly once. Show that this problem is polynomial time decidable if the input is restricted to the graphs with the property that each node has at most two neighbors (i.e., at most two adjacent nodes).

Thought Process

Okay. Um. Maybe DFS from a start node? When you hit the start node again, it's a cycle. Normally, scales like $O(\text{Nodes} + \text{Edges})$. Infinite edges $\Rightarrow$ 2 scales, huge. Restrict edges to 2 per node $=$ 2x # nodes $\Rightarrow O(\#\text{Nodes})$ $= O(\text{Nodes})$ only.

DFS:

queue a start node
while the queue isn't empty:
  take out a node
  if it's the start node:
    we did it! return true
  if we haven't visited it yet: (track of node variable or something)
    add to queue it's children (realistically, one other child)
return false

This works because @ most 2 edges restricts to a set # of possible graphs:

OK, you listed possible scenarios, 10 points

10. **Other Topics** DONE

Give a detailed explanation of any one approach to machine learning. Give a substantial example that illustrates the technical operation of the approach, and demonstrates interesting knowledge learned.

↳ Okay. Broadly, the field of ML can be grouped as follows:

OK, 10 points

Unsupervised learning

- no labels for input data
- exploratory
- examples:
  - clustering
  - dimensionality reduction
- concrete examples
  - Principal Component Analysis (PCA)
  - K-means clustering
  - Uniform manifold approximation & projection (UMAP)

Supervised Learning

- labels for input (training) data
- goal is to learn mapping of input data to either a categorical label (classification) or a value &
- example models range from simple (linear regression) to complex (neural nets)

↳ Unsupervised Learning Example: K-Means clustering

- K-means is based on the idea of grouping sets of data into a (user-defined #) set of K clusters.
- it has a 2 step algorithmic approach:

① Assignment: each datum is assigned to the cluster centroid C to minimize the Euclidean distance from the datum to C.

② Update: the centroids $C_1, C_2, \ldots, C_K$ are moved to minimize the variance^2 in distance to the points in their cluster.

This continues until no datum changes assignment / no centroid moves? (local) minimum

K-means has several issues though:

- the initial placement of cluster centers (random? how to determine?)
- the # of clusters (user-defined)
- this algorithm can get stuck in local minima (i.e. AN assignment, but not THE BEST assignment)

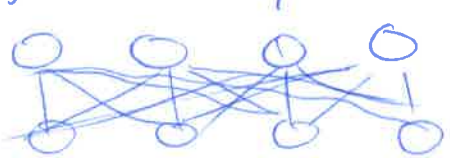
Supervised Learning Example: Convolutional neural nets for image classification

- Training set of x images x 224 wide x 224 high x 3 color channels
  - ↳ input tensor is X x 224 x 224 x 3

- Goes through:
  - convolutional layers: inputs are convolved w/ a filter/kernel (square, odd ⇒ 3x3, etc.)
  - activation layers: alters raw outputs from conv. layer, common functions include RELU
    - RELU function allows all positive values to remain unchanged, sets (-) vals to ∅ (0)

$$RELU(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

  - fully connected (FC) layers: all neurons (individual weighting fns) in one layer connected to those in next



  - softmax layer ⇒ normalizes inputs s.t. they form a probability dist. (an array w/ # of classes cells, sums to 1)

- Train network by 2-pass method: propagate training data forward, get predicted outputs, use predictions to backpropagate to alter weights at each layer.

Ex. VGG16

Architecture:
Input → 5 blocks of (convolutional layer + activation layer) →
2 FC layers →
softmax layer

Weights of model trained on ImageNet (millions of img / 1K classes) available through Tensorflow