York University
EECS 2011Z Winter 2018 – Midterm Tuesday Feb 27th: Solutions
Instructor: James Elder

1. (6 + 6 = 12 marks) Provide tight asymptotic bounds for the following functions of n. Justify your answer as formally as possible.

   (a) $f(n) = \frac{(n+1)^2}{n}$

   - Answer: Claim: $f(n) \in \mathcal{O}(n)$.

     Proof: We need to find $c, n_0 > 0$ such that $f(n) \leq cg(n) \forall n \geq n_0$.

     Note that $f(n) = n + 2 + 1/n$. Let $c = 2, n_0 = 3$.

     Then $cg(n) - f(n) = n - 2 - 1/n$. For $n \geq n_0 = 3$, $cg(n) - f(n) \geq 1 - 1/3 > 0$. Thus $f(n) \leq cg(n) \forall n \geq n_0 \rightarrow f(n) \in \mathcal{O}(n)$.

     Claim: $f(n) \in \Omega(n)$.

     Proof: We need to find $c, n_0 > 0$ such that $f(n) \geq cg(n) \forall n \geq n_0$. Let $c = 1$. Then $f(n) - cg(n) = 2 + 1/n > 0 \forall n > 0$. Thus $f(n) \in \Omega(n)$

     $f(n) \in \mathcal{O}(n)$ and $f(n) \in \Omega(n) \rightarrow f(n) \in \Theta(n)$.

   (b) $f(n) = 10n^2 + \frac{1}{\sqrt{n}} n^3$

   - Answer:

     Claim: $f(n) \in \mathcal{O}\left(n^{5/2}\right)$.

     Proof: We need to find $c, n_0 > 0$ such that $f(n) \leq cg\left(n^{5/2}\right) \forall n \geq n_0$.

     Let $c = 2, n_0 = 100$.

     Then $cg(n) - f(n) = n^{5/2} - 10n^2 = n^2(\sqrt{n} - 10) > 0 \forall n \geq n_0 = 100$.

     Thus $f(n) \leq cg(n) \forall n \geq n_0 \rightarrow f(n) \in \mathcal{O}(n)$.

     Claim: $f(n) \in \Omega\left(n^{5/2}\right)$.

     Proof: We need to find $c, n_0 > 0$ such that $f(n) \geq cg\left(n^{5/2}\right) \forall n \geq n_0$.

     Let $c = 1, n_0 = 1$.

     Then $f(n) - cg(n) = 10n^2 > 0 \forall n \geq n_0 = 1$. Thus $f(n) \in \Omega(n)$

     $f(n) \in \mathcal{O}(n)$ and $f(n) \in \Omega(n) \rightarrow f(n) \in \Theta(n)$.

2. ($5 \times 2 = 10$ marks) Provide a tight bound on the asymptotic run time of each of the following Java code fragments, in terms of n. You do not need to justify your answer.

(a)
```
for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j++) {
        System.out.println("All work and no play makes Jack a dull boy.");
    }
}
```

Answer:

- Answer: n^2

```
for (i = 1; i <= n*n; i++) {
    for (j = 1; j <= n; j++) {
        System.out.println("All work and no play makes Jack a dull boy.");
    }
}
```

Answer:

(b)
- Answer: n^3

(c)
```
for (i = 1; i <= n; i++) {
    for (j = 1; j <= n*n; j++) {
        System.out.println("All work and no play makes Jack a dull boy.");
    }
}
```

Answer:

- Answer: n^3

```
for (i = 1; i <= Math.sqrt(n); i++) {
    for (j = 1; j <= Math.sqrt(n); j++) {
        System.out.println("All work and no play makes Jack a dull boy.");
    }
}
```

Answer:

(d)
- Answer: n

(e)
```
for (i = 1; i <= Math.sqrt(n); i++) {
    for (j = 1; j <= n*n; j++) {
        System.out.println("All work and no play makes Jack a dull boy.");
    }
}
```

Answer:

- Answer: $n^2\sqrt{n}$

3. (6 + 6 = 12 marks) You are designing a new linear algebra Java library. One of the classes in your library is called Matrix. An object of class Matrix contains a 2D rectangular array of double values.

   The signature for the Matrix constructor is shown below:

```
public Matrix (double[][] matrix) throws InvalidMatrixException
```

   (a) Please explain the conditions under which your constructor will throw an InvalidMatrixException. Note that row vectors, column vectors, scalars and null values are all considered Matrices and should not result in an exception.

   - Answer: The constructor should throw an exception if matrix is not rectangular, i.e., if the lengths of its rows are not equal.

   (b) Please explain in English, pseudocode or Java how you would test for these conditions.

   Answer: I would iterate over the rows, checking that they are all equal in length:

```
int rowLength = matrix[0].length;
for (i=1;i<matrix.length;i++) {
 if (matrix[i].length != rowLength) {
  throw InvalidMatrixException;
 }
}
```

4. ($4 \times 4 = 16$ marks) You are designing a system that automatically selects a sequence of requested songs to play on York's student-run radio station. A requested song can be played at most once per day. Each song is represented by an entry in your data structure, and you are free to add elements to the contents of an entry (e.g., a key). Please indicate the ADT you will use to store requested songs to efficiently satisfy the daily conditions below. Justify your answer in at most two sentences.

   (a) On Mondays, you play the most recently requested song next.

   - Answer: I would use a stack. Since it is LIFO, the most recently requested (pushed) song will be the next played (popped). Adding and selecting songs for airplay are both $\mathcal{O}(1)$.

   (b) On Tuesdays, you play songs in the order they are requested.

   - Answer: I would use a queue. Since it is FIFO, the first song to be requested (queued) that hasn't been played will be played (dequeued) next. Adding and selecting songs for airplay are both $\mathcal{O}(1)$.

   (c) On Wednesdays, you play songs that have been requested so far that day, but in order of their ranking on the Billboard 100 charts for that week.

   - Answer: I would use a min-heap-based priority queue with the Billboard 100 rank as key. Adding and selecting songs for airplay are both $\mathcal{O}(\log n)$.

   (d) On Thursdays, you select songs based upon how many requests they have received so far that day - the most requested song is played next. Note that requests come in continuously. You may assume that each request enters the system as a complete entry for the requested song.

   - Answer: I would use a max-heap-based adaptable priority queue with the number of requests as key. Storing the position of the entry in the entry itself will allow the entry to be located when a song is requested in constant time so that the number of requests for that song can be incremented. Adding and selecting songs for airplay are both $\mathcal{O}(\log n)$.

5. (20+4 = 24 marks) You have designed a clinical study for a new drug called Einstein that is purported to raise IQ by 30 points. To test the drug you gave it to a sample of n_t test subjects, and gave a placebo to a separate sample of n_c control subjects. The data records for the two samples of subjects are stored in two singly-linked lists named test and control, and are sorted by pre-study (baseline) IQ (a positive integer).

   To perform a statistical analysis of results, you now wish to construct the largest possible subset of IQ-matched 1:1 pairings of test and control subjects. In other words, for every test subject in your subset there should be a unique control subject with matching baseline IQ, and vice-versa.

   (a) Provide the pseudo-code for an iterative algorithm (testp, controlp) = pair(test, control) that efficiently extracts and returns this maximal 1:1 pairing of test and control subjects as two new singly-linked lists testp and controlp. You may assume that each list L provides an iterator over nodes L.iterator(), that an element can be accessed from a node A using A.getElement(), and that the IQ stored in an element e can be accessed using e.getIQ(). An element can be added to the end of a list L using L.add(e).

   Answer:

```
(testp, controlp) = pair(test, control)
 itTest = test.iterator()
 itControl = control.iterator()
 if itTest.hasNext() && itControl.hasNext()
  t = itTest.next()
  c = itControl.next()
  while (t != null && c != null)
   if t.getElement().getIQ = c.getElement().getIQ
    testp.add(t.getElement())
    controlp.add(c.getElement())
    t = itTest.next
    c = itControl.next
   else if t.getElement().getIQ < c.getElement().getIQ
    t = itTest.next
   else
    c = itControl.next
```

   (b) What is the asymptotic run time of your algorithm, in terms of n_t and n_c? Justify your answer in one or two sentences.

   - Answer: In the worst case we will have to step through all elements in both lists, so the run time is $\Theta(n_t + n_c)$.

6. ($13 + 4 + 3 + 3 + 3 = 26$ marks) Neurotree is an online project that documents academic genealogy according to mentoring relationships in the neurosciences. Every node in the tree contains an element that represents a neuroscientist, and the children of the node (the 'mentees') are the graduate students and postdoctoral fellows they have mentored. Let us assume for simplicity that each neuroscientist appears only once in the tree and that a mentee has at most one mentor. We define the mentorship distance dist(A,b) as the depth of the node B containing element b in the subtree rooted at node A. If B is not a descendent of A, dist(A,b)= ∞. The tree uses a linked structure. An iterable collection of the child nodes of node A can be accessed using A.children(), and its parent node can be accessed using A.parent(). A.element() returns the element stored at node A and a comparator compare(a,b) can be used to compare the values of two elements a and b.

   (a) Provide an efficient recursive pseudocode implementation of dist(A,b).

   Answer:

```
dist(A,b)
 if compare(A.element(),b) = 0
  return 0
 d = ∞
 for C in  A.children()
  d = min(d, dist(C,b)) + 1
 return d
```

   (b) Provide a tight bound on the asymptotic run time of your algorithm in terms of the number n of nodes in the subtree rooted at A. Justify your answer in one or two sentences.

   - Answer: In the worst case, all nodes in the subtree will be visited, so the run time is $\Theta(n)$.

   (c) Suppose that the signature for your algorithm is changed to be dist(A,B), where B is now the node containing mentee element b.

      i. Describe in one to two sentences how you would change your algorithm to make it more efficient.

      - Answer: It will now generally be faster to traverse from B to A, since each node has at most one parent.

      ii. What is the asymptotic run time of your algorithm in terms of the height h of the subtree rooted at A?

      - Answer: $\Theta(h)$

      iii. If the tree is a complete binary tree, what is the run time in terms of the number n of nodes in the subtree rooted at A?

      - Answer: $\Theta(\log n)$, since h is $\Theta(\log n)$.