CSCI 3104 Spring 2018 Problem Set 3
Mohamed Al-Rasbi (07/25)

Problem 1

(5 pts total) For parts (1a) and (1b), justify your answers in terms of deterministic QuickSort, and for part (1c), refer to Randomized QuickSort. In both cases, refer to the versions of the algorithms given in Lecture 3.

(a) *What is the asymptotic running time of QuickSort when every element of the input A is identical, i.e., for $1 \leq i; j \leq n$, $A[i] = A[j]$?*

The running time will be $\Theta(n^2)$. The reason is because in the partition function we have $\leq$ sign, so as long as the elements have the same value, then we will exchange them everytime.

(b) *Let the input array A = [9, 7, 5, 11, 12, 2, 14, 3, 10, 6]. What is the number of times a comparison is made to the element with value 3?*

The number of times is 3. The value 3 will be a pivot two times, and will be compared to a pivot one time.

(c) *How many calls are made to* `random-int` *in (i) the worst case and (ii) the best case? Give your answers in asymptotic notation.*

Worst case: $O(n^2)$
Best case: $\Omega(n \log n)$

Problem 2

(30 pts total) Professor Trelawney has acquired $\boldsymbol{n}$ *enchanted crystal balls, of dubious origin and dubious reliability. Trelawney needs your help to identify which crystal balls are accurate and which are inaccurate. She has constructed a strange contraption that fits over two crystal balls at a time to perform a test. When the contraption is activated, each crystal ball glows one of two colors depending on whether the* ***other*** *crystal ball is accurate or not. An accurate crystal ball always glows correctly according to whether the other crystal ball is accurate or not, but the glow of an inaccurate crystal ball cannot be trusted. You quickly notice that there are four possible test outcomes:*

crystal ball i glows	crystal ball j glows		
red	red	$\Rightarrow$	at least one is inaccurate
red	green	$\Rightarrow$	at least one is inaccurate
green	red	$\Rightarrow$	at least one is inaccurate
green	green	$\Rightarrow$	both accurate, or both inaccurate

(a) *Prove that if* $n/2$ *or more crystal balls are inaccurate, Trelawney cannot necessarily determine which crystal balls are accurate using any strategy based on this kind of pairwise test. Assume a worst-case scenario in which the inaccurate crystal balls contain malicious spirits that collectively conspire to fool Trelawney.*

If we have n/2 balls inaccurate, then when we compare two balls with each other, we have these 3 cases:
1. red-red: at least one is inaccurate, so we can't know if there is one of them is accurate.
2. red-green: at least one is inaccurate, so we can't know if there is one of them is accurate.
3. green-green: Either both of them is accurate, or both of them are inaccurate. Since we know that inaccurate balls contain malicious spirits that conspire to fool Trelawney, there is know way for her to know if both of them are accurate or inaccurate. Because the balls should fool her every time she uses the pair-wise comparison.
Therefore, if n/2 or more crystal balls are inaccurate, there is now way for Trelawney to determine which crystall balls are accurate.

(b) *Suppose Trelawney knows that more than* $n/2$ *of the crystal balls are accurate, but not which ones. Prove that* $\lfloor n/2 \rfloor$ *pairwise tests are sufficient to reduce the problem to one of nearly half the size.*

Let's say we have 5 balls. We will make 2 comparisons since $\lfloor 5/2 \rfloor = 2$. Let's say we get back from the first comparison green-red glow or red-red glow, we know

for sure that at least one of these balls is inaccurate.
If we get from the second comparison green-green, then we know that those two balls have to be accurate because if they are not, then there will be at least 3 inaccurate balls, and that contradicts with the question.
Now we have two accurate balls, and 3 known balls. We can compare one of the accurate balls with one of the balls from the first comparison. Depending on the glow, we will know which one of them are accurate. And then we do another comparison with the fifth ball.
This proves that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

(c) *Now, under the same assumptions as part (2b), prove that all of the accurate crystal balls can be identified with $\Theta(n)$ pairwise tests. Give and solve the recurrence that describes the number of tests.*

The running time of this algorithm is $T(n) = T(n/2) + n/2$ because when we do $\lfloor n/2 \rfloor$ comparisons, we will reduce the problem to half the size and do other comparisons and that means $T(n/2)$.
$T(n) = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n/2}{n/2} = 1$, so $T(n) = \Theta(n/2) = \Theta(n)$

Problem 3

(20 pts) Professor Dumbledore needs your help. He gives you an array A consisting of n integers $A[1], A[2] \ldots, A[n]$ and asks you to output a two-dimensional $n \times n$ array B in which $B[i, j](for i < j)$ contains the sum of array elements $A[i]$ through $A[j]$, i.e., $B[i, j] = A[i] + A[i + 1] + \cdots + A[j]$. (The value of array element $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what the output is for these values.) Dumbledore suggests the following simple algorithm to solve this problem:

```
dumbledoreSolve(A) {
  for i = 1 to n {
    for j = i+1 to n {
      s = sum(A[i..j]) // look very closely here
      B[i,j] = s
}}}
```

(a) *For some function* `g` *that you should choose, give a bound of the form $\Omega(g(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm).*

$g(n) = n^3$, $\Omega(n^3)$
We have three loops; two for loops and sum (which is basically a for loop). This is why in asymptotic analysis this is $\Omega(n^3)$.

(b) *For this same function* `g`*, show that the running time of the algorithm on an input of size n is also $\mathcal{O}(g(n))$. (This shows an asymptotically tight bound of $\Theta(g(n))$ on the running time.)*

The reason I chose $g(n) = n^3$ because we have three inner loops, or in other words, three sums. And in asymptotic analysis that means $g(n) = n^3$. The first loop starts from $i = 1$ to n, the second loop starts from $j = i$ to n, and the third loop is ths sum from i to j.
$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{i}^{j} k$
This means that the worst case is the same as the best case, and that is a tight bound. $\Omega(n^3) = O(n^3) = \Theta(n^3)$

(c) *Although Dumbledore's algorithm is a natural way to solve the problem–after all, it just iterates through the relevant elements of B, filling in a value for each– it contains some highly unnecessary sources of ineffciency. Give an algorithm that solves this problem in time $\mathcal{O}(g(n)/n)$ (asymptotically faster) and prove its correctness.*

We first should know the source of the insufficient in this algorithm. We can see

that through each iteration we sum the items from i to j, and in the next iteration we sum from i to j+1. From here we can see that we can use the previous sums for each iteration.

```
dumbledoreSolve(A) {
  for i = 1 to n {
    for j = i+1 to n {
      if j == i+1
        B[i,j] = A[i]+A[j]
      else
        B[i,j] = B[i,j-1]+A[j]

}}}
```

Problem 4

(15 pts extra credit) With a sly wink, Dumbledore says his real goal was actually to calculate and return the largest value in the matrix B, that is, the largest subarray sum in A. Butting in, Professor Hagrid claims to know a fast divide and conquer algorithm for this problem that takes only $\mathcal{O}(n \log n)$ time (compared to applying a linear search to the B matrix, which would take $\mathcal{O}(n^2)$ time).

Hagrid says his algorithm works like this:

- Divide the array A into left and right halves
- Recursively find the largest subarray sum for the left half
- Recursively find the largest subarray sum for the right half
- Find largest subarray sum for a subarray that spans between the left and right halves
- Return the largest of these three answers

On the chalkboard, which appears out of nowhere in a gentle puff of smoke, Hagrid writes the following pseudocode for his algorithm:

```
hagridSolve(A) {
  if (A.length()==0) { return 0 }
    return hagHelp(A,1,A.length())
}

hagHelp(A, s, t) {
  if (s > t) { return 0 }
  if (s == t) { return A[s] }

  m = (s + t) / 2

  leftMax = sum = 0

  for (i = m-1, i >= s, i--) {
    sum += A[i]
    if (sum > leftMax) { leftMax = sum }
  }
```

```
  rightMax = sum = 0

  for (i = m+1, i <= t, i++) {
    sum += A[i]
    if (sum > rightMax) { rightMax = sum }
  }

  spanMax = leftMax + rightMax
  halfMax = max( hagHelp(s, m-1), hagHelp(m+1, t) )
  return max(spanMax, halfMax)
}
```

Hagrid claims that his algorithm is correct, but Dumbledore says "tut tut." (i) Identify and fix the errors in Hagrid's code, (ii) prove that the corrected algorithm works, (iii) give the recurrence relation for its running time, and (iv) solve for its asymptotic behavior.

(i) there are many errors, I wrote the code again with the fixed errors.

```
  hagridSolve(A) {
    if (A.length()==0) { return 0 }
      return hagHelp(A,1,A.length())
  }

  hagHelp(A, s, t) {
    if (s > t) { return 0 }
    if (s == t) { return A[s] }

    m = (s + t) / 2

    leftMax = sum = 0

    for (i = m, i >= s, i--) {
      sum += A[i]
      if (sum >= leftMax) { leftMax = sum }
    }

    rightMax = sum = 0

    for (i = m+1, i <= t, i++) {
```

```
    sum += A[i]
    if (sum > rightMax) { rightMax = sum }
  }

  spanMax = leftMax + rightMax
  halfMax = max( hagHelp(s, m), hagHelp(m+1, t) )
  return max(spanMax, halfMax)
}
```

(ii) `hagridSolve` is an equivalent implementation of the recursive algorithm for solving the maximum subarray problem, which was proven in the book.

(iii) $T(n) = 2T(n/2) + \Theta(n)$

I worked with:
Omar Mohammed, Mahmoud Almansouri, Firas Al Mahrouky, Yazeed Almuqwishi, Absdullah Bajkhaif