Assignment 2

Due 23:59, Friday, February 14, 2020

CS ID 1: Cinda

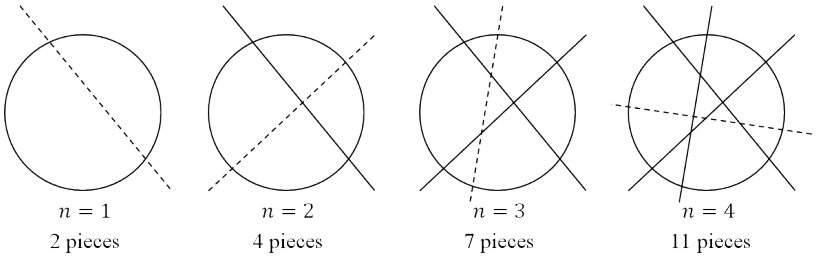
CS ID 2:

Instructions:

1. Do not change the problem statements we are giving you. Simply add your solutions by editing this latex document.
2. Take as much space as you need for each problem. You'll tell us where your solutions are when you submit your paper to gradescope.
3. Export the completed assignment as a PDF file for upload to gradescope.
4. On gradescope, upload only **one** copy per partnership. (Instructions for uploading to gradescope will be posted on the HW2 page of the course website.)

1. **Pizza! (22 points).**

   Suppose we have an infinitely large pizza (every computer scientist's dream), and we wish to determine what is the maximum number of pizza pieces that we can produce, using only n perfectly straight (and infinitely long) cuts. The figure below illustrates the most pieces we can make, for $n = 1, 2, 3$ and 4 cuts. Pay careful attention to how many/which pieces are divided (and thus added) by the newest cut.



   Let $S(n)$ denote the number of slices, given n cuts across the pie. From the illustration above, we see that $S(1) = 2$, and $S(2) = 4$, and we can also infer from the description that $S(0) = 1$.

   (a) **[1 marks]** What is the value of $S(5)$? $\boxed{16}$

   (b) **[3 marks]** Complete the following recurrence relation for $S(n)$:

$$S(0) = 1$$

$$S(n) = S(\boxed{n-1}) + \boxed{n} \qquad \text{for } n > 0$$

   (c) **[6 marks]** What is $S(n)$ as a function of n? Your solution should not be a recurrence, contain a summation, or use asymptotic notation.

$$\boxed{1 + \frac{n(n+1)}{2}}$$

$$
\begin{aligned}
S(n) &= S(n-1) + n \\
&= S(n-2) + (n-1) + n \\
&\vdots \\
&= S(n-k) + \sum_{i=0}^{k-1} (n-i) \\
&\vdots \\
&= S(0) + \sum_{i=0}^{n-1} (n-i) \\
&= 1 + \sum_{j=1}^{n} j \\
&= 1 + \frac{n(n+1)}{2}
\end{aligned}
$$

Prove using induction, that the maximum number of pizza pieces made using n cuts ($n \geq 0$) is given by your closed form in part (c).

(d) **[3 marks]** Base case: For $n =$ $\boxed{0}$, (complete the rest)

The recurrence gives $S(n) = 1$ and the closed form gives $S(n) = 1 + \frac{0(1)}{2} = 1$.

(e) **[3 marks]** State your inductive hypothesis:
Answer: For any $0 \leq j < n$, $S(j) = 1 + \frac{j(j+1)}{2}$

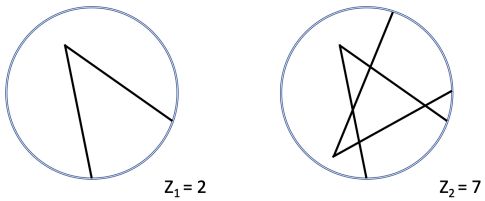
(f) **[2 marks]** State for your inductive step, the number of pieces in the current step ($S(n)$), in terms of the number of pieces from the previous step ($S(n-1)$):
Answer: $S(n) = S(n-1) + n$

(g) **[4 marks]** Complete your inductive step in the space below:
Note, by IH, $S(n-1) = 1 + \frac{(n-1)n}{2}$.

$$
\begin{aligned}
S(n) &= S(n-1) + n && \texttt{recurrence} \\
&= 1 + \frac{(n-1)n}{2} + n && \texttt{IH} \\
&= 1 + \frac{n(n+1)}{2} && \texttt{algebra}
\end{aligned}
$$

2. **Still Hungry! (12 points).**

Suppose we still have an infinitely large pizza, and we still wish to determine what is the maximum number of pizza pieces that we can produce, but this time our cuts are shaped like Vs. The figure below illustrates the most pieces we can make, for $n = 1$,and $n = 2$ V-shaped cuts. Note that it's important to remember that the size of the pizza is so large that you can consider the cuts to be infinitely long, and the points of the Vs can be placed anywhere that maximizes the number of regions.



Let $Z(n)$ denote the number of slices, given n V-shaped cuts in the pie. From the illustration above, we see that $Z(1) = 2$, and $Z(2) = 7$, and we can also infer from the description that $Z(0) = 1$.

(a) **[2 marks]** What is the value of $Z(3)$? $\boxed{16}$

(b) **[5 marks]** Write an expression for $Z(n)$ in terms of function $S(\cdot)$ from the previous problem. This will require lots of sketching! It may help to realize that a drawn V is just half of a drawn X.

$$Z(n) = \boxed{S(2n) - 2n}$$

This comes from the fact that a V shape arises from drawing two lines that cross, and then erasing half of each line. The number of regions you eliminate when doing that erasing is 2 for each line, which explains the $-2n$ term. See pages 7-8 in Don Knuth's concrete math book.

(c) **[5 marks]** What is $Z(n)$ as a function of n? Your solution should not be a recurrence, contain a summation, or use asymptotic notation.

$$\boxed{2n^2 - n + 1}$$

3. **Dinner Rolls [10 points].** In each case, you will state an exact (not asymptotic) solution. Please show us the details of your work. Place your final formula in the box.

   (a) (5 points) For this problem you may assume that n is a power of 2.

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 2T(\frac{n}{2}) + 1 & \text{if } n > 1 \end{cases}$$

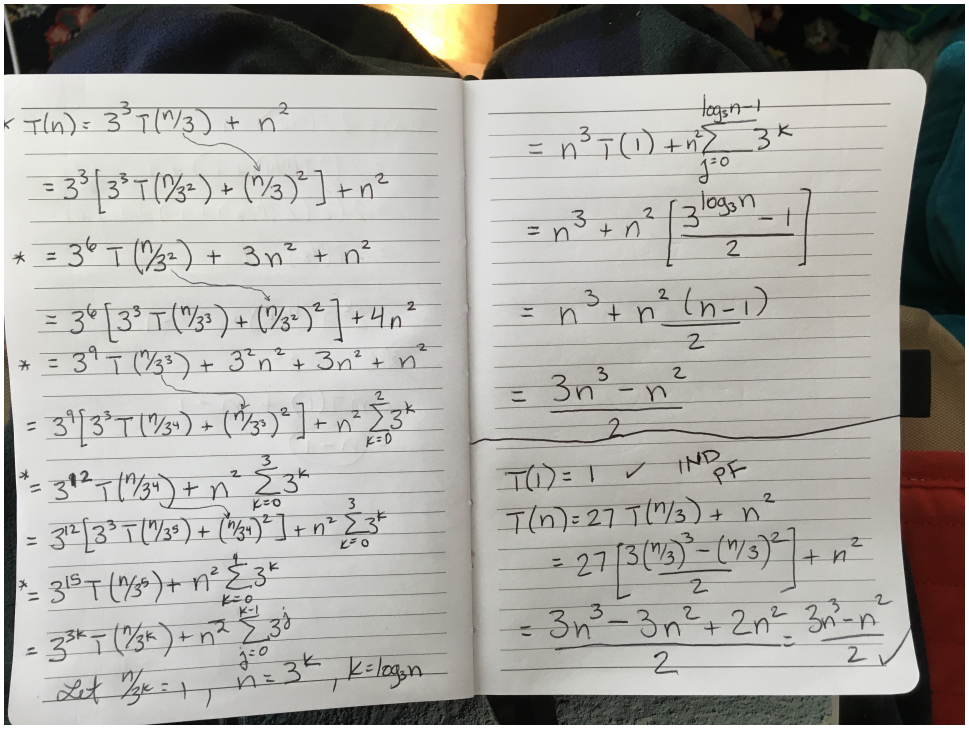
$$\boxed{T(n) = n - 1}$$

$$\begin{aligned} T(n) &= 2T(\frac{n}{2}) + 1 \\ &= 4T(\frac{n}{4}) + 3 \\ &= 8T(\frac{n}{8}) + 7 \\ &\vdots \\ &= 2^k T(\frac{n}{2^k}) + 2^k - 1 \\ &\vdots \quad (\texttt{let n=}2^k) \\ &= nT(1) + n - 1 \\ &= n - 1 \end{aligned}$$

   (b) (5 points) For this problem you may assume that n is a power of 3.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 27T(\frac{n}{3}) + n^2 & \text{if } n > 1 \end{cases}$$

$$\boxed{T(n) = \frac{3n^3 - n^2}{2}}$$



(c) (5 points) Assume that $\exists k \in Z, n = 2^{2^k}$.

$$T(n) = \begin{cases} 1 & \text{if } n \leq 2 \\ T(\sqrt{n}) + 1 & \text{if } n > 2 \end{cases}$$

$$\boxed{T(n) = 1 + \log\log n}$$

Achieved via unrolling, and letting $n^{\frac{1}{2^k}} = 2$. Easy proof by induction.

(d) Assume that $\exists k \in Z, n = 2^{2^k}$. As a hint, try the following substitution: $S(n) = \frac{T(n)}{n}$.

$$T(n) = \begin{cases} 200 & \text{if } n \leq 2 \\ \sqrt{n}T(\sqrt{n}) + 100n & \text{if } n > 2 \end{cases}$$

$$\boxed{T(n) = 100n + 100n\log\log n}$$

Following the hint, define $S(n) = \frac{T(n)}{n}$ and find $S(n) = S(n^{\frac{1}{2}}) + 100$. Solve *this* recurrence exactly as you did for the previous problem, and then multiply it by n to get the desired result.

4. **Quicksort (20 points).**

   In practice, quicksort is one of the fastest sorting algorithms.

   1. Pick a **pivot** (e.g. the first element)

bee	bed	cab	ace	dad	baa	add	ebb

   2. Reorder the array such that all elements $<$ pivot are to its left, and all elements $\geq$ pivot are to its right.

add	bed	ace	baa		bee		cab	dad	ebb
left_partition							right_partition		

   3. Recursively sort each partition.

```
void qsort(vector<string> & x, int lo, int hi) {
  if (lo >= hi) return;
  int p = lo;
  for( int i=lo+1; i <= hi; i++ )
    if( x[i] < x[lo] ) { p++; swap(x[p], x[i]); }
  swap(x[lo], x[p]);
  qsort(x, lo, p-1);
  qsort(x, p+1, hi);
}
void quicksort(vector<string> & x) {
  qsort(x, 0, x.size()-1);
}
```

   (a) (4 points) Show the contents of the array `x` at the start of each iteration of the for-loop, after the for-loop, and after the swap, in the call `qsort(x, 0, 3)` on the input shown below:

	`lo=0`	1	2	`hi=3`	4	5	6	7
start of iter $i = 1$	add	bed	ace	baa	bee	cab	dad	ebb
start of iter $i = 2$	**add**	**bed**	**ace**	**baa**	bee	cab	dad	ebb
start of iter $i = 3$	**add**	**ace**	**bed**	**baa**	bee	cab	dad	ebb
end of iter $i = 3$	**add**	**ace**	**bed**	**baa**	bee	cab	dad	ebb
after `swap(x[lo],x[p])`	**ace**	**add**	**bed**	**baa**	bee	cab	dad	ebb

   (b) (4 points) Complete the following loop invariant by providing the range of indices for which the statement is true. Note that `x[` a ... b `]` is empty if $a > b$.

   At the start of iteration i of the for-loop in `qsort`:

   (A) `x[lo+1 ... p]` $<$ `x[lo]` and

   (B) `x[p+1 ... i-1]` $\geq$ `x[lo]`

(c) (4 points) To prove the loop invariant, we use induction on the iteration index i. In the base case, when `i=lo+1` and `p=lo`, the ranges for parts (A) and (B) are empty and the invariant is trivially true. For the inductive step, going from iteration i to iteration $i' = i + 1$:

i. If `x[i] < x[lo]`, the value of p in the next iteration is $p' = p+1$ and the order of strings in the vector x changes. If we replaced `swap(x[p],x[i])` with `x[p]=x[i]`, which part of the invariant would be false at iteration $i + 1$, (A) or (B)?

B

ii. If `x[i] >= x[lo]`, which part of the invariant at iteration $i + 1$ is *the same* as at iteration i, (A) or (B)?

A

(d) (4 points) What is the worst case running time of Quicksort on inputs of size n? First, show the recurrence relation representing the running time for a worst case input. Then give the asymptotic result.

$$T(n) = T(n-1) + \Theta(n), T(1) = \Theta(1)$$

$$\Theta(n^2)$$

(e) (4 points) Suppose in every recursive call to sort a subarray, we choose a pivot, in time proportional to the size of the subarrary, that is never one of the smallest 1/4 or largest 1/4 elements. What is the worst case running time of Quicksort on inputs of size n in this case? Give the recurrence and the asymptotic closed form.

$$T(n) = T(3n/4) + T(n/4) + \Theta(n), T(1) = \Theta(1)$$

$$\Theta(n \log n)$$