CS 180 - Homework 1

Darren Tsang, Discussion 1I

Produced on Sunday, Apr. 19 2020 @ 06:28:35 PM

Question 1

The statement is true.

Proof: Assume, for the sake of contradiction, that there exists a stable matching S that does not contain the pair *(m, w)*. Then, without loss of generality, we know that there are pairs (m, w') and (m', w) in S. We know that m is ranked first on the preference list of w and w is ranked first on the preference list of m. More specifically, m ranks w over w' and w ranks m over m'. Thus, S is an unstable matching, a contradiction.

Question 2

Part a.

It suffices to show that there is a stable matching S such that $(m_1, w_1), (m_2, w_2) \in S$ because it shows that w_1 is a valid partern for m_1 and w_2 is a valid partner for m_2. Furthermore, we know that the Gale-Shapley algorithm with men proposing yields men optimal matching, meaning that every m is matched with their best valid partner. Note that m_1 ranks w_1 over every other w and m_2 ranks w_2 over every other w.

Claim: There exists a stable matching S such that $(m_1, w_1), (m_2, w_2) \in S$.

Proof of Claim: Assume, for the sake of contradiction, that there exists a stable matching S_1 such that $(m_1, w_1), (m_2, w_2) \notin S_1$. Then, without loss of generality, $(m_1, w'), (m_2, w''), (m', w_1), (m'', w_2) \in S_1$. We know that m_1 prefers w_1 over w' and w_1 prefers m_1 over m'. Thus, S_1 is unstable, a contradiction.

Part b.

Assume, for the sake of contradiction, that there exists a stable matching S_1 such that $(m_1, w_1), (m_2, w_2) \notin S_1$ and $(m_1, w_2), (m_2, w_1) \notin S_1$. Then, without loss of generality, $(m_1, w'), (m_2, w''), (m', w_1), (m'', w_2) \in S_1$. We know that m_1 prefers w_1 over w' and w_1 prefers m_1 over m'', which means that S_1 is an unstable matching, a contradiction.

Question 3

$f_1(n) = n^{2.5}$

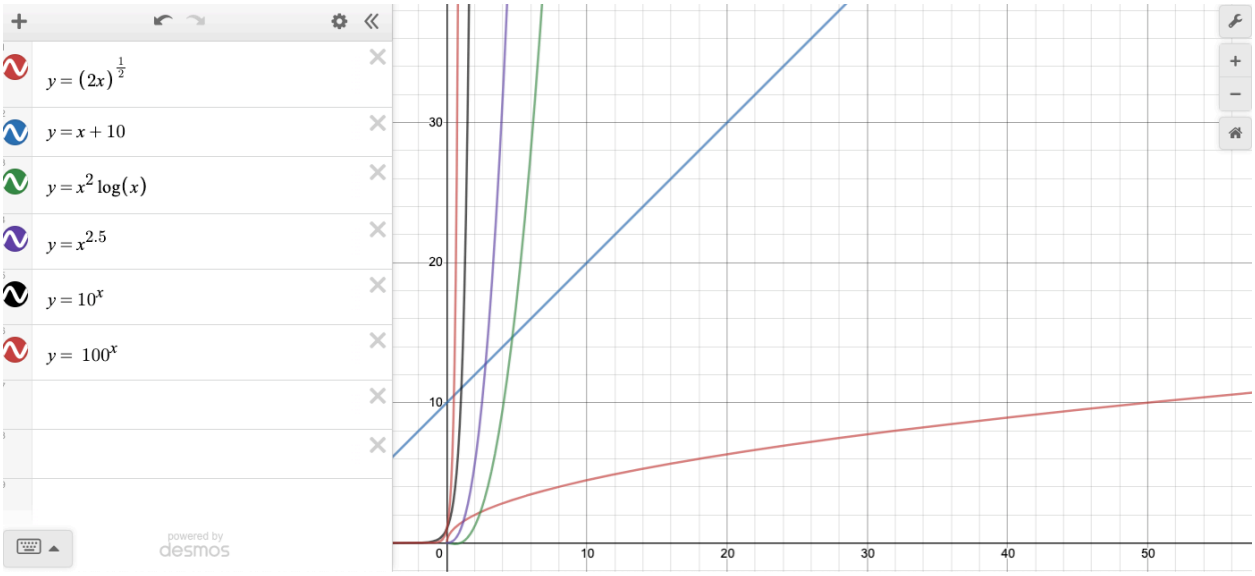
$f_2(n) = \sqrt{2n}$

$f_3(n) = n + 10$

$f_4(m) = 10^n$

$f_5(n) = 100^n$

$f_6(n) = n^2 log(n)$



The order is $f_2(n), f_3(n), f_6(n), f_1(n), f_4(n), f_5(n)$ because from the graph above, we can see that $\sqrt{2x} < x + 10 < x^2 log x < x^{2.5} < 10^x < 100^x, \forall x > 10$.

Question 4

Part a.

We know that $f(n) = O(g(n))$, which means $\exists c > 0, n_0 \geq 0$ such that $f(n) \leq c \cdot g(n), \forall n \geq n_0$. This can be rewritten as $\exists c > 0, n_0 \geq 0$ such that $\frac{f(n)}{c} \leq g(n), \forall n \geq n_0$. Since $\frac{1}{c} > 0$, I have successfully showed $g(n) = \Omega(f(n))$.

Part b.

We know that $f(n) = O(g(n))$, which means $\exists c_1 > 0, n_0 \geq 0$ such that $f(n) \leq c_1 \cdot g(n), \forall n \geq n_0$. Also, it is true that $g(n) = O(g(n))$ (choose constant to be 1). Thus, if we were to choose $c = max(c_1, 1)+1, f(n) \cdot g(n) \leq cg(n) \cdot g(n) = c \cdot g(n)^2, \forall n \geq n_0$. Thus, $f(n) \cdot g(n) = O(g(n)^2)$, as desired

Part c.

The statement is false.

Counterexample: Let $f(n) = 2log_2 n$ and $g(n) = log_2 n$. We can see that $f(n) = O(g(n))$ because $f(n) = 2log_2 n < 3g(n) = 3log_2 n, \forall n > 0$. Then, $2^{f(n)} = 2^{2log_2 n} = n^2$ and $2^{g(n)} = 2^{log_2 n} = n$. At this point, it suffices to show that $\forall c, \nexists n_0$ such that $n^2 < c \cdot n, \forall n > n_0$.

Proof of claim: Let $c > 0$. Then, $n^2 > c \cdot n$, $\forall n > c$. Thus, $\nexists n_0$ such that $n^2 < c \cdot n$, $\forall n > n_0$.

Question 5

isPalindrome(ListNode head){
  get to the ceiling(number of elements/2)-th element of the list
  reverse the list starting from the element found above to the end of the list
  denote list1 as the first half that was not reversed and list2 as the reversed list

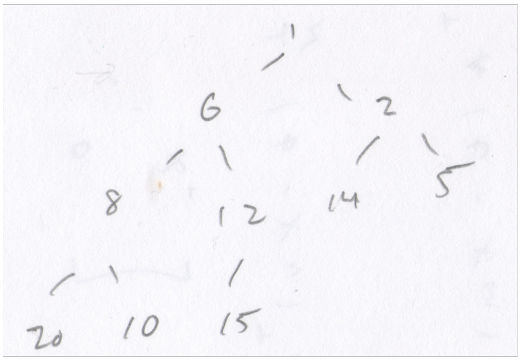
  iteratively compare the values of the list1 to list2 up until list1 reaches the head of list2
}

The algorithm above is "splitting" the list into two: one below ceiling(number of elements/2) and one above the ceiling(number of elements/2). Then, it will compare the two lists iteratively. If there is a difference when comparing, false will be returend. The comparisons will stop when the first list gets to the beginning of the second list. If it gets through all the comparisons without returning false, true will be returned.

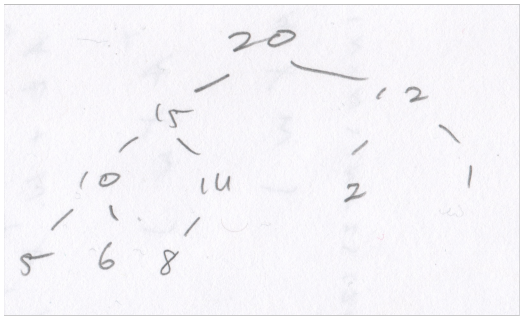
The algorith is in $O(n)$ time because it only traverses the list once. Also, it only uses $O(1)$ additional memory because we are only storing points to help iterate through the two lists; we use the same amount of pointers every time, which is why it is $O(1)$.

Question 6

Part a.



Part b.



Part c.

```
Medium Heap push(newValue):
   if maxHeap is empty
      maxHeap.push(newValue)
   else if newInteger <= mediumHeap.findMedium()
      maxHeap.push(newValue)
   else
      minHeap.push(newValue)

   if size of maxHeap - size of minHeap >= 2
      minHeap.push(maxHeap.pop())
   else if size of minHeap - size of maxHeap >= 2
      maxHeap.push(minHeap.pop())
end push
```

```
Medium Heap findMedium():
  if size of maxHeap = size of minHeap
     return average of maxHeap.findMax() and minHeap.findMin()
  else if size of maxHeap > size of minHeap
     return maxHeap.findMax()
  else
     return minHeap.findMin()
end findMedium
```

We are using a min heap and a max heap to implement the medium heap. The min heap will be storing the values that are bigger than the current median, and the max heap will be storing the values of that are smaller than the current median. From lecture, we know that finding the max/min of a heap can be doing in $O(1)$ time and pushing/popping from a heap can be done in $O(logn)$ time. Thus, the add function and findMedium function I designed above is in $O(logn)$ time and $O(1)$ time, repsectively.

In the findMedium, we will output the the root fo the heap that has a larger length (the difference between the length of the two heaps will be at most 1 because we are making sure that that is the case at the end of push). If the two lengths are the same, the medium will be the average of the roots of both heaps.