University of Waterloo
CS240E, Winter 2021
Assignment 1

Due Date: Wednesday, January 27, 2021 at 5pm

The integrity of the grade you receive in this course is very important to you and the University of Waterloo. As part of every assessment in this course you must read and sign an Academic Integrity Declaration (AID) before you start working on the assessment and submit it **before the deadline of January 27** along with your answers to the assignment; i.e. **read, sign and submit A01-AID.txt now or as soon as possible**. The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please read `http://www.student.cs.uwaterloo.ca/~cs240e/w21/guidelines/guidelines.pdf` for guidelines on submission. **Each written question solution must be submitted individually to MarkUs as a PDF** with the corresponding file names: a1q1.pdf, a1q2.pdf, ... , a1q6.pdf .

It is a good idea to submit questions as you go so you aren't trying to create several PDF files at the last minute. **Remember, late assignments will not be marked but can be submitted to MarkUs after the deadline for feedback if you email cs240e@uwaterloo.ca and let the ISAs know to look for it. (#3).**

For all questions, you may use facts from calculus, but review briefly the name of the fact and exactly what it is stating. (Refer to how the notes review concavity or l'Hôpital's rule for examples.)

Question 1 [1+2=3 marks]

Dr. I. M. Smart has invented a new class of functions, denoted $O'(f)$: A function $f(n)$ is in $O'(g(n))$ if there is a constant $c > 0$ such that $f(n) \leq cg(n)$ for all $n > 0$.

All functions map positive integers to positive integers.

a) Prove that $f(n) \in O'(g(n))$ implies that $f(n) \in O(g(n))$.

b) Prove that $f(n) \in O(g(n))$ implies that $f(n) \in O'(g(n))$.

Question 2 [7(+2) marks]

Consider the following (rather strange) code-fragment:

```
mystery (int n) {
// pre: n >= 2
    compute L := floor( log (n) )
    if L is even print all subsets of {1,...,L}
    else print all subsets of {1,...,2L}
}
```

For example, for $n = 17$, we have $L = \lfloor \log(17) \rfloor = 4$, so the code prints all 16 subsets of $\{1, 2, 3, 4\}$.

Let $f(n)$ be the run-time of this code. You may assume that computing $\log(n)$ takes $\Theta(1)$ time, and printing the subsets of $\{1, \ldots, k\}$ takes $\Theta(2^k)$ time.

~~Show that cannot exist a constant d such that $f(n) \in \Theta(n^2)$.~~

Show that $f(n) \notin \Theta(n^2)$.

Bonus: Show that cannot exist a constant d such that $f(n) \in \Theta(n^d)$.

Question 3 [2+6+4=12 marks]

To reduce the height of the heap one could use a d-way heap. This is a tree where each node contains up to d children, all except the bottommost level are completely filled, and the bottommost level is filled from the left. It also satisfies that parents are bigger than all their children.

a) Explain how to store a d-way heap in an array A of size $O(n)$ such that the root is at $A[0]$. Also state how you find parents and children of the node stored at $A[i]$. You need not justify your answer.

b) What is the height of a d-ary heap on n nodes? Give a tight asymptotic bound that depends on d and n. You may assume that n and d sufficiently big (e.g. $d \geq 3$ and $n \geq 10$).

c) Assume that $n \geq 4$ is a perfect square. What is the height of a d-ary heap for $d = \sqrt{n}$? Give an exact bound (i.e., not asymptotic).

Question 4 [3+7=10 marks]

How would you implement *increaseKey*(v, k) in a binomial heap? The method is given as parameter a node v and the new value k that its key should have.

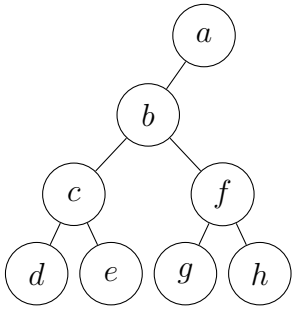
a) Prof. Dodo thinks he can implement this using *fix-up* as follows:

Show that Prof. Dodo is incorrect. Thus, give an example of a flagged tree that satisfies the binomial-heap-order property, indicate a node v and a key $k > v.key$, and show that calling `increaseKey(v,k)` with the above code would result in a flagged tree that does not satisfy the binomial-heap-order property. (Try to keep your tree small, no more than 16 nodes.)

Algorithm 1: *increaseKey*(v, k)

```
1 if (k > v.key()) then
2     v.key ← k;
      // perform fix-up
3     while p ← v.parent is not NIL and p.key < v.key do
4         swap key-value pairs of v and p;
5         v ← p;
```

The tree on the right is drawn here so that you can see (in the published LaTeX-file) how one can easily draw a tree using `tikz`. You don't have to draw your tree with `tikz`, but are encouraged to do so.



b) Give an implementation of *increaseKey* in a binomial heap that has worst-case run-time $O(\log n)$.

Question 5 [1+1+2(+5) = 4(+5) marks]

Consider the following pseudo-code, which is a variant of *quick-sort*:

Algorithm 2: *mysteryQS*$(A, n \leftarrow A.size)$

```
   Input: array A of size at least n
1  if n > 1 then
2      p ← choose-pivot(A);
3      i ← partition(A, p);
4      mysteryQS(A[0, 1, ..., i − 1]);
5      mysteryQS(A[i + 1, ..., n − 1]);
6      for j = 1 to i do
7          for k = i + 1 to n − 1 do
8              mystery(A, j, k) ;          // This uses one key-comparison
9  else if n = 1 then
10     mystery(A, 0, 0);
```

Here *choose-pivot* always returns $n - 1$, and *partition* is Hoare's partition-routine and uses n key-comparisons. Let $T^{\alpha}(n)$ be the number of key-comparisons used by *mysteryQS*

(for various scenarios α). You may assume that n is divisible as needed (i.e., ignore rounding issues).

a) State the recursive formula for $T^{\alpha}(n)$ if we know the pivot-index i. No explanation needed.

b) Show that $T^{\text{worst}}(n) \in \Omega(n^2)$.

c) Show that $T^{\text{best}}(n) \in O(n^2 \log n)$.

d) (Bonus) Give asymptotically tight bounds for $T^{\text{worst}}(n)$ and $T^{\text{best}}(n)$. Doing this question will also give you the credit for (b) and (c).

Hint: Study the function $f(x) = x(n - x)$. Where is its minimum and maximum?