Solutions for Data Structures and Algorithms Spring 2023 — Problem Sets

By Dmitriy Okoneshnikov, B22-DSAI-04

February 18, 2023

Homework 1. Problem set

Asymptotics (32 points)

1. Prove or disprove the following statements. You must provide a formal proof. You may use the definitions of the asymptotic notations as well as their properties and properties of common functions, **as long as you properly reference** them (e.g. by specifying the exact place in Cormen where this property is introduced)!

   (a) $n^3 \log n = \Omega(3n \log n)$
   Solution.
   By definition [Cormen, Section 3.2] $f(n) = \Omega(g(n))$ means that $\exists c > 0, n_0 > 0 : \forall n > n_0 : f(n) \geq cg(n) \geq 0$.
   $n^3 \log n \geq 3cn \log n$
   Let $n > n_0 > 1$
   $n^3 \geq 3cn$
   $n^2 \geq 3c$
   $c \leq \frac{n^2}{3}$
   This inequality holds true, e.g., when $c = 2, n = 3$.
   Q.E.D.

   (b) $n^{\frac{9}{2}} + n^4 \log n + n^2 = O(n^4 \log n)$
   Solution.
   By definition [Cormen, Section 3.2] $f(n) = O(g(n))$ means that $\exists c > 0, n_0 > 0 : \forall n > n_0 : cg(n) \geq f(n) \geq 0$.
   $n^{\frac{9}{2}} + n^4 \log n + n^2 \leq cn^4 \log n$
   $n^{\frac{1}{2}} + \log n + \frac{1}{n^2} \leq c \log n$
   Let $n > n_0 > 1$
   $\frac{n^{\frac{1}{2}}}{\log n} + 1 + \frac{1}{n^2 \log n} \leq c$
   But $n^{\frac{1}{2}}$ grows faster that $\log n$ [Cormen, Section 3.3, Equation 3.24]. Therefore, we can't choose any c and n that will hold the inequality true.
   The statement is not true.

   (c) $6^{n+1} + 6(n+1)! + 24n^{42} = O(n!)$
   Solution.
   By definition [Cormen, Section 3.2] $f(n) = O(g(n))$ means that $\exists c > 0, n_0 > 0 : \forall n > n_0 : cg(n) \geq f(n) \geq 0$.

$6^{n+1} + 6(n+1)! + 24n^{42} \leq cn!$
$\frac{6^{n+1}}{n!} + 6(n+1) + \frac{24n^{42}}{n!} \leq c$
But it is obvious that there are no c, n_0 that will satisfy the inequality. As for any c, n_0 there will be such $n > n_0$ that $6(n+1) > c$.
The statement is not true.

(d) There exists a constant $\varepsilon > 0$ such that $\frac{n}{\log n} = O(n^{1-\varepsilon})$
By definition [Cormen, Section 3.2] $f(n) = O(g(n))$ means that $\exists c > 0, n_0 > 0 : \forall n > n_0 : cg(n) \geq f(n) \geq 0$.
$\frac{n}{\log n} \leq cn^{1-\varepsilon}$
$\frac{n^{\varepsilon}}{\log n} \leq c$
$n^{\varepsilon} \leq c \log n$
The latter means $n^{\varepsilon} = O(\log n)$, but n^{ε} grows faster that $\log n$ [Cormen, Section 3.3, Equation 3.24].
The statement is not true.

2. For each of the following recurrences, apply the master theorem yielding a closed form formula. You must specify which case is applied, explicitly check the necessary conditions, and provide final answer using Θ-notation. If the theorem cannot be applied you must provide justification.

(a) $T(n) = 2T(\frac{n}{3}) + \log n$
Solution.
Case #1 is used [Cormen, Section 4.5], which states that if $\exists \varepsilon > 0 : f(n) = O(n^{\log_b a - \varepsilon})$, then $T(n) = \Theta(n^{\log_b a})$.
$log(n) = O(n^{\log_3 2 - \varepsilon})$
From [Cormen, Section 3.3, Equation 3.24] we know that $log^b n = o(n^{\alpha})$, where $\alpha > 0$. Therefore, $log^b n = O(n^{\alpha})$. Then we need any ε that will make the logarithm positive. For example, $\varepsilon = 0.5$.
The case's condition is satisfied, therefore, $T(n) = \Theta(n^{\log_3 2})$.
Answer. $\Theta(n^{\log_3 2})$

(b) $T(n) = 3T(\frac{n}{9}) + \sqrt{n}$
Solution.
Case #2 is used [Cormen, Section 4.5], which states that if $\exists k \geq 0 : f(n) = \Theta(n^{\log_b a} \lg^k n)$, then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
$\sqrt{n} = \Theta(n^{\frac{1}{2}} \lg^k n)$
Let $k = 0$
$\sqrt{n} = \Theta(\sqrt{n})$. It is true as $f(n) = \Theta(f(n))$ [Cormen, Section 3.2]
The case's condition is satisfied, therefore, $T(n) = \Theta(\sqrt{n} \lg n)$
Answer. $\Theta(\sqrt{n} \lg n)$

(c) $T(n) = 4T(\frac{n}{9}) + \sqrt{n} \log n$
Solution.
Case #1 is used [Cormen, Section 4.5], which states that if $\exists \varepsilon > 0 : f(n) = O(n^{\log_b a - \varepsilon})$, then $T(n) = \Theta(n^{\log_b a})$.
$\sqrt{n} \log n = O(n^{\log_9 4 - \varepsilon})$
Let $\varepsilon = 0.5$
$\sqrt{n} \log n = O(n^{\log_9 3.5})$
Using the definition of O [Cormen, Section 3.2]

$n^{0.5} \log n \leq cn^{\log_9 3.5}$
$n^{0.5-\log_9 3.5} \log n \leq c$
$\frac{1}{n^{\log_9 3.5-0.5}} \log n \leq c$
$\log n \leq cn^{\log_9 3.5-0.5}$
Due to the definition of O [Cormen, Section 3.2], the latter means $\log n = O(n^{\log_9 3.5-0.5})$. But from [Cormen, Section 3.3, Equation 3.24] we know that $log^b n = o(n^\alpha)$, where $\alpha > 0$. Therefore, $log^b n = O(n^\alpha)$.
The case's condition is satisfied, therefore, $T(n) = \Theta(n^{\log_9 4})$
Answer. $\Theta(n^{\log_9 4})$

(d) $T(n) = 5T(\frac{n}{5}) + \frac{n}{1000}$
Solution.
Case #2 is used [Cormen, Section 4.5], which states that if $\exists k \geq 0 : f(n) = \Theta(n^{\log_b a} \lg^k n)$, then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
$\frac{n}{1000} = \Theta(n^{\log_5 5} \lg^k n)$
$\frac{n}{1000} = \Theta(n \lg^k n)$
Let $k = 0$
$\frac{n}{1000} = \Theta(n)$
The latter is true as we can choose $c_1 = \frac{1}{1000}$, $c_2 = 1$, $n_0 = 1$ such that $0 \leq c_1 n \leq \frac{n}{1000} \leq c_2 n$ for all $n > n_0$ [Cormen, Section 3.2], therefore, $0 \leq \frac{1}{1000} \cdot n \leq \frac{n}{1000} \leq 1 \cdot n$.
The case's condition is satisfied, therefore, $T(n) = \Theta(n \lg n)$
Answer. $\Theta(n \lg n)$

Segmented List (18 points)

Recall the methods of the List ADT:

```
public interface List<E>
{
    int size();
    boolean isEmpty()
    void add(int position, E element);
    E remove (int position);
    E get(int position);
    E set(int position, E element);
}
```

Consider a `SegmentedList` implementation, that consists of a doubly linked list of arrays (segments) of fixed capacity k:

1. all segments, except the last one, must contain at least $\lfloor \frac{k}{2} \rfloor$ elements;
2. when adding at the end of `SegmentedList`, the element is added in the last segment, if it is not full; otherwise, a new segment of capacity k is added at the end of the linked list;
3. when adding x at any position, the doubly linked list is scanned from left to right (or right to left) to find the corresponding segment; if the segment is full, it is split into two segments with $\frac{k}{2}$ elements each, and x is inserted into one of them, depending on the insertion position.
4. when removing an element, only elements in its segment are shifted; if the segment size becomes less than $\lfloor \frac{k}{2} \rfloor$, then we rebalance:
   (a) if it is the last segment, do nothing;

(b) if it is a middle segment s_i and the sum $m = size(s_{i-1}) + size(s_i) + size(s_{i+1}) < 2k$, then replace these three segments with two segments of size $\frac{m}{2}$;

(c) if it is a middle segment s_i and the sum $m = size(s_{i-1}) + size(s_i) + size(s_{i+1}) \geq 2k$, then replace these three segments with three segments of size $\frac{m}{3}$;

(d) if it is the first segment, and there exist two segments after, then proceed similarly to previous two cases;

(e) if it is the first segment and only one segment after it exists, move as many elements from the last segment to the first one as possible; if the last segment becomes empty, remove it.

Perform the following analysis for the `SegmentedList`:

1. Argue that the worst case time complexity of `add(i, e)` is $O(\frac{n}{k} + k)$.

   Solution.

   To add an element to a position, we need to first scan the doubly linked lists to find the segment, it will take in the worst case $O(\frac{n}{k})$, because there are $\frac{n}{k}$ segments. If the segment is full, then it needs to be split into two segments, taking in the worst case $O(k)$, because we need to move the second half of the splitted segment. Therefore, the total worst case time complexity of `add(i, e)` is $O(\frac{n}{k} + k)$.

   Q.E.D

2. Argue that the worst case time complexity of `remove(i)` is $O(\frac{n}{k} + k)$.

   Solution.

   To remove an element at a position, we need to first scan the doubly linked lists to find the segment, it will take in the worst case $O(\frac{n}{k})$, because there are $\frac{n}{k}$ segments. Then, we need to shift the elements in the corresponding segment, which takes in the worst case $O(k)$, because we need to move at most k elements. If the size of the segment becomes less than $\lfloor \frac{k}{2} \rfloor$, we need to rebalance, which takes in the worst case $O(k)$. Therefore, the total worst case time complexity of `remove(i)` is $O(\frac{n}{k} + k)$.

   Q.E.D.

3. What value of k should be chosen in practice? Why?

   Answer.

   A small value of k would make adding and removing elements faster as we need to shift fewer elements, but it would increase the number of segments, which would require more memory.

   A larger value of k would reduce the number of segments, but it would make adding and removing elements slower as we would need to shift more elements.

   So the optimal value of k can vary depending on the needs and the number of elements.

Segmented Queue (+1% extra credit)

Consider `SegmentedList` used as a Queue:

1. we enqueue (`offer(e)`) elements by adding them to the end of the segmented list;

2. we dequeue (`poll()`) elements by removing the first element of the segmented list;

3. in an attempt to make `remove` more efficient, we do not perform shifts when removing an element in a segment (so there can be a gap on the left of the cells where values are stored); `add` will create a new segment when it reaches the right end of the last segment, regardless of whether there is empty space to the left;

4. rebalancing part of `remove` remains, to keep the invariant that each segment has at least $\lfloor \frac{k}{2} \rfloor$ elements.

Perform amortised analysis for arbitrary sequences of `offer(e)` and `poll()` operations applied to an initially empty queue, implemented using `SegmentedList` in a way described above. Show that amortized cost for any such sequence of length N is $O(N)$ (does not depend on the value of k). You **must** use the accounting method or the potential method.

NO SOLUTION.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms, Fourth Edition.* The MIT Press 2022