Advanced Algorithms, Homework 3

Nathan Stouffer Kevin Browder

due: 17 September 2020

This homework assignment is due on 3 September 2020, and should be submitted as a single PDF file to to Gradescope.

General homework expectations:

- Homework should be typeset using LaTex.
- Answers should be in complete sentences and proofread.
- This homework can be submitted as a group.

CSCI 432 Problem 3-1

Collaborators: *n/a*

Work in a group of size ≥ 2. Explain your strategy for working in a group.

Answer TODO: your answer goes between these lines

CSCI 432 Problem 3-2

Collaborators: *Nathan Stouffer and Kevin Browder*

Your group should make at least five contributions to the Piazza board. A contribution can be either asking a relevant question, responding to another student's question, responding to an instructor's question, or choosing a question from Chapter 1 and attempting to solve it, then describing where you get stuck in answering it.

Answer Our groups contributions are:

1. (TODO: state the problem number, name of poster, and date/time). TODO: copy the post here.
2. (TODO: state the problem number, name of poster, and date/time). TODO: copy the post here.
3. (TODO: state the problem number, name of poster, and date/time). TODO: copy the post here.
4. (TODO: state the problem number, name of poster, and date/time). TODO: copy the post here.
5. (TODO: state the problem number, name of poster, and date/time). TODO: copy the post here.

Problem number 3-3 (hw 3), Poster: Nathan Stouffer, Date: 9/15/2020 9:34 am Hello everyone,

Some languages (like java, c, c++ ...) allow any boolean condition to be a test for whether the loop should continue running and also allow any incrementing function. However, the examples that I have seen in this class seem to have a for loop running over a set of integers. Does the real-ram model allow for for loops that have a condition like end ¿ beg with some specified incrementing function?

The reason I ask is because of hw problem 3-3. The question asks that we write an algorithm for binary search using a for loop instead of recursion. If we must run over a set of integers, we can compute an upper bound on how many iterations the for loop will run so it is possible to create a correct for loop. I also think proving termination would be easier when running over a set of integers. However, the first option would be able to exit the for-loop before reaching the upper bound of iterations. This would not improve worst case run time but it would affect the average case run time. Let me know what we are allowed to do with this model of computation.

Another idea would be to break out the loop if a certain condition is true. Is this allowable with our model of computation?

CSCI 432 Problem 3-3

Collaborators: *Nathan Stouffer and Kevin Browder*

Give the algorithm for binary search, using a for loop and no recursion.

1. Describe the problem in your own words, including describing what the input and output is.
2. Describe, in paragraph form, the algorithm you propose.
3. Provide this algorithm in the algorithm environment.
4. Use a decrementing function to prove that the loop terminates.
5. What is the loop invariant? Provide the proof.

Answer

1. The problem that binary search solves is returning the location of a target value in an array. The input to the problem must be a sorted array of comparable items paired with a target value of the same type. The output will either by the index of the target value or -1 to flag that the target is not in the array.
2.
3. Here is the algorithm.

```
BinarySearch(A[1..n], targ)
1.  beg ← 1
2.  end ← n
3.  indx ← -1 // assume value is not in array
4.  for 1..⌈log2(len(A))⌉
5.      mid ← ⌊(beg + end)/2⌋
6.      if (A[mid] = targ)
7.          indx ← mid
8.      if (A[mid] > targ)
9.          end ← mid - 1
10.     if (A[mid] < targ)
11.         beg ← mid + 1
12. return indx
```

4. prove termination
5. Our loop invariant is that $arr[beg] < targ < arr[end]$.

CSCI 432 Problem 3-4

Collaborators: *Nathan Stouffer and Kevin Browder*

Chapter 2, Problem 1b (Generalized SubsetSum).

1. Describe the problem in your own words, including describing what the input and output is.
2. Describe, in paragraph form, the algorithm you propose.
3. Provide this algorithm in the algorithm environment.
4. What is the runtime of your algorithm?
5. Prove partial correctness (that if your algorithm terminates, it is correct).

Answer

1. For the Generalized SubsetSum problem, our input has two parts. First, we have two equally sized arrays X and W containing positive integers paired with another positive integer T. Each value of X has a corresponding weight in W that can be found at the same index (ie weight for $X[i]$ can be found at $W[i]$). The second part of the input is an array called I (which has length $len(X)$ and begins consisting entirely of 0s) paired with an index i (which starts at 1). If it exists, our task is to find the subset of X that sums to T with the heaviest weight. If no subset of X sums to T, we will return $-\infty$. Our output will either be $-\infty$ or a positve integer (the weight of the heaviest subset that sums to T).
2. algorithm description
   The array I contains activations for whether to include the i^{th} element of X in the subset (0 means don't include, 1 means include). The index i denotes the first index of I that has not been yet been called.
3. Here is the algorithm.

```
SubsetSum(X, W, T, I, i)
1.  if (Sum(X, I) > T)
2.      return −∞
3.  if (Sum(X, I) = T)
4.      return Weight(W, I)
5.  skip ← SubsetSum(X, W, T, I, i + 1)
6.  I[i] ← 1
7.  take ← SubsetSum(X, W, T, I, i + 1)
8.  return max(skip, take)

Sum(X, I)
1.  sum ← 0
2.  for i in 1..len(X)
3.      sum ← sum + X[i] ∗ include[i]
4.  return sum

Weight(W, I)
1.  weight ← 0
2.  for i in 1..len(W)
3.      weight ← weight + W[i] ∗ include[i]
4.  return weight
```

4. Towards giving the runtime of our algorithm, we start by giving the run times of Sum and Weight. Let n be the length of X (which matches the lengths of W and I). Both Sum and Weight run in $\Theta(n)$ (since they have constant time operations that run n times). Now we give the worst case recurrence relation for SubsetSeum: $T(n) = \Theta(n) + \Theta(n) + T(n-1) + O(1) + T(n-1) = 2 * T(n-1) + \Theta(n)$.

CSCI 432 Problem 3-5

Collaborators: *Nathan Stouffer and Kevin Browder*

Describe two different data structures that you can use to store a graph. Please give a complete description (i.e., a response of "an array" will not suffice).

Answer TODO: your answer goes between these lines

CSCI 432 Problem 3-6

Collaborators: *Nathan Stouffer and Kevin Browder*

Walk through the exponential time Longest Increasing Subsequence (LIS) algorithm on page 108 for the input: $[1, 7, 6, 11, 3, 11]$.

Walk through the algorithm using the Dynamic Programming algorithm present in Section 3.6.

Answer TODO: your answer goes between these lines

CSCI 432 Problem 3-7

Collaborators: *Nathan Stouffer and Kevin Browder*

What is the closed form of the following recurrence relations? Use Master's theorem to justify your answers:

1. $T(n) = 16T(n/4) + \Theta(n)$
2. $T(n) = 2T(n/2) + n \log n$
3. $T(n) = 6T(n/3) + n^2 \log n$
4. $T(n) = 4T(n/2) + n^2$
5. $T(n) = 9T(n/3) + n$

Note: we assume that $T(1) = \Theta(1)$ whenever it is not explicitly given.

Answer TODO: your answer goes between these lines

CSCI 432 Problem 3-8

Collaborators: *Nathan Stouffer and Kevin Browder*

The skyline problem: You are in Camden, NJ waiting for the ferry across the river to get into Philadelphia, and are looking at the skyline. You take a photo, and notice that each building has the silhouette of a rectangle. Suppose you represent each building b as a triple $(x_b^{(1)}, x_b^{(2)}, y_b)$, where the building can be seen from $x_b^{(1)}$ to $x_b^{(2)}$ horizontally and has a height of y_b. Let `rect(b)` be the set of points inside this rectangle (including the boundary). Let `buildings` be a set of n such triples representing buildings. Design an algorithm that takes `buildings` as input, and returns the skyline, where the skyline is a sequence of (x, y) coordinates defining $\cup_{b \in \texttt{buildings}} \texttt{rect}(b)$. The output should start with $(\min_b x_b^{(1)}, 0)$ and end with $(\max_b x_b^{(1)}, 0)$.

1. Describe the problem in your own words, including describing what the input and output is.
2. Describe, in paragraph form, the algorithm you propose.
3. Provide this algorithm in the algorithm environment.
4. What is the runtime of your algorithm? If you do not know, either give the tightest bounds you know, or provide a decrementing function to show that it does terminate.
5. Prove partial correctness (that if your algorithm terminates, it is correct).

Answer TODO: your answer goes between these lines