CPSC 320 2021W2: Tutorial Quiz 2, Individual Portion

CS ID

Please enter your 4 or 5 digit CSID in this box:

1 Recurrence relations for uneven splits

As stated in class, most divide and conquer algorithms divide the inputs into a number of subproblems that are all (almost) the same size. In this case, we can use the Master theorem to derive the solutions of the recurrence relations that describe their running times. Unfortunately, the Master theorem can not be applied to recurrences that correspond to algorithms that divide their input unevenly. In this tutorial quiz, we consider some of these recurrence relations.

1. Using recursion trees, derive a good *lower bound* on the solution of the following recurrence:

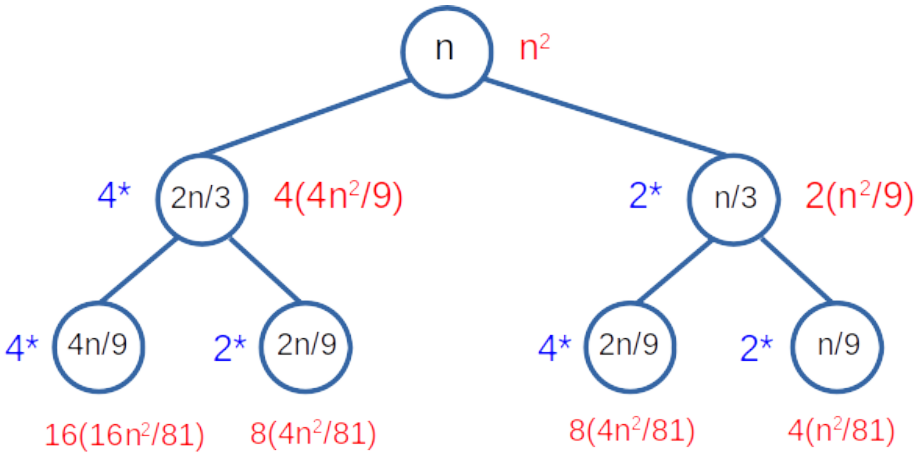
$$T(n) = \begin{cases} 4T(2n/3) + 2T(n/3) + n^2 & \text{if } n \geq 5 \\ \Theta(1) & \text{if} n \leq 4 \end{cases}$$

Hint: it is possible to evaluate the sum of the work on the relevant rows of the tree directly. However, for a less algebraically intense solution, note that this recurrence relation does the same amount of work per level (up to the level that contains the first leaf) as the recurrence

$$T(n) = \begin{cases} xT(n/3) + n^2 & \text{if } n \geq 4 \\ \Theta(1) & \text{if} n \leq 3 \end{cases}$$

for some integer value x.

Solution: We get the following recursion tree (only the first three levels are shown):



- The root of the tree does n^2 work.
- The second level does $4(2n/3)^2 + 2(n/3)^2 = 4(4n^2/9) + 2(n^2/9) = 18n^2/9 = 2n^2$ work.
- The third level does $4n^2$ work.
- And so on...

The first leaf is at level $\log_3 n$. We can sum the work on the first $\log_3 n$ levels directly to get an upper bound. Or we can observe that the recursion tree for the recurrence

$$T'(n) = \begin{cases} 18T(n/3) + n^2 & \text{if } n \geq 4 \\ \Theta(1) & \text{if} n \leq 3 \end{cases}$$

does the same amount of work on each level for the first $\log_3 n$ levels. By the Master theorem, $T'(n) \in \Theta(n^{\log_3 18}) \approx \Theta(n^{2.63})$, which means that $T(n) \in \Omega(n^{\log_3 18})$.

2. Consider once again the recurrence:

$$T(n) = \begin{cases} 4T(2n/3) + 2T(n/3) + n^2 & \text{if } n \geq 5 \\ \Theta(1) & \text{if} n \leq 4 \end{cases}$$

Using recursion trees, derive a good *upper* bound on $T(n)$.

Solution: The recursion tree is identical to the one drawn in the solutions to Question 1.

- The root of the tree does n^2 work.
- The second level does $4(2n/3)^2 + 2(n/3)^2 = 4(4n^2/9) + 2(n^2/9) = 18n^2/9 = 2n^2$ work.
- The third level does $4n^2$ work.
- And so on...

The last leaf is at level $\log_{3/2} n$. We can sum the work on the first $\log_{3/2} n$ levels directly to get an upper bound. Or we can observe that the recursion tree for the recurrence

$$T'(n) = \begin{cases} 4.5T(2n/3) + n^2 & \text{if } n \geq 4 \\ \Theta(1) & \text{if} n \leq 3 \end{cases}$$

does the same amount of work on each level for the first $\log_{3/2} n$ levels. By the Master theorem, $T'(n) \in \Theta(n^{\log_{3/2} 4.5}) \approx \Theta(n^{3.71})$, which means that $T(n) \in \Omega(n^{\log_{3/2} 4.5})$.

CPSC 320 2021W2: Tutorial Quiz 2, Individual Portion

CS ID

Please enter your 4 or 5 digit CSID in this box:

Forgetful oceanographers

The British Columbia department of fisheries is asking one of its oceanographers to take water samples at regular intervals along a portion of the coast line. The samples will then be analyzed to determine the location and amounts of pollutant present in the water. Unfortunately not only is the definition of "regular intervals" left to the oceanographer, but this oceanographer is known to be forgetful, and some of the locations might get skipped. You have been hired to determine efficiently which locations are missing.

The input to your algorithm will be an array A of locations. For simplicity, we will assume each location is a non-negative integer. If the oceanographer didn't forget any location, then the array A will be of the form $[a_1, a_1 + c, a_1 + 2c, \ldots, a_1 + (n-1)c]$, where A has length n (for $n \geq 2$) and c is the (constant) distance between two consecutive locations.

1. Suppose the oceanographer forgot exactly one of the locations, and that it's neither the first one nor the last one. For example the missing location in $[3, 6, 12, 15, 18]$ is 9, while the missing location in $[1, 15, 22, 29, 36]$ is 8. Describe a way to calculate c in constant time.

   Solution: If $n = 2$, then the missing location must lie between $A[0]$ and $A[1]$, and so $c = (A[1] - A[0])/2$. Otherwise c will be minimum of $A[1] - A[0]$ and $A[2] - A[1]$ because the missing location can not be, at the same time, immediately after $A[0]$ and immediately after $A[1]$.

2. Design an algorithm to efficiently find the missing location in the array.

   Solution: We use an algorithm very similar to binary search to locate the missing element. At each iteration of the loop, we maintain the invariant that the missing element will be between `A[low]` and `A[high]`.

```
define locate-missing(A, n):
    if n == 2:
        return (A[0] + A[1]) / 2

    low = 0
    high = n - 1
    c = min(A[1] - A[0], A[2] - A[1])
    while low < high - 1:
        mid = (low + high) // 2
```

```
        if A[mid] - A[low] == c * (mid - low):
            low = mid
        else
            high = mid
    return A[low] + c
```

3. Briefly justify a good asymptotic runtime of your algorithm.

   Solution: The algorithm runs in $O(\log n)$ time, because it is nearly identical to binary search.

4. Now suppose that the arithmetic array has **two** elements missing, neither of which is either the first or the last element. Design an algorithm to efficiently find the missing numbers in the array.

   Solution: The algorithm is similar to our answer to question 2, but instead of two possibilities (before or after the mid element) we have three (both before, one before and one after, both after). Finding c is also a little bit trickier. We treat both $n = 2$ and $n = 3$ as special cases.

```
define locate-both-missing(A, n):
    if n == 2:
        return (2*A[0] + A[1]) / 3, (A[0] + 2*A[1]) / 3

    if n == 3:
        if A[1] - A[0] == A[2] - A[1]:
            return (A[0] + A[1]) / 2, (A[1] + A[2]) / 2
        else if A[1] - A[0] > A[2] - A[1]:
            return (2*A[0] + A[1]) / 3, (A[0] + 2*A[1]) / 3
        else
            return (2*A[1] + A[2]) / 3, (A[1] + 2*A[2]) / 3

    low = 0
    high = n - 1
    c = min(A[1] - A[0], A[2] - A[1], A[3] - A[2])
    while low < high - 1:
        mid = (low + high) // 2
        if A[mid] - A[low] == c * (mid - low): // both on the right
            low = mid
        else if A[high] - A[mid] = c * (high - mid): // both on the left
            high = mid
        else // one on each side
            return locate-missing(A[low...mid]), locate-missing(A[mid...high])
    return A[low] + c, A[low] + 2*c
```

CPSC 320 2021W2: Tutorial Quiz 2, Individual Portion

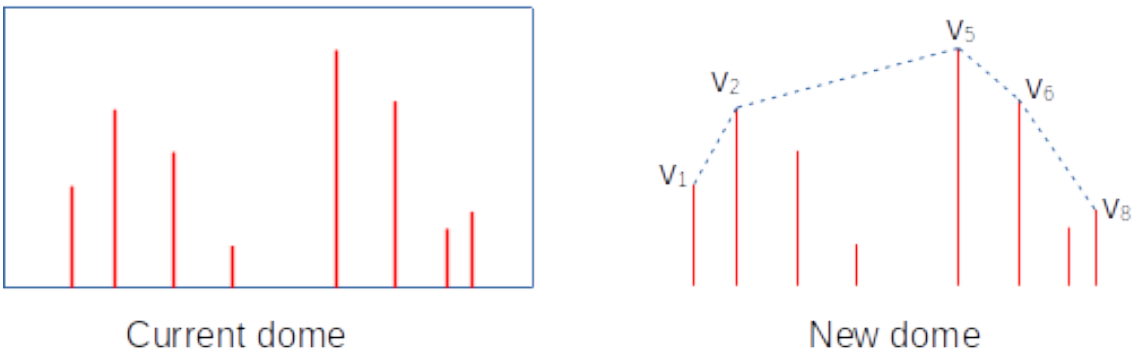
CS ID

Please enter your 4 or 5 digit CSID in this box:

1 Rebuilding the Mars dome efficiently

Fifty years after establishing a colony on Mars, its settlers need to replace the dome that protects their colony from the martian atmosphere and cosmic rays. You have been asked to help them plan the replacement. For simplicity, we will assume that the martian surface is the x-axis, and that each of the n buildings of the colony is a vertical line segment that grows upward from the x-axis, as illustrated in the following figure. So building i is represented by the pair (x_i, y_i) where x_i is its position along the x-axis, and y_i is its height. We will also assume that the buildings are sorted by increasing x-coordinate. That is, $x_1 < x_2 < \cdots < x_n$.

To minimize costs, the colonists are looking for a dome that uses the least material possible. Such a dome is shown in the right half of the figure: observe that it consists of a sequence of edges whose endpoints are the tops of some of the buildings, and that any horizontal line drawn through the figure either does not intersect the dome, or does so in a single line segment. We will denote an optimal dome that covers buildings x_i, x_{i+1}, ..., x_j by $D_{i,j}$. In the example in the figure, $D_{1,8}$ contains the 4 edges $\{(v_1, v_2), (v_2, v_5), (v_5, v_6), (v_6, v_8)\}$.



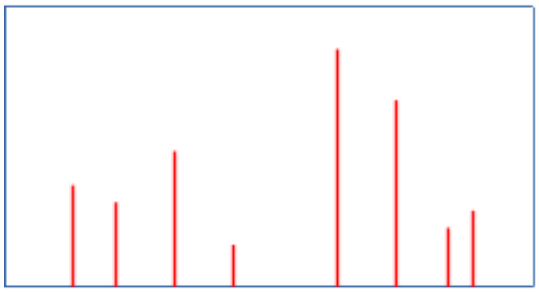
1. Consider the statement

   The point of $D_{1,\lfloor n/2 \rfloor}$ with the largest y-coordinate is also a point of $D_{1,n}$.

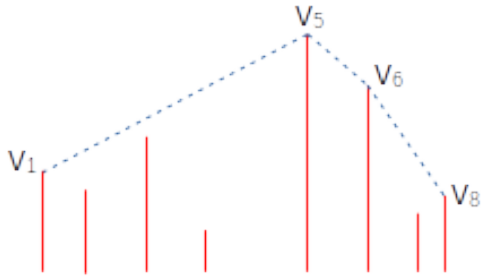
   Determine how frequently this statement is true. If you choose "Always" or "Never", then explain why briefly. If you choose "Sometimes" then draw one example where the statement is true, and one where it is false. Note that we are looking at all possible instances of the problem, not just the example from the figure on the previous page.

   ○ Always ○ Never ● Sometimes

The example on the previous page shows that the point of $D_{1,\lfloor n/2 \rfloor}$ with the largest y-coordinate might also be a point of $D_{1,n}$. The example below shows that it is not always a point of $D_{1,n}$.



Current dome



New dome

2. How many edges of $D_{1,n}$ do not appear in $D_{1,\lfloor n/2 \rfloor}$ and also do not appear in $D_{1+\lfloor n/2 \rfloor,n}$? Once again, we are looking at all possible instances of the problem, not just the example from the figure on the previous page.

   ○ None ● One ○ One or Two ○ Two ○ Some other answer

3. For this question, assume that no three of the points representing the top of the buildings are collinear. That is, assume that no line contains more than two of these points.

   Let l the the infinite line that contains an edge of $D_{1,n}$ that connects a vertex of $v_i = (x_i, y_i)$ of $D_{1,\lfloor n/2 \rfloor}$ to a vertex $v_j = (x_j, y_j)$ of $D_{1+\lfloor n/2 \rfloor,n}$. If v_i has two neighbours on $D_{1,\lfloor n/2 \rfloor}$, then these two neighbours are

   ○ Both above l ● Both below l ○ One above l and the other below l

   ○ More than one of these can happen