

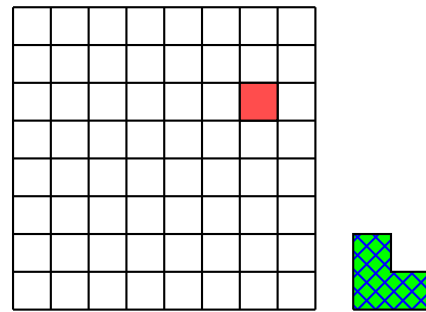
School of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 8

Sample Answers

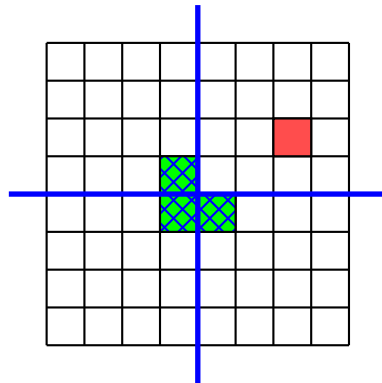
The exercises

50. A *tromino* is an L-shaped tile made up of three 1×1 squares (green/hatched in the diagram below). You are given a $2^n \times 2^n$ chessboard with one missing square (red/grey in the diagram below). The task is to cover the remaining squares with trominos, without any overlap. Design a divide-and-conquer method for this. Express the cost of solving the problem as a recurrence relation and use the Master Theorem to find the order of growth of the cost.

Hint: This is a nice example where it is useful to split the original problem into *four* instances to solve recursively.



Answer: If $n = 0$ then we have a 1×1 board with a missing square, so there is nothing to cover. So let $n > 1$. Breaking the given board into four quarters corresponds to decrementing n by 1.

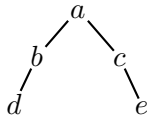


One of the quarters will have the missing square, so place a tromino so that it borders that quarter, straddling the other three. Now we have four sub-problems of the same kind as the original, but each of size $2^{n-1} \times 2^{n-1}$, and we simply solve these recursively.

Let us use m to denote the size of the problem, so $m = 2^{2n}$. The recurrence relation for the cost, in terms of m , is

$$T(m) = 4 T(m/4) + 1 = 4 T(m/4) + m^0$$

with $T(1) = 1$. The Master Theorem tells us that $T(m) = \Theta(n^{\log_4 4}) = \Theta(n)$. That is, our method for solving the puzzle is linear in the size of the board.

51. Traverse  in preorder, inorder, and postorder.

Answer: Preorder: $a-b-d-c-e$. Inorder: $d-b-a-c-e$. Postorder: $d-b-e-c-a$.

52. A certain binary tree yields a, b, c, d, e when traversed preorder, and it yields the same sequence when traversed inorder. What does the binary tree look like?

Answer: It is a stick, with nodes $a \cdots e$, each parent having only a right-child.

53. A certain binary tree yields 14, 83, 63, 42, 19, 27, 74, 99, 51, 37 when traversed preorder and 63, 83, 19, 42, 14, 27, 99, 51, 74, 37 when traversed inorder. Which sequence does it yield when traversed postorder?

Answer: Postorder yields 63, 19, 42, 83, 51, 99, 37, 74, 27, 14. To see this, construct the binary tree from the preorder and inorder sequences (this is best handled as a recursive process).

54. The following algorithm was designed to compute the number of leaves in a binary tree T . We denote the empty tree by $null$.

```

function LEAFCOUNT( $T$ )
  if  $T = null$  then
    return 0
  else
    return LEAFCOUNT( $T.left$ ) + LEAFCOUNT( $T.right$ )

```

Fix the error in this algorithm.

Answer: We need to check for the case of a single-node tree (or sub-tree):

```

function LEAFCOUNT( $T$ )
  if  $T = null$  then
    return 0
  else
    if  $T.left = null$  and  $T.right = null$  then
      return 1
    else
      return LEAFCOUNT( $T.left$ ) + LEAFCOUNT( $T.right$ )

```

55. (Optional, for those who are keen to practice induction proofs.) In lecture 13 (on slide 16) we analyzed the worst-case time complexity of bottom-up heap creation. The analysis made use of this fact:

$$\sum_{i=1}^h i \cdot 2^{h-i} = 2^{h+1} - h - 2$$

Use mathematical induction to prove it.

Answer: It is easy to check that the statement holds for $h = 1$. The induction hypothesis $P(n)$ is that the statement holds for $h = n$, that is,

$$\sum_{i=1}^n i \cdot 2^{n-i} = 2^{n+1} - n - 2 \quad (1)$$

We wish to prove that, for all $n > 0$, $P(n+1)$ holds, assuming $P(n)$ does. That is, we wish to prove

$$\sum_{i=1}^{n+1} i \cdot 2^{n+1-i} = 2^{n+2} - n - 3 \quad (2)$$

Separating out the summand for $i = n+1$ on the left gives us

$$\sum_{i=1}^n i \cdot 2^{n+1-i} + (n+1) \cdot 2^0 = 2 \cdot \sum_{i=1}^n i \cdot 2^{n-i} + n+1$$

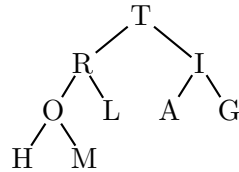
which by (1) is the same as

$$2 \cdot (2^{n+1} - n - 2) + n + 1 = 2^{n+2} - 2n - 4 + n + 1 = 2^{n+2} - n - 3$$

so indeed (2) holds. Since $P(1)$ holds and, moreover, for all $n > 0$, $P(n+1)$ follows from $P(n)$, we conclude that $P(n)$ holds for all positive integers. (It holds for $h = 0$ as well.)

56. Make a max-heap out of the keys A, L, G, O, R, I, T, H, M, using the bottom-up algorithm.

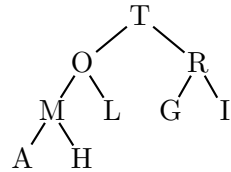
Answer:



Or, as an array: T, R, I, O, L, A, G, H, M.

57. Construct a max-heap from the empty heap by inserting the keys A, L, G, O, R, I, T, H, M, one by one, in that order. Is the result the same as the heap from the previous question?

Answer:



Or, as an array: T, O, R, M, L, G, I, A, H, which is different from the previous question's answer.

58. Give an algorithm for deciding whether an array $A[1]..A[n]$ is a heap.

Answer: Easy:

```

function ISHEAP( $A[\cdot], n$ )
  for  $i \leftarrow 2$  to  $n$  do
    if  $A[i] > A[i/2]$  then
      return False
  return True

```