Chapter 2

—–Material covered in class—–

1. Show the trace of Merge-sort on the following inputs.
   (a) $\{4, 8, 3, 1, 1, 9, 1, 4\}$
   (b) $\{1, 2, 3, 4, 5, 6, 7, 8\}$
2. This question is about Strassen's algorithm. In class, we saw that $C_{11} = P_5 + P_4 - P_2 + P_6$. Verify the other three equalities: $C_{12} = P_1 + P_2$, $C_{21} = P_3 + P_4$ and $C_{22} = P_5 + P_1 - P_3 - P_7$.
3. Once in your life, you should write the trace of Strassen's algorithm. Try the following input...
$$\begin{pmatrix} 1 & 0 & -1 & 2 \\ 3 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 2 & 4 & 7 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 2 & 3 \\ -2 & 1 & -1 & 1 \\ 1 & 0 & 1 & 0 \\ 5 & 1 & -2 & -1 \end{pmatrix}$$
4. Write a complete and detailed example of the $O(n)$ time Selection algorithm. Since we divide the input into groups of 5, it is more convenient if n is a power of 5, but at the same time, we don't want n to be too big...
5. Solve the following recursion equations using the Master Theorem.
   (a)
$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2\,T\left(\frac{1}{5}n\right) + \sqrt{n} & \text{if } n \geq 2. \end{cases}$$
   (b)
$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ n + 2\,T\left(\frac{7}{10}n\right) & \text{if } n \geq 2. \end{cases}$$
   (c)
$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ n + 2017\,T\left(\frac{1}{2017}n\right) & \text{if } n \geq 2. \end{cases}$$
   (d)
$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 8\,T\left(\frac{1}{4}n\right) + n^{3/2} & \text{if } n \geq 2. \end{cases}$$

6. Solve the following recursion equations without the Master Theorem.

   (a)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2\,T\left(\frac{1}{3}n\right) + n & \text{if } n \geq 2. \end{cases}$$

   (b)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 3\,T\left(\frac{1}{2}n\right) + n & \text{if } n \geq 2. \end{cases}$$

   (c)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ \frac{1}{2}\,T(n) + n & \text{if } n \geq 2. \end{cases}$$

   (d)
   $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ \frac{1}{2}\,T\left(\frac{1}{2}n\right) + n & \text{if } n \geq 2. \end{cases}$$

——Design & analysis of algorithms——

7. What is the maximum number of comparisons that binary search must perform before finding (or not) a given item in a list of 10^9 numbers?

8. Show the trace of the binary search algorithm on the following input:

   $$\{110, 155, 315, 401, 415, 810, 1001, 1120, 1315\}$$

   when searching for the number 1120.

9. Merge-sort splits the input into two sub-lists and then it makes two recursive calls, once for each sub-list. What if it was splitting the input into three sub-lists? Write an algorithm that sorts a list of n items by dividing it into three sublists of about $n/3$ items, sorting the sub-lists recursively and then merging them. Analyze the running time of your algorithm and give the result using the O-notation.

10. The binary search algorithm gets a sorted list and a number x as input. It splits this sorted list into two sorted sub-lists of equal size, decide in which of the two sub-lists x is and make a recursive call with this sub-list and x as input. What if it was splitting the input list into three sub-lists? Describe a *ternary search* algorithm which splits the input list into three sub-lists of equal size and then makes one recursive call. Analyze the running time of your algorithm and give the result using the O-notation.

11. Consider the following problem:

   (a) Suppose there are 9 identical-looking coins, all of which have the same weight, except for one, which is heavier than the others. Moreover, suppose you have one balance scale and are allowed only two weighings. Explain how to find the heavier coin.

   (b) Suppose now that there are n identical-looking coins, all having the same weight, except for one heavier coin. Moreover, suppose that n is a power of 3 and that you are allowed $\log_3 n$ weighings to determine the heavier coin. Write an algorithm that solves this problem.

12. Write a divide-and-conquer algorithm to find the smallest item in a list. Analyze your algorithm and give the results using the O-notation.

13. A *tromino* is a group of three unit squares arranged in an L-shape. Consider the following tiling problem (refer to Figure 1).

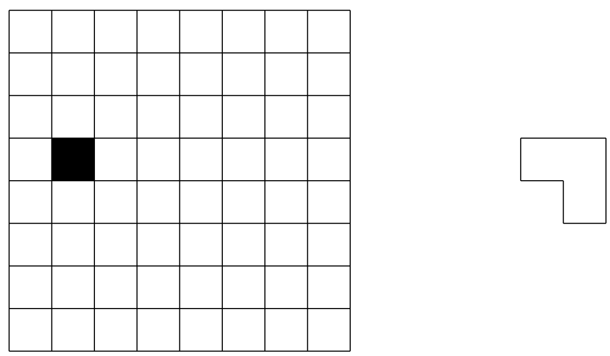

Figure 1: Illustration of Question 13.

The input is an $n \times n$ array of unit squares where n is a power of two and exactly one square is missing. The output is a tiling of the array that satisfies the following:

- Every non-missing square is covered by a tromino.
- No tromino covers the missing square.
- No two trominos overlap.
- No tromino extends outside the array.

Write a divide-and-conquer algorithm that solves this problem.

14. Write an $O(n \log n)$ time algorithm that partitions a list of $2n$ distinct positive integers into two sub-lists of size n such that the difference between the sums of the integers in each sub-list is maximized.

15. Write an efficient algorithm that searches for a value in an $m \times n$ matrix whose rows are sorted from left to right and whose columns are sorted from top to bottom. Analyze the running time of your algorithm and give the result using the O-notation.

16. You are given two sorted arrays $A[1..n]$ and $B[1..(n+1)]$. Except for one number, A and B are identical. Find the index of this extra element in $O(\log(n))$ time. For instance, if the input is

$$A = \{21, 32, 43, 54, 66, 77, 88\} \qquad\qquad B = \{21, 32, 43, 54, 55, 66, 77, 88\},$$

then the output should be 5 since $B[5] = 55 \notin A$.

17. You are given two sorted arrays $A[1..n]$ and $B[1..n]$. Find the median of the array you would get by merging A and B. Your algorithm must take $O(\log(n))$ time.

18. * Given n rectangular buildings in a 2-dimensional city, compute the skyline of these buildings, by eliminating hidden lines. The main task is to view buildings from a side and remove all sections that are not visible. All buildings share common bottom and every *building* is represented by two ordered pairs $\{(\text{left}, \text{ht}), (\text{right}, 0)\}$, where

    - left: is x coordinated of left side (or wall).
    - ht: is height of building.
    - right: is x coordinate of right side

    For instance, the building on the left of Figure 2 is represented as $\{(2, 7), (5, 0)\}$. A

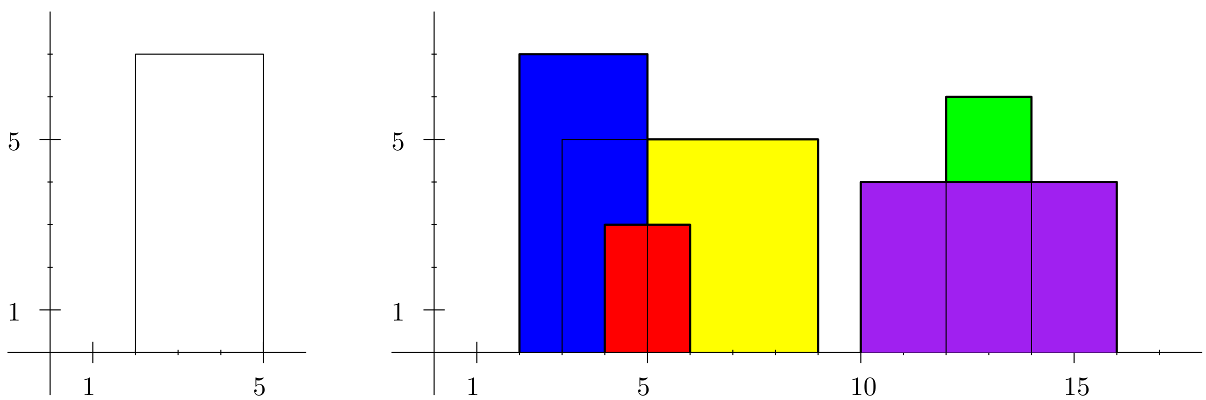

Figure 2: Illustration of Question 18. This city is made of 5 buildings. The corresponding skyline is $\{(2, 7), (5, 5), (9, 0), (10, 4), (12, 6), (14, 4), (16, 0)\}$.

skyline is a collection of rectangular strips. A rectangular strip is represented as a pair (left, ht) where left is x coordinate of left side of strip and ht is height of strip (refer to Figure 2).

Your algorithm must take $O(n \log(n))$ time.

19. Suppose there are $n = 2^k$ teams in an elimination tournament, in which there are $n/2$ games in the first round, with the $n/2 = 2^{k-1}$ winners playing in the second round and so on.

   (a) Develop a recurrence relation for the number of rounds in the tournament.

   (b) Solve the recurrence relation of part (a).

   (c) How many rounds are there in the tournament when there are 64 teams?

——Proofs——

20. Suppose that, in a divide-and-conquer algorithm, we always divide an instance of size n of a problem into 10 sub-instances of size $n/3$ and the dividing and combining steps take $\Theta(n^2)$ time. Write a recurrence relation for the running time of this algorithm and solve it.

21. Let $T(n)$ be the running time of Merge-sort on an input of size n. In class we saw that $T(n)$ satisfies

$$T(n) \leq \begin{cases} c & \text{if } n = 1, \\ c\,n + 2\,T\left(\frac{1}{2}n\right) & \text{if } n \geq 2. \end{cases},$$

for some constant $c > 0$. Then, we proved that if $c = 1$ and n is a power of two, then $T(n) = O(n\log(n))$.

   (a) Explain why for a general n,

$$T(n) \leq \begin{cases} c & \text{if } n = 1, \\ c\,n + T\left(\lfloor\frac{1}{2}n\rfloor\right) + T\left(\lceil\frac{1}{2}n\rceil\right) & \text{if } n \geq 2. \end{cases}$$

   (b) Prove by induction that in (a), $T(n)$ satisfies $T(n) \leq c\,n + 3c\,n\log_2(n)$.

   (c) Conclude that in (a), $T(n)$ satisfies $T(n) = O(n\log(n))$.

22. The Selection algorithm we studied in class takes $O(n)$ time. To prove that, we showed by induction that the recursion equation $T(n) \leq n + T\left(\frac{1}{5}n\right) + T\left(\frac{7}{10}n\right)$ solves to $T(n) = O(n)$. What if instead of a proof by induction, we wrote

$$T(n) \leq n + T\left(\frac{1}{5}n\right) + T\left(\frac{7}{10}n\right) \leq n + 2\,T\left(\frac{7}{10}n\right)$$

and then apply the Master theorem?

23. Consider a recursion equation of the form

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ a\,T\left(\frac{1}{b}n\right) + n^d & \text{if } n \geq 2. \end{cases}$$

(a) If $a = b = 2$, what are the values of d for which $T(n) = O(n)$? Prove that your answer is correct.

(b) If $a = b = 2$, what are the values of d for which $T(n) = O(n \log(n))$? Prove that your answer is correct.

(c) If $a = b = 2$, what are the values of d for which $T(n) = O(n^2)$? Prove that your answer is correct.