Algorithms Example Sheet 1: Problems

Exercise 1.P.1 [Asymptotics] For each of the following "=" lines, identify the constants k, k_1, k_2, N as appropriate. For each of the "$\neq$" lines, show they can't possibly exist.

(a) $|\sin(n)| = \mathcal{O}(1)$,

(b) $200 + \sin(n) = \Theta(1)$,

(c) $123456n + 654321 = \Theta(n)$,

(d) $2n - 7 = \mathcal{O}\left(17n^2\right)$,

(e) $\log(n) = \mathcal{O}(n)$,

(f) $\log(n) \neq \Theta(n)$,

(g) $n^{100} = \mathcal{O}(2^n)$,

(h) $1 + 100/n = \Theta(1)$,

(i) (+++) $|\sin(n)| \neq \Theta(1)$.

Exercise 1.P.2 [More Asymptotics]

(a) Show that for $a > b > 0$, $n^b \in \mathcal{O}(n^a)$.

(b) Show that for $a > b > 0$, $b^n \in \mathcal{O}(a^n)$.

(c) Compare $n \cdot 2^n$ and 3^n.

(d) Compare n and $(\log n)^{10}$.

(e) Compare n and $\exp((\log n)^{1/2})$.

(f) Compare n^3 and $\exp((\log n)^{9/2})$.

(g) Compare $f(n) = \sum_{i=1}^{n} i^5$ and 2^n. (*Hint:* Use the Discrete Maths exercise for the sum of k-th powers)

(h) Sort the functions in increasing order of asymptotic complexity: $f_1(n) = n^{0.999} \log n$, $f_2(n) = 10^9 \cdot n$, $f_3(n) = 1.00001^n$ and $f_4(n) = n^{1.01}$.

(i) Sort the functions in increasing order of asymptotic complexity: $f_1(n) = 2^{2^{5000}}$, $f_2(n) = 2^{1000n}$, $f_3(n) = \binom{n}{2}$ and $f_4(n) = n\sqrt{n}$.

(j) Sort the functions in increasing order of asymptotic complexity: $f_1(n) = n^{\sqrt{n}}$, $f_2(n) = 2^n$, $f_3(n) = n^{10} \cdot 2^{n/2}$ and $f_4(n) = \sum_{i=1}^{n} i$.

(k) (+) Attempt **[2016P1Q8 (a)]**.

(l) (optional ++) Problems 3.2, 3.3, 3.4, 3.6 from CLRS.

Exercise 1.P.3 [Matrix exponentiation (++)] (Only attempt this if you know about matrices). Consider two 2×2 matrices A and B.

(a) Implement an OCaml function that takes the elements of A and B and returns the matrix product of these two.

$$A \cdot B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

(b) Modify the `power` function (you learnt in FoCS) to compute the power of a matrix.

(c) What is the time complexity of your algorithm?

(d) Implement an OCaml function that takes a 2×2 matrix A and a 2-element vector v and computes $A \cdot v$.

(e) The Fibonacci sequence is defined as $F_n = F_{n-1} + F_{n-2}$ (for $n > 1$) with $F_0 = 0$ and $F_1 = 1$. Show that for $n > 0$

$$\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix}$$

(f) Using the functions you developed above, show how to compute the n-th Fibonacci number in $\mathcal{O}(\log n)$ time.

Exercise 1.P.4 [Max/Min]

(a) Design an algorithm for finding the maximum in an array of n elements.

(b) What is the time complexity of your algorithm? Prove a corresponding lower bound.

(c) Repeat the first two parts for the minimum.

(d) Given an array of n elements, find the pair of elements that has the largest difference. You can submit your solution on **[SPOJ EIUMADIS]**.

(e) (+) Think about how you would solve **[SPOJ DIFERENC]** in $\mathcal{O}\left(N^2\right)$ time (no need to implement it).

Exercise 1.P.5 [Second max]

(a) Describe an algorithm for finding the second largest element in an array of n elements.

(b) What is the time complexity of your algorithm? Prove an asymptotic lower bound (i.e. we do not care about multiplicative constants).

(c) (optional) See **[2007P10Q10 (d)]**.

Binary search

Exercise 1.P.6 [Binary Search on the answer] Read the statement of **[SPOJ AGGRCOW]**

(a) Try to solve this problem first: Given that all stalls are separated by a distance of at least d, determine if it is possible to place the cows.

(b) Let $f(d)$ (with $f : \mathbb{N} \to \{0, 1\}$) be the answer to the above problem. Argue that this function is monotonic.

(c) Use binary search to solve the original problem.

(d) (optional) Implement the solution.

Exercise 1.P.7 [Binary Search on the answer] Try to solve the problem **[SPOJ BOOKS1]** using the same technique as in **[SPOJ AGGRCOW]**.

Exercise 1.P.8 [Binary Search on rotated array] Try the following common interview question: **[LeetCode 81]**.

Exercise 1.P.9 [Removing duplicates] Design an algorithm to remove all duplicate elements from an array. For example, given $[1; 2; 6; 2; 1; 3; 2]$, it should return $[1; 6; 3; 2]$ (in any order). What is the worst-case time complexity of your algorithm?

Exercise 1.P.10 [Intersection of two arrays] Describe an algorithm to compute the intersection of two arrays. For example, given $[1; 2; 6; 2; 1; 3; 2]$ and $[6; 1; 7; 2; 2; 4;]$, it should return $[1; 6; 2; 2;]$ (in any order). What is the worst-case time complexity of your algorithm?
You can test your implementation on **[LeetCode 350]**.

Exercise 1.P.11 [Union of two arrays] Design an algorithm to compute the union of two arrays. For example, given $[1; 2; 6; 2; 1; 3; 2]$ and $[6; 1; 7; 2; 2; 4;]$, it should return $[1; 6; 7; 2; 4; 3]$ (in any order). What is the worst-case time complexity of your algorithm?
You can test your implementation on **[GeeksForGeeks Union of Arrays]**.

Exercise 1.P.12 (optional) Attempt **[2010P1Q5 (c)]**.

Exercise 1.P.13 (optional) Attempt **[2018P1Q7 (a)]**.

Exercise 1.P.14 [Most frequent elements] Describe an algorithm to find the most frequent elements in an array. For example, given $[1; 2; 1; 1; 3; 3; 2; 4; 3]$, it returns $[1; 3]$ since they both occur 3 times.

Exercise 1.P.15 (optional) Attempt **[2011P1Q5]**.

Chapter 2.11/2.12

Exercise 1.P.16 Attempt **[2006P1Q4 (c)]**.

Exercise 1.P.17 [Finding top k elements]

(a) You are given an array containing pairs of `(friendID, time spent last week)`. How you would find the IDs of your friends with whom you spent most time together during the last week? What is the time complexity of your algorithm?

(b) (optional) See **[2007P10Q10 (b)]**.

Chapter 2.14

Exercise 1.P.18 [All items] You are given n boxes each one coloured with one of the M available colours. Describe an algorithm that checks if there is at least one box from each possible color.

Exercise 1.P.19 [Union/Intersection using counting sort] Modify your solution for finding the union and intersection of two arrays to use counting sort. What is the time complexity of this approach?

Exercise 1.P.20 [Does pair exist with given sum] Given an array A of integers and an integer K, determine if there are two elements $A[i]$ and $A[j]$ such that $A[i] + A[j] = k$. What is the time complexity of your algorithm?

Exercise 1.P.21 [Median using counting sort] How can you use counting sort to find the median of an array?

Exercise 1.P.22 [Bucket Sort] (optional) Attempt **[2018P1Q7 (b)]**.

Problems using sorting techniques

Exercise 1.P.23 [Union of intervals] You are given n intervals $[a_i, b_i]$ (with $a_i \neq b_i \in \mathbb{N}$). The area covered by the intervals is that of covered by the union of intervals. For example, the intervals

$$[[1, 5]; [2, 6]; [4, 9]; [14, 16]]$$

cover a total area of 12.

(a) Describe an algorithm for finding the area covered by the intervals. What is the time complexity of your algorithm?

(b) (optional) How would you modify your solution if $a_i, b_i \in \mathbb{Z}$?

(c) (optional) What if $a_i, b_i \in \mathbb{R}$?

If you want you can submit an implementation to **[LeetCode 56]**

Exercise 1.P.24 [Interval with all types of elements] There are n cows standing on a line at known positions x_i for the i-th cow. The type of i-th cow is t_i. There are B breads of cows. You want to take a photograph of the cows. The photograph covers M unit steps of the line. Describe an algorithm to check if there is an interval of length M (where M is given) that contains all B types of cows.

Exercise 1.P.25 [Smallest interval with all types of elements] Extending the setting of Exercise 24, but this time we want to search for the smallest interval length M such that you can capture all B types. **[Usaco Cow Lineup]**