Algorithms and Data Structures Course
English Exercises

Algorithms and Complexity Group, TU Wien
algodat@ac.tuwien.ac.at

Oktober 2021

Exercise 1 Four students (A, B, C, D) participate in a student exchange. Each of them should be matched to one of the host families (W, X, Y, Z). Each student has a preference ordering on the families and each family has a preference ordering on the students. Find a stable matching using the Gale-Shapley algorithm. If there are several options, choose students/families in alphabetical order. Give all intermediate steps, represent each step of the algorithm by stating all fixed pairs.

	1.	2.	3.	4.
W	B	A	D	C
X	B	C	D	A
Y	A	B	D	C
Z	D	C	B	A

	1.	2.	3.	4.
A	Z	Y	X	W
B	X	Y	W	Z
C	Z	X	Y	W
D	Z	W	Y	X

Exercise 2 Consider functions $f(n)$ and $g(n)$ such that $f(n)$ is in $O(g(n))$ and $g(n)$ is in $O(f(n))$. Prove or disprove (via a counter-example) the following statements:

- $f(n) = \Theta(g(n))$
- $f(n) = g(n)$

Exercise 3 Consider the following functions:

$$f(n) = 2^n(100n^2 + 20n + 7)$$

$$g_1(n) = \begin{cases} 100n^2 + 2^n \cdot n^2 + 7n + (\frac{3}{2})^n & \text{if } (n < 10^3) \text{ or } (n \geq 10^5) \\ 5^n + 2^{\frac{3n}{2}} + 100n & \text{else} \end{cases}$$

$$g_2(n) = \frac{2^{2^n}}{10000}$$

$$g_3(n) = \begin{cases} 60 & \text{if } n \text{ is a prime} \\ n^{n^n} & \text{else} \end{cases}$$

Check all boxes that apply in the following table and explain your answers.

$f(n)$ is in	$\Theta(.)$	$O(.)$	$\Omega(.)$	none
$g_1(n)$				
$g_2(n)$				
$g_3(n)$				

Hint: Replace the dot with the corresponding function (g_1, g_2 resp. g_3). For example, the first box is supposed to be read as "$f(n)$ is in $\Theta(g_1(n))$".

Exercise 4 Order the following functions with respect to their asymptotic growth, starting with the function with the slowest asymptotic growth. Additionally mark functions that have the same asymptotic growth rate.

$$\log\left(n^{12}\right), \quad \left(\frac{9}{8}\right)^n, \quad 2n^{10} - 8n^4, \quad \left(\frac{5}{6}\right)^n, \quad \sqrt{n^{16}}, \quad n^2 (\log n)^2, \quad n^5 \cdot 5 \cdot n^5$$

(It suffices to give the correct order. No formal proof is required).

Exercise 5 Answer the following questions and explain the reasoning behind your answers:

- Consider an algorithm whose worst case running time is in $\Theta(n^2)$. Is it possible that its running time on **all** inputs is in $O(n)$?
- Consider an algorithm whose worst case running time is in $\Theta(n^2)$. Is it possible that its running time on **some** inputs is in $O(n)$?

Exercise 6 Prove: $f(n) + g(n) = O\left(\max(f(n), g(n))\right)$.

Exercise 7 Determine for the following pseudo code:

- the **running time** in terms of n in Θ notation and
- the **values** of the variables a and b after the execution of the code in terms of n in Θ notation.

$a \leftarrow 1$
$b \leftarrow 1$
for $c \leftarrow 1, \ldots, \lfloor\sqrt{n}\rfloor$
    $a \leftarrow a + a$
    $b \leftarrow b + c$
$i \leftarrow 2^{2b}$
while $i > 1$
    $i \leftarrow \lfloor \frac{i}{2} \rfloor$

Exercise 8 Determine the running time of the following pseudo code in terms of n in Θ notation.

$i \leftarrow n$
$j \leftarrow n$
$k \leftarrow 0$
while $i > 0$
    $j \leftarrow j \cdot n$
    $i \leftarrow i - 1$
while $j > 0$
    if $j = \lfloor \frac{j}{2} \rfloor \cdot 2$ **then**
        $k \leftarrow k + 1$
    $j \leftarrow \lfloor \frac{j}{2} \rfloor$

Exercise 9 Determine the running time of the following pseudo code in terms of n in Θ notation.

```
max ← 0
ind ← 0
for i ← 1, ..., n
    A[i] ← n
if n = 1 then
    while A[1] > 0
        A[1] ← A[1] − 1
else
    while A[n] > 0
        max ← 0
        for j ← 1, ..., n
            if A[j] > max then
                max ← A[j]
                ind ← j
                if A[1] > A[2] then
                    for k ← 1, ..., j
                        max ← max + 1
        A[ind] ← A[ind] − 1
```

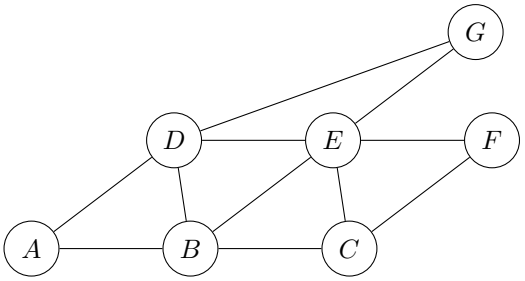
Exercise 10 Let $G = (V, E)$ be an undirected graph. We want to determine if G contains a cycle of length 3.

- First give a simple algorithm with running time in $O(n^3)$ for solving this problem. It suffices to sketch the algorithm and the argumentation for its correctness and its running time.
- Now find a better algorithm that solves the problem in running time $O(n \cdot m)$. You can assume that $n \leq m$ holds. Give the algorithm in detailed pseudo code and argue that it is correct and that it has the expected running time.

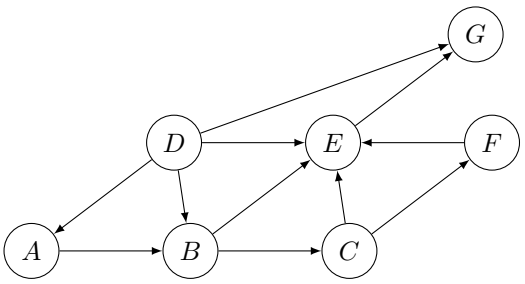
  Hint: It is possible to switch between the representation of the graph by an adjacency matrix and by an adjacency list in $O(n \cdot m)$.

Exercise 11

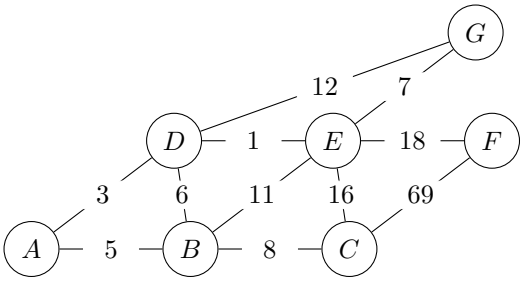
(a) Perform on the following graph a breadth-first search and a depth-first search. Give the order in which the vertices are discovered. Use A as your starting point.



(b) Determine a topological ordering on the following graph. (It suffices to give an ordered list of vertices.)



(c) Determine a minimal spanning tree of the following graph using Prim's algorithm or Kruskal's algorithm. (It suffices to give an ordered list of vertices.)



Exercise 12 Construct an undirected graph with at least five vertices and at least three vertices of degree three such that breadth-first and depth-first search discover its vertices in the same order.

Exercise 13 Let G be a undirected graph. Give an algorithm that computes in linear time the size of the largest connected component of G. (The size of a connected component is the number of vertices in the connected component.) Give the algorithm in detailed pseudo code and argue that it is correct and that it has the expected running time.

Exercise 14 Let $G = (V, E)$ be an undirected graph and let F be a set with k elements called colors. Then we call a mapping $f : V \to F$ a k-coloring of G if $(u, v) \in E$ implies $f(u) \neq f(v)$, i.e. if no two adjacent vertices are assigned the same color. We say a undirected graph G is k-colorable if there exists a k-coloring of G.

Find a linearer time algorithm that decides if a given graph is 2-colorable. Give the algorithm in detailed pseudo code and argue that it is correct and that it has the expected running time.

Exercise 15 We consider a computer science program at a university that consists of n mandatory courses. Some of these courses are dependent on other courses and can not be completed in the same semester as the courses they depend on. These dependences are given as a directed graph $G = (V, E)$. Each course is represented by a vertex and an edge $(u, v) \in E$ means that course v is dependent on course u.

Develop an algorithm that finds in linear time the minimal number of semesters needed to finish the program. You can assume that one can complete arbitrary many courses in one semester. Give the algorithm in detailed pseudo code and argue that it is correct and that it has the expected running time.

Exercise 16 Construct a weighted graphs that has exactly one minimal spanning tree T and that contains two vertices a and b such that:

- the only shortest path P from a to b does not contain any edge that is also contained in T and
- P contains at least two edges.

Exercise 17 Consider an array A of size n with red, green and blue elements. Find an algorithm that rearranges the elements of A in linear time such that the new array contains first all red, then all green and finally all blue elements. The algorithm can use only the following operations on the array:

- **Examine(i)** – gives back the color of the i-th element in the array.
- **Swap(i, j)** – swaps the i-th element with the j-th element in the array.

Give the algorithm in detailed pseudo code and argue that it is correct and that it has the expected running time.

Exercise 18 You wish to store a set of n numbers in either a min-heap or a sorted array. For each application below, state which data structure is better, or if it does not matter. Explain your answers.

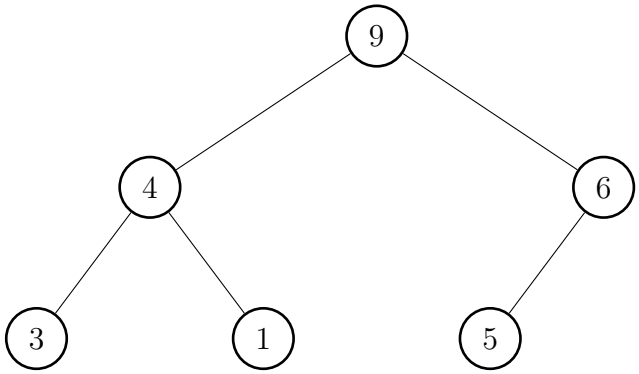
(a) Want to find the minimum element quickly.

(b) Want to be able to delete an element quickly.

(c) Want to be able to form the structure quickly.

(d) Want to find the maximum element quickly.

Exercise 19 Consider the following max-heap:



Hint: A max-heap is similar to the (min)-heap defined in the book, but the maximal element is the root and the values of children are always smaller or equal than their parents values. The heap operations have to be adapted accordingly.

(a) Add the element 7. Give all intermediate steps and the final heap.

(b) From the final heap of exercise (a), delete the element 9 and then the element 1. Give all intermediate steps and the final heap.

Exercise 20 Consider n programs $P_1, \ldots, P_n$ which should be stored on a hard-drive that has a capacity of D megabyte. Program P_i needs s_i megabyte space. Assume that it is not possible to store all programs at once ($D < \sum_{i=1}^{n} s_i$).

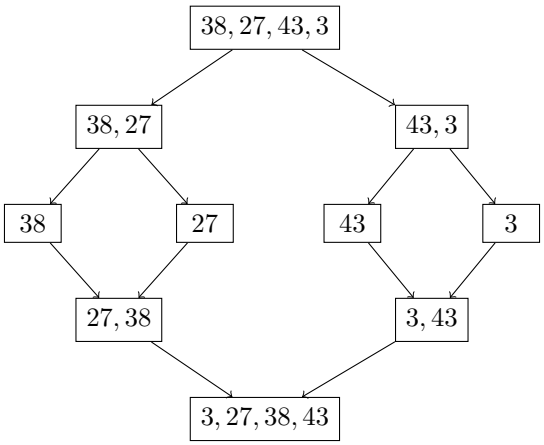
(a) Does a greedy algorithm that selects programs in order of nondecreasing size maximize the number of programs held on the disk? Prove or give a counterexample.

(b) Does a greedy algorithm that selects programs in order of nonincreasing size use as much of the capacity of the disk as possible? Prove or give a counterexample.

Exercise 21 Apply Mergesort to the following array:

$$[51, 13, 28, 4, 60, 9].$$

Draw a "recursion tree" similar to this one:



Exercise 22

(a) Let $\mathcal{P}$ be a decision problem and let n be the size of its input. Assume that there is a reduction from $\mathcal{P}$ to a problem $\mathcal{Q}$ with running time $O(n^3 \cdot \log n)$. Furthermore, assume that Q can be solved in $O(m^2)$, where m is the size of the input of $\mathcal{Q}$.

- Can every instance of $\mathcal{P}$ be solved in polynomial time?
- Give an upper bound on the running time of an (optimal) algorithm for $\mathcal{P}$ and explain your answer.

(b) Let $\mathcal{P}^*$ be a decision problem. Assume that there is linear time reduction from $\mathcal{P}^*$ to a problem $\mathcal{Q}^*$. Furthermore, assume that $\mathcal{Q}^*$ is NP-complete. Which of the following statements are true? Explain your answers.

- $\mathcal{P}^*$ is in NP.
- $\mathcal{P}^*$ is NP-complete.
- If SAT can be solved in polynomial time then $\mathcal{P}^*$ can be solved in polynomial time.
- If SAT can be solved in linear time then $\mathcal{P}^*$ can be solved in linear time.

Exercise 23 Are the following problems in NP? If yes, describe how a certificate could look like.

- Given a weighted graph G, a natural number k and vertices u and v, is there a path of length k or smaller from u to v?
- Given a position in chess, is there a winning strategy for white?
- Given two natural numbers m and n, is there a natural number $k \neq 1$ smaller than m such that $\frac{n}{k}$ is a natural number.
- Given a graph G and a natural number k, is there a minimal spanning tree T of G such that every vertex in T has degree smaller k.

Exercise 24 Recall the reduction from Vertex Cover to Set Cover from the book. Describe and solve the instances of Set Cover which results when the reduction is applied on the following instances of Vertex Cover:

a) A cycle C_4 with four vertices $\{a, b, c, d\}$, an additional vertex x and edges between x and all other vertices, $k = 2$.

b) A complete bipartite graph $K_{3,2}$ with three vertices $\{a, b.c\}$ on the one side and two vertices $\{x, y\}$ on the other side, $k = 2$.

Hint: A bipartite graph is complete if every vertex on one side is connected with every vertex on the other side.

Exercise 25 Recall the reduction from 3-SAT to Independent Set from the book. Draw the graph G which results from applying the reduction on the following 3-SAT instance ϕ.

$$\phi = (x_3 \vee \overline{x_4} \vee x_1) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_1} \vee x_4) \wedge (x_4 \vee \overline{x_1} \vee x_3)$$

Exercise 26 Assume that you have an algorithm $\mathcal{A}$ that can solve Vertex Cover on graphs with a certain *monotone property* P in time $O(n^{5/2})$. Use $\mathcal{A}$ to design and describe a polynomial-time optimization algorithm $\mathcal{B}$ which takes as input a graph H with property P and finds a minimal vertex cover of H. What is the runtime of $\mathcal{B}$ and why is it important that P is a monotone property?

Hint: A graph property P is called *monotone*, if the fact that a graph G has property P implies that every subgraph of G has property P. For example, if G is a bipartite graph, then every subgraph of G is also bipartite.

Exercise 27 Solve the following instance of the Weighted Interval Scheduling Problem using the dynamic programming algorithm from the book. Give the array M and, for every j the values of $p(j)$, $v_j + M[p(j)]$ and $M[j-1]$. Finally, compute the optimal solution using M-Compute-Opt(n).

Request	Start time	Finish time	Value
1	4	8	1
2	2	7	5
3	9	11	1
4	1	3	2
5	2	17	7
6	8	10	3
7	3	4	1
8	3	5	2
9	13	16	9
10	10	14	10
11	7	12	11

Exercise 28 Let $G = (V, E)$ be a graph. We call a set of vertices $D \subseteq V$ a dominating set, if every vertex in $V \setminus D$ is adjacent to at least one vertex in D. The Dominating Set problem is defined as follows:

Given a graph $G = (V, E)$ and a natural number k, is there a dominating set D on G with $|D| \leq k$?

Give a polynomial time reduction from Dominating Set on Set Cover and argue that your reduction is correct.

Exercise 29 The problem Minimum Dominating Set is defined as follows:

Given a graph $G = (V, E)$, find the smallest dominating set D on G.

Minimum Dominating Set is a NP-hard optimization problem. Develop a local search algorithm for Minimum Dominating Set. Describe a suitable neighbor relation and, using this relation, a local search algorithm. Furthermore, give an example where your algorithm gives a solution that is at least twice as big as the optimal solution.

Exercise 30 Recall from the book the Load Balancing Problem and the simple Greedy-Balance algorithm for solving it. It is shown in the book that this algorithm is a 2-approximation algorithm.

Now assume that we consider only instances with the following properties:

- There are at least 100 Jobs.
- There are at most 5 machines.
- Each job j has processing time $1 \leq t_j \leq 10$.

Does Greedy-Balance perform better on these instances than on arbitrary instances? In other words, has Greedy-Balance and approximation factor < 2 on these instance? If yes, prove an approximation factor < 2, if no give a counter example where Greedy-Balance gives a result that is twice as big as the optimal result.
