**Analysis of Algorithms WS 2022**
Prof. Dr. P. Rossmanith
M. Gehnen, H. Lotze, D. Mock

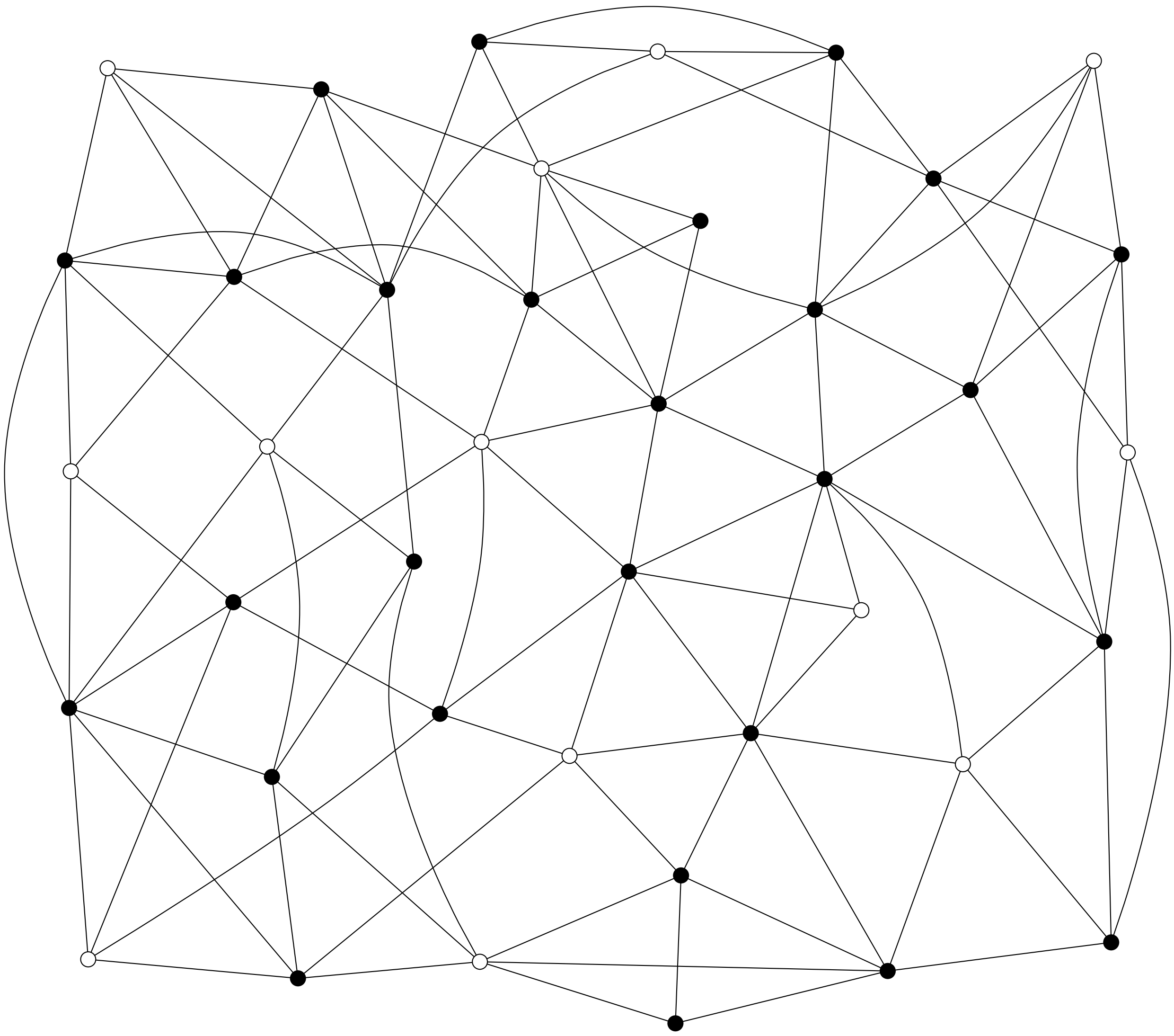
Date: November 21, 2022

# Exercise Sheet with solutions 04

Due date: next tutorial session

### Tutorial Exercise T4.1

A *vertex cover* of an undirected graph $G = (V, E)$ is a subset $C \subseteq V$ of its vertices such that at least one endpoint of every edge is in $C$, i.e., for every $(v_i, v_j) \in E$, either $v_i \in C$ or $v_j \in C$. Informally speaking, "$C$ covers all edges." It is an NP-complete problem to find out whether a graph has a vertex cover of a given size. The following example shows a graph of order 40. The black vertices comprise a minimum size vertex cover.



Now, given a graph $G = (V, E)$ we define its size as the number of vertices $|V|$ of the graph. So, let us consider a graph of size $n$ and let $k$ be targeted vertex cover size. Find an algorithm for Vertex Cover that runs in time $1.5^k n^{O(1)}$. Hints: If a graph has maximum degree two, i.e., without any vertex of degree three or more, this problem can be solved in polynomial time. On the other hand, if a vertex $v$ is not in the vertex cover, all of its neighbors have to be there.

**Solution**

Let us denote by $a_k$ the running time of our algorithm if the targeted size of the vertex cover is $k$. Given a graph $G = (V, E)$ of size $n$, first we check that there is a vertex of degree higher than two. As the hint states, if a given graph has degree two, one can solve it in polynomial time. Otherwise, let us take a vertex $v$ of degree 3 or more, and let $N(v) \subseteq V$ be its neighborhood. We do the following branching procedure.

If we take the vertex $v$ into the vertex cover, we only need to find a vertex cover of size $k-1$ on the graph without $v$ or its neighbors, i.e., $G \setminus \{v, N(v)\}$. This will have a running time of $a_{k-1}$. Otherwise, if $v$ is not the vertex cover, then, as the hint says, $N(v)$ needs to be in the vertex cover, and $N(v)$ contains at least three vertices. So, one will ned to find a vertex cover of size $k-3$ on the same graph $G \setminus \{v, N(v)\}$ as before. This will have a running time of at most $a_{k-3}$. This gives us the recurrence relation $a_k = a_{k-1} + a_{k-3}$ for the dependence in $k$. Additionally one can take into account the dependence in $n$, which is linear in each iteration, so we know that in total it will be a polynomial term multiplying the term with the dependency in $k$. So, in order to know the running time of this algorithm, we only need to solve the recurrence relation we just found by using the usual methods.

The recurrence relation $a_k = a_{k-1} + a_{k-3}$ is linear and has constant coefficients, its characteristic polynomial is $\chi(z) = z^3 - z^2 - 1$. The roots of this polynomial can be found using the closed formula for cubic polynomials or a solver and they are

$$\begin{aligned}
\alpha_1 &= \frac{1}{3}\left(1 + \sqrt[3]{\frac{1}{2}(29 - 3\sqrt{93})} + \sqrt[3]{\frac{1}{2}(29 + 3\sqrt{93})}\right), \\
\alpha_2 &= \frac{1}{3} - \frac{1}{6}(1 - i\sqrt{3})\sqrt[3]{\frac{1}{2}(29 - 3\sqrt{93})} - \frac{1}{6}(1 + i\sqrt{3})\sqrt[3]{\frac{1}{2}(29 + 3\sqrt{93})}, \\
\alpha_3 &= \frac{1}{3} - \frac{1}{6}(1 + i\sqrt{3})\sqrt[3]{\frac{1}{2}(29 - 3\sqrt{93})} - \frac{1}{6}(1 - i\sqrt{3})\sqrt[3]{\frac{1}{2}(29 + 3\sqrt{93})}.
\end{aligned}$$

If we want an exact formula we can take the initial terms $a_0$, $a_1$, and $a_2$ and find the coefficients of the generic solution $a_k = \mu_1(\alpha_1)^k + \mu_2(\alpha_2)^k + \mu_3(\alpha_3)^k$. This is complicated, however, because the roots of this polynomial are not simple. Moreover, we might not need an exact solution; if we look at the approximate values we see that $\alpha_1 \approx 1.46$, $\alpha_2 \approx -0.2 - 0.8i$, and $\alpha_3 \approx -0.2 + 0.8i$. Because $\alpha_2$ and $\alpha_3$ are much smaller than $\alpha_1$ in terms of absolute value, the first root dominates and we can say that $a_k = O(1.5^k)$, which means that the given algorithm for vertex cover achieves the targeted running time.

**Tutorial Exercise T4.2**

Solve the following recurrence: Let $a_0 = 1$, $a_1 = 1$, $a_2 = 4$ and

$$a_n = 2a_{n-1} - a_{n-2} + 2a_{n-3}, \text{ for } n \geq 3.$$

**Solution**

The characteristic polynomial is $z^3 - 2z^2 + z - 2$. This can be factorized as $(z^2 + 1)(z - 2)$, which gives the set of roots as $\{\pm i, 2\}$. The general solution is thus:

$$a_n = \alpha \cdot 2^n + \beta \cdot (-i)^n + \gamma \cdot i^n.$$

Substituting the values for $n = 0, 1, 2$, we obtain the following system of linear equations:

$$\alpha + \beta + \gamma = 1 \tag{1}$$
$$2\alpha - i\beta + i\gamma = 1 \tag{2}$$
$$4\alpha - \beta - \gamma = 4 \tag{3}$$

Solving this system yields $\alpha = 1$, $\beta = -i/2$ and $\gamma = i/2$. Note that $\beta$ and $\gamma$ are complex conjugates. The solution is thus:

$$a_n = 2^n - \frac{i}{2} \cdot (-i)^n + \frac{i}{2} i^n \tag{4}$$
$$= 2^n + (1 + (-1)^{n+1}) \cdot \frac{i^{n+1}}{2}. \tag{5}$$

This solution can be written in a much nicer form using Euler's formula:

$$e^{i\theta} = \cos\theta + i \sin\theta.$$

We may write $i^{n+1}/2$ as follows:

$$\frac{i^{n+1}}{2} = \frac{1}{2}\left(\cos\frac{\pi}{2} + i\sin\frac{\pi}{2}\right)^{n+1} = \frac{e^{\frac{i\pi}{2}\cdot(n+1)}}{2}.$$

Similarly, $(-1)^{n+1} i^{n+1}/2$ may be written as $\frac{1}{2} \cdot e^{-\frac{i\pi}{2}\cdot(n+1)}$. Now note that

$$\frac{e^{\frac{i\pi}{2}\cdot(n+1)} + e^{-\frac{i\pi}{2}\cdot(n+1)}}{2} = \cos\left(\frac{\pi(n+1)}{2}\right).$$

Thus $a_n = 2^n + \cos\frac{\pi(n+1)}{2}$.

**Tutorial Exercise T4.3**

Given an array $a$ of length $n$, an algorithm compares all pairs $(a[i], a[j])$ for all $i < j \leq n$, and then calls itself recursively on all proper prefixes of $a$.
How often does the algorithm compare two pairs? Use the repertoire method!

**Solution**

The recurrence is $R_0 = 0$ and

$$R_n = \binom{n}{2} + \sum_{k=0}^{n-1} R_k$$

for $n \geq 1$. Testing a couple of values for $R_n$, we obtain the repertoire:

| $R_0$ | $R_n$ | $f_n = R_n - \sum_{k=0}^{n-1} R_k$ |
|---|---|---|
| 1 | $a^n$ | $\frac{2-a}{1-a} a^n - \frac{1}{1-a}$ |
| 1 | $2^n$ | 1 |
| 1 | 1 | $1 - n$ |
| 0 | $n$ | $n - \binom{n}{2}$ |

To get $\binom{n}{2}$, we can use the last line. To get rid of the linear and constant factors, we use the third and finally the second line, and obtain

$$-\left(n - \binom{n}{2}\right) - (1 - n) + 1 = \binom{n}{2}.$$

Fortunately, this also holds for $R_0 = 0$, and the solution is

$$R_n = 2^n - n - 1.$$

**Homework Exercise H4.1**
Solve the following recurrence: Let $a_0 = 0$, $a_1 = 3$ and

$$a_n = 4a_{n-1} - 4a_{n-2} \text{ for } n > 1.$$

**Solution**
The characteristic polynomial is $x^2 - 4x + 4 = (x-2)^2$ with the only root $x_0 = 2$. The solution is therefore of form $a_n = \lambda 2^n + \mu n 2^n$. Our constraints yield $\lambda = 0$ and $\mu = \frac{3}{2}$.

**Homework Exercise H4.2**
Use the repertoire method to find a closed form for the following recurrence:

$$\begin{aligned} a_0 &= 5 \\ a_1 &= 9 \\ a_n &= na_{n-1} + n^2 a_{n-2} - n^4 - 3n^2 + 5 \qquad \text{for } n \geq 2 \end{aligned}$$

**Solution**
Let $f(n) = -n^4 - 3n^2 + 5$, i.e., $f(n) = a_n - na_{n-1} - n^2 a_{n-2}$.

| $a_n$ | $f(n)$ | $a_0$ | $a_1$ |
|---|---|---|---|
| 1 | $-n^2 - n + 1$ | 1 | 1 |
| $n$ | $-n^3 + n^2 + 2n$ | 0 | 1 |
| $n^2$ | $-n^4 + 3n^3 - n^2 - n$ | 0 | 1 |

Let $Z_i$ for $i = 1, 2, 3$ be the solutions of the first, second, and third line, respectively. Then $f(n) = 5Z_1 + 3Z_2 + Z_3$. For these, $a_0$ and $a_1$ are correct, and thus $a_n = 5 \cdot 1 + 3n + n^2$.

**Homework Exercise H4.3**
Solve the following recurrence and find a nice representation of the solution (in a mathematical sense).

$$\begin{aligned} c_0 &= 2 \\ c_1 &= 4 \\ c_n &= c_{n-2}^{\log c_{n-1}} \end{aligned}$$

Hint: Let $F_n$ be the $n$th Fibonacci number. Write $c_n$ as some function of $F_n$.

**Solution**
We apply the logarithm and obtain

$$\log c_n = \log c_{n-1} \cdot \log c_{n-2}.$$

In order to obtain a sum instead of a product, we repeat this and obtain

$$\log \log c_n = \log \log c_{n-1} + \log \log c_{n-2}.$$

Substituting $d_n = \log \log c_n$ yields

$$\begin{aligned} d_0 &= 0 \\ d_1 &= 1 \\ d_n &= d_{n-1} + d_{n-2} \end{aligned}$$

Since this describes the Fibonacci numbers, we obtain $c_n = 2^{2^{F_n}}$, where $F_n$ denotes the $n$-th Fibonacci number.