Aufgabe 7.2

Annahme: Es wird im folgenden verwendet, dass für die Betrachtung der Komplexität des Sortieralgorithmus nur die Vergleiche zwischen Elementen des zu Sortierenden Vektors relevant sind nicht aber die Vergleichsoperationen in der for-Schleife oder der If-Abfrage, diese ändern das Ergebnis nur um einen konstanten Faktor und bringen keinen zusätzlichen Erkenntnisgewinn.
Es gilt für den Fall, dass der Vektor bereits sortiert ist, dass

$$f_1(n) = n - 1 \in \Omega(n). \tag{1}$$

Es muss hier die Ω-Notation verwendet werden, weil man am Ende wissen will, zwischen welchen beiden Schranken die Funktion wächst. Die untere Schranke legt man dann durch Ω fest, die Obere durch $\mathcal{O}$.
Die Maximale Anzahl an Vergleichen wird benötigt, wenn der Vektor von hinten nach vorne sortiert ist, damit ergibt sich

$$f_2(n) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2} \in \mathcal{O}(n^2). \tag{2}$$

Für die zufällige Sortierung ergibt sich

$$f_3(n) = \sum_{k=1}^{n-1} \frac{k}{2} = \frac{n(n-1)}{4} \in \mathcal{O}(n^2) \tag{3}$$

Der Code in „sort_time.cpp“gab die folgenden Laufzeitsergebnisse mit Codeoptimierung:

bestcase		typical case		worstcase		std::sort	
n	Laufzeit t[s]	n	Laufzeit t[s]	n	Laufzeit t[s]	n	Laufzeit t[s]
10000000	0.120	5000	0.0540	5000	0.109	500000	0.211
25000000	0.301	10000	0.216	10000	0.433	1000000	0.444
50000000	0.599	15000	0.488	15000	0.976	1500000	0.681
75000000	0.901	20000	0.867	20000	1.74	2000000	0.926
100000000	1.203	25000	1.35	25000	2.72	2500000	1.17

und ohne:

bestcase		typical case		worstcase		std::sort	
n	Laufzeit t[s]	n	Laufzeit t[s]	n	Laufzeit t[s]	n	Laufzeit t[s]
50000000	0.0748	20000	0.0890	20000	0.170	500000	0.0387
100000000	0.150	40000	0.340	40000	0.690	1000000	0.0818
250000000	0.372	60000	0.774	60000	1.56	1500000	0.126
500000000	0.738	80000	1.38	80000	2.79	2000000	0.171
750000000	1.11	100000	2.17	100000	4.37	2500000	0.217

damit ergeben sich für die optimierte Version

$$C_1 = 1.20 \cdot 10^{-8}\text{s} \tag{4}$$

$$C_2 = 2.16 \cdot 10^{-9}\text{s} \tag{5}$$

$$C_3 = 4.34 \cdot 10^{-9}\text{s} \tag{6}$$

$$C_4 = 4.51 \cdot 10^{-7}\text{s} \tag{7}$$

wobei $i = 1$ dem best case, $i = 2$ dem typical case, $i = 3$ dem worstcase und $i = 4$ der std-Sortierung entspricht. Für die nicht-optimierte Variante ergeben sich

$$C_1 = 1.49 \cdot 10^{-9}\text{s} \tag{8}$$

$$C_2 = 2.17 \cdot 10^{-10}\text{s} \tag{9}$$
$$C_3 = 4.33 \cdot 10^{-10}\text{s} \tag{10}$$
$$C_4 = 1.36 \cdot 10^{-8}\text{s} \tag{11}$$

Außerdem sieht man bei Berechnung der Werte, dass sie unabhängig von n sind. Damit ergeben sich mit Optimierung

$$t_1(n) = 1.20 \cdot 10^{-8}\text{s} \cdot n \tag{12}$$
$$t_2(n) = 2.16 \cdot 10^{-9}\text{s} \cdot n^2 \tag{13}$$
$$t_3(n) = 4.34 \cdot 10^{-9}\text{s} \cdot n^2 \tag{14}$$
$$t_4(n) = 4.51 \cdot 10^{-7}\text{s} \cdot n \log(n) \tag{15}$$

und ohne Optimierung

$$t_1(n) = 1.49 \cdot 10^{-9}\text{s} \cdot n \tag{16}$$
$$t_2(n) = 2.17 \cdot 10^{-10}\text{s} \cdot n^2 \tag{17}$$
$$t_3(n) = 4.33 \cdot 10^{-10}\text{s} \cdot n^2 \tag{18}$$
$$t_4(n) = 1.36 \cdot 10^{-8}\text{s} \cdot n \log(n). \tag{19}$$

Löst man nun die Gleichung

$$t_2(n) = t_4(n) \tag{20}$$

so erhält man, dass für den typtischen Fall die Insertionsortvariante mit Optimierung bis $n \approx 1531$ mit std::sort mithalten kann, ohne Optimierung nur bis $n \approx 371$. Genaue Zahlenwerte hängen jedoch vermutlich auch signifikant von der verwendeten Hardware ab, weshalb diese Ergebnisse im Detail (von den genauen Zahlenwerten her) vermutlich nicht übertragbar sind.