Übersichtsblatt:
Insertion Sort

Idee:

Der Insertion Sort ist ein Sortieralgorithmus der auf das folgende Prinzip aufbaut:
Es gibt ein Feld und ein unsortiertes Feld. Man nimmt ein Element aus dem unsortierten Feld, und geht dann das sortierte Feld, von links nach rechts, ab, bis man die Stelle Gefunden hat an die das Element gehört. Dann nimmt man sich das nächste Element aus dem unsortierten Feld und wiederholt den Vorgang so lange bis das unsortierte Feld leer ist.

Pseudocode:

Eingabe: Feld A mit n Komponenten
Ergebnis: Sortiertes Feld A' mit A'[0] ≤ A'[1] ≤ … ≤ A'[n-1]

```
 1| for(j = 1 to n-1){
 2|     key = A[j];
 3|     i = j - 1;
 4|     while( (i >= 0) && (A[i] > key) ){
 5|        A[i+1] = A[i];
 6|        i--;
 7|     }
 8|     A[i+1] = key;
 9| }
10| return A;
```

Analyse:

Zur Laufzeitanalyse des Algorithmus wollen wir uns wie viele Schritte nötig sind um ihn komplett auszuführen wobei t_j im Folgenden die Ausführungsanzahl der while-Schleife sei:

Zeile 1:	n	Schritte
Zeile 2:	n-1	Schritte
Zeile 3:	n-1	Schritte
Zeile 4:	$\sum_{j=1}^{n-1} t_j$	Schritte
Zeile 5:	$\sum_{j=1}^{n-1} (t_j - 1)$	Schritte
Zeile 6:	$\sum_{j=1}^{n-1} (t_j - 1)$	Schritte
Zeile 8:	n-1	Schritte
Zeile 10:	1	Schritte
Insgesamt:	$\frac{2}{3}n^2 + \frac{7}{2}n - 3$	Schritte

Bemerkung:

Auf die Detaillierte Darstellung aller Rechenschritte wurde hier aus Platzgründen verzichtet.

Es ist zu beachten das:

$$\sum_{i=1}^{n-1} j = \frac{n(n-1)}{2} \text{ und}$$

$$\sum_{i=1}^{n-1} (j+1) = \frac{n(n-1)}{2} + (n-1)$$

Dies ist zumindest im worst-case der Fall. Der Einfachheit halber wollen wir uns hier nicht mit anderen Analysen wie z.B. der durchschnittlichen Laufzeit des Algorithmus aufhalten. Da wir Informatiker ja geborene Optimisten sind beschränken wir uns generell meist auf die Analyse des schlechtesten Falls.

Man kann die obere Laufzeitanalyse auch in einer etwas vereinfachten Form darstellen, in dem man auf die so genannte O-Notation zurückgreift. Die Anweisungen in der for-Schreife werden n-mal aufgerufen. Dazu zählt auch die while-Schleife. Die Anweisungen in der while-Schleife werden dann auch n-mal aufgerufen. Die Einzelschritte können angesichts dieser Erkenntnis vernachlässigt werden. Und Was für die Laufzeit als wesentliche Information bleibt ist:

$$\text{Laufzeit} = n \cdot n = n^2$$

Diesen Sachverhalt drückt man als **$O(n^2)$** aus.