Sortierverfahren und Komplexität von Algorithmen

Wir haben uns folgendes klargemacht:
1. Es gibt unterschiedliche Möglichkeiten, eine Menge von Objekten automatisch zu sortieren.
Dazu gehört:
1a: Insertion-Sort
Funktionsweise am Beispiel: alphabetische Sortierung einer Wortliste.
Man beginnt mit einem ersten Wort. Dann schaut man sich das zweite Wort an und stellt fest, dass man dies entsprechend vor oder nach dem ersten Wort einsortiert. Dann hat man bereits eine "sortierte Liste" bestehend aus 2 Wörtern. Man nimmt dann wiederum das nächste "unsortierte Wort" und fügt ("insert") es in die Liste der bereits sortierten Worte ein. Man fährt fort, bis die "unsortierte Liste" leer ist.

1b: Bubble-Sort
Funktionsweise:
Man vergleicht erstes und zweites Wort und tauscht ggf..
Nächster Vergleich: 2. mit 3. Wort
Nächster Vergleich: 3. mit 4. Wort
.... bis man ganz durch ist. Dann steht das alphabetisch letzte Wort bereits an der richtigen Stelle.
Dieser Vorgang muss so oft wiederholt werden, wie es Elemente (-1) in der Wortliste gibt.

2. Anzahl der Vergleiche
Bei Bubble-Sort war es einfach, die Anzahl der Vergleiche zu zählen. Beispielsweise konnten 10 Elemente mit 45 Vergleichen sortiert werden. Bei 20 Elementen benötigt man schon 190 Vergleiche. Bei n Elementen benötigt man $\frac{1}{2}\cdot((n-1)^2 + (n-1)) = \frac{1}{2}\cdot n^2 - \frac{1}{2}\cdot n$ Vergleiche.

3. Akinator
Wir haben uns den "Akinator" angesehen, der durch mehrmaliges Fragen eine Person aus "deinen Gedanken erraten" kann. Angenommen (Optimalfall!), jede Frage teilt die Menge der noch verbleibenden Personen in zwei gleichgroße Teile (wovon nur noch ein Teil, also die Hälfte, für die nächste Frage relevant ist). Dann benötigt man bei 8 Personen 3 Fragen. Bei 32 Personen benötigt man 5 Fragen (Denn 2 hoch 5 = 32 und log2(32) = 5) und bei n Personen benötigt man Aufrunden(log2(n)) Fragen.

4. Das TSP (Travelling-Salesman-Problem)
Beim TSP kann man durch geschickte Ideen gute Lösungen finden. Aber man weiß erstmal nicht, ob die gefundene Tour die optimalste Tour ist.
Es ist bislang kein Algorithmus bekannt, der die optimale Tour findet, ohne alles durchzuprobieren.
Die Probierlösung (bei z.B. 58 Städten) funktioniert wie folgt:
Man startet mit einer ersten Stadt.
- als nächste Station kommt eine von 57 Städten in Frage
- bei jeder der 57 Teilstrecken kommt eine von 56 Folgestädten in Frage
- entsprechende Fortführungen liefert folgende Anzahl von Wegen: $57 \cdot 56 \cdot 55 \cdot \ldots \cdot 3 \cdot 2 \cdot 1$
- Mathematisch beschrieben ist das „57 Fakultät“ bzw. 57!

5. Komplexitätsbetrachtungen:
Beim Bubble-Sort mit n Elementen kommt man auf folgende Formel:
$$\text{AnzVergleiche}(n) = \frac{1}{2}\cdot n^2 - \frac{1}{2}\cdot n$$
Beim Akinator mit n Personen in der Datenbank kommt man auf die Formel:
$$\text{AnzFragen}(n) = \text{Aufrunden}(\log_2(n))$$
Beim Travelling-Salesman-Problem mit n Städten kommt man auf die Formel:
$$\text{AnzVergleichsstrecken}(n) = (n-1)!$$

Das n in den obigen Formeln heißt "Problemgröße".
Da die sogenannte "binäre Suche" (z.B. im Telefonbuch) mit jedem Schritt die Menge der zu untersuchenden Namen halbiert, gilt hier die gleiche Formel wie beim Akinator. Man könnte auch sagen, Akinator und binäre Suche sind "strukturgleich".

Für die Problemgröße n=1000 erhalten wir:
-- beim Bubblesort: AnzVergleiche(1000) = 499500
-- beim Akinator: AnzFragen(1000) = 10
-- beim TSP: AnzVergleichsstrecken(n) = (zu hoch für GTR)

Die Ergebnisse sind offenbar völlig unterschiedlich. Wenn man diese drei Algorithmen vergleicht, so kann man sagen, dass das Akinator-Vorgehen in einer gewissen Art "besser" oder "schneller" oder "effizienter" als der Bubble-Sort ist. Natürlich sind die Anwendungsgebiete ("Sortieren" oder "Personen erraten") völlig unterschiedlich. Dennoch kann man die beiden Formeln vergleichen und sagen: Für große Werte von n benötigt man beim Akinator weniger "Schritte".
Wenn man solche Formeln in der Theoretischen Informatik vergleicht, so interessiert man sich bei der Bubble-Sort-Formel gar nicht mehr für den Vorfaktor ½ oder den zweiten Summanden " - ½n", da dies im Vergleich zu anderen Formeln (wie die Akinator-Formel) gar nicht mehr ins Gewicht fällt. Man sagt (ungenau): Bubble-Sort hat eine Komplexität von "n^2", während Akinator eine Komplexität von "log(n)" hat.

6. Herausforderungen:
Wir stellen fest, dass der Bubble-Sort selbst bei moderaten Werten für n schon verflixt viele Schritte benötigt (ganz zu schweigen vom TSP). Da fragt man sich, ob es neben Bubble-Sort noch andere Sortierverfahren gibt, die "schneller sind", also ein besseres Komplexitätsverhalten haben. Die gibt es! Das kommt erst in der Q1, aber man kann schonmal sagen: Die Sortierverfahren "Quicksort" oder "Mergesort" haben eine Komplexität von "$n \cdot \log_2(n)$".
Wir vergleichen mal: Angenommen, wir sortieren 1000 Elemente. Dann ergibt sich mit der "ungenauen" Bubblesort-Komplexität von n^2 eine Größenordnung von 1Mio Vergleichen. Mit der "ungenauen" Quicksort-Komplexität "$n \cdot \log_2(n)$" ergibt sich $1000 \cdot 10 = 10$tsd (also 50 mal besser). Das ist schon schneller. Für höhere Werte von n ergibt sich:

n = 10000
Bubble-Sort: n^2 → 100 000 000
Quicksort: $n \cdot \log_2(n)$ → 140 000

n = 1Mio
Bubble-Sort: n^2 → 1 Billion = 1 000 000 000 000
Quicksort: $n \cdot \log_2(n)$ → ca. 1 Mio · 20 = 20 000 000
also ca. 50tsd mal besser!

Man sagt daher auch: Quicksort (oder Mergesort) liegt in einer anderen Komplexitätsklasse als Bubble-Sort.