Söftw.
1

Greedy-Schema

1) Behandle einfache/triviale Fälle.
2) Reduziere das Problem auf eine kleinere Größe.
3) Löse das Problem rekursiv.

Divide and Conquer Schema

1) Behandle einfache/triviale Fälle.
2) Divide: Teile das Problem in zwei oder mehr Teilprobleme ungefähr gleicher Größe auf.
3) Conquer: löse die Teilprobleme.
4) Füge die Teillösungen zu einer ganzen Lösung zusammen.

Quick-Sort Algorithmus

1) Wahl d. Pivot-Elements:
   - Zufall
   - Median aus 3 (z.B. Anfang, Mitte, Ende)
2) Divide: Wähle ein Element vor dem Pivot Element, das größer ist ~~da~~ und eines nach dem Pivot, das kleiner ist. Vertausche die Werte. Wiederhole bis partitioniert.
3) Conquer: Starte für jedes Teilfeld neu bei 1). Wiederhole ~~1 bis~~ 3 solange bis nur noch 2er Felder übrig sind.
4) Setze zusammen.

Quick-Sort Komplexität (wann?)

1) Günstigster Fall: $O(N \cdot \log(N))$
2) Durchschnittlicher Fall: $O(N \cdot \log(N))$
3) Schlechteste Fall: $O(N^2)$

~~Ab~~ ① wenn der vordere und hintere Teil d. Feldes die gleiche Größe besitzen.
③ Wenn ~~das~~ konsequent das kleinste ~~oder~~ größte Element als Pivot gewählt wird.

`p->x = 5;`

`(*p).x = 5;`

Softw. 2

Bubble-Sort Algorithmus

1) Vergleiche Felder n und (n+1) für n=0. Vertausche, falls Feld (n+1) < Feld n.
2) Wiederhole für n=0 bis (Größe d. Felds − Anzahl d. Durchläufe)
~~bis das Feld sich durch einen Durchlauf nicht mehr ändert.~~
3) Wiederhole 1 u. 2 bis sich durch einen Durchlauf das Feld nicht mehr ändert.

Bubble-Sort Komplexität

~~a) Günstigste Fall~~
~~Durchschnittliche Fall~~

1) Günstigster Fall: $O(N)$
Feld bereits sortiert, ein Durchlauf.
2) Durchschnittliche Fall: $O(N^2)$
3) Ungünstigste Fall: $O(N^2)$
Umgekehrt angeordnetes Feld
⇒ n−1 Durchläufe.

Überlauf Addition 2er Komplement

- Übertrag in die ganz linke Stelle XOR
- Übertrag aus der ganz linken Stelle

Euklidische Algorithmus

1) Falls A<B, vertausche A und B
2) ersetze B durch A%B und A durch das bisherige B
3) Falls B ≠ 0 gehe zu Schritt 2
~~4)~~ ansonsten ist A das Ergebnis

Sequentielle Suche Komplexität

1) Günstigste Fall: $O(1)$
Erster Eintrag ist gesuchter Wert
2) Durchschnittlicher Fall: $O(N/2) \hat{=} O(N)$
3) Schlechtester Fall: $O(N)$
Needle nicht vorhanden.
Alle werden ~~verglichen~~ mit Needle verglichen.

Binäre Suche Komplexität

~~$O(\log_2 N)$~~
Schlechtester Fall $O(\log_2 N + 1)$

Softw. 3

Selection Sort Algorithmus

1) Minimum suchen.
2) Ersten Eintrag und Minimum tauschen.
3) Ab Eintrag (Anzahl der Durchläufe) ① u. 2 wiederholen.
4) ①-③ wiederholen bis kein Minimum mehr gefunden wird.

Selection Sort Komplexität

~~O(N²)~~ $O(N^2)$ immer

Insetion Sort Algorithmus

1) Vergleiche n und n+1 für n=0. Tausche entsprechend
2) Betrachte n, n=3 und sortiere an die richtigen Stelle zwische 0 und n-1 ein
3) Wiederhole ② bis n = size -1

Insetion Sort Komplexität

1) Beste Fall: ~~O(N)~~ $O(N \cdot \log N)$
Da immer der sortierte Bereich durchsucht werden muss.
2) Durchschnittlicher Fall = Schlechteste Fall $O(N^2)$
Da die Verschiebeoperation beim einfügen von Mal zu Mal komplexer wird.

Merge Sort Algorithmus

1) Divide: In zwei Teile zu legen.
2) Conquer: Rekursiv Teilfelder sortieren
3) ~~Combine~~
Anschließend Teilfelder so mischen, dass das gesamte Feld sortiert ist.

Merge Sort Komplexität

Immer: $O(N \cdot \log N)$
jedoch mit speicherplatz der Größe $O(N)$

Optimierung durch entfernen der Rekursion oder durch verwenden eines anderen Algorithmus bei kleinen Teilfeldern wie bei Binarysearch.