SORTIERVERFAHREN IN PROLOG:

```
/* Erzeugung einer Zufallsliste                                          */
/* N: Anzahl, G: Obergrenze, L: Liste                                    */
zliste(0,G,[]).
zliste(N,G,[Z|Zs]):- N>0, N1 is N-1, random(0,G,Z), zliste(N1,G,Zs).

/* Permutation-Sortierung (hochgradig nichtdeterministisch, n!-Aufwand)  */
/* es werden alle Permutationen probiert, bis die Liste sortiert ist     */
p_sortiert(Xs,Ys):- permutiert(Xs,Ys), geordnet(Ys).
geordnet([_]).                              /* geordnete Liste?          */
geordnet([X,Y|Zs]):- X =< Y, geordnet([Y|Zs]).
ausgewaehlt(X,[X|Xs],Xs).                   /* wählt El. aus Liste aus   */
ausgewaehlt(Y,[X|Xs],[X|Zs]):- ausgewaehlt(Y,Xs,Zs). /* und löscht es dann */
permutiert([],[]).                          /* bildet alle Permutationen */
permutiert(Xs,[Z|Zs]):- ausgewaehlt(Z,Xs,Ys), permutiert(Ys,Zs).

/* Sortierung durch Auswahl (Selection-Sort) (so leider nur n³-Aufwand)  */
/* Aus der Liste wird das kleinste Element herausgenommen (Präd. kleinstes).*/
/* Anschließend wird der Rest der Liste rekursiv sortiert und das kleinste */
/* Element vorne angefügt.                                               */
a_sortiert([X],[X]).
a_sortiert(Xs,[X|Ys]):- kleinstes(Xs,X,Rest),a_sortiert(Rest,Ys).
kleinstes([X],X,[]).     /* liefert kleinstes Element (Problem: n²-Aufwand) */
kleinstes([K|Rs],K,Rs):- kleinstes(Rs,M,Hilfsrest), M>=K.
kleinstes([K|Rs],M,[K|Zs]):- kleinstes(Rs,M,Zs), M<K.

/* Sortierung durch Vert. (Bubble-Sort)  (so nur theoretisch n²-Aufwand) */
/* Aus der Liste wird mit Hilfe von append an einer beliebigen Stelle die */
/* Liste aufgebrochen und die beiden ersten Elemente der zweiten Teilliste */
/* überprüft. Sind diese falsch sortiert, so werden sie in umgekehrter   */
/* Reihenfolge zusammen mit den beiden Teillisten zu einer neuen Liste   */
/* zusammen gefügt und die Sortierung wird rekursiv auf dieser neuen Liste */
/* fortgesetzt.                                                          */
b_sortiert(Xs,Xs):- geordnet(Xs).
b_sortiert(Xs,Ys):- append(As,[X,Y|Rs],Xs), X>Y,
                    append(As,[Y,X|Rs],Xs1),b_sortiert(Xs1,Ys).

/* Sortierung durch Einfügen (Insertion-Sort) (n²-Aufwand)               */
/* Die Liste wird ohne das erste El. rekursiv sortiert. Anschließend wird */
/* das erste Element in diese Liste eingefügt.                           */
i_sortiert([],[]).
i_sortiert([X|Xs],Ys):- i_sortiert(Xs,Zs), eingefuegt(X,Zs,Ys).
eingefuegt(X,[],[X]).                       /* Fügt El. an der richtigen */
eingefuegt(X,[Y|Ys],[X,Y|Ys]):- X =< Y.     /* Stelle ein.               */
eingefuegt(X,[Y|Ys],[Y|Zs]):- X>Y, eingefuegt(X,Ys,Zs).

/* Sortierung durch Teilungsverfahren (Quick-Sort)  (n*log(n)-Aufwand)   */
/* Die Liste wird unter Verwendung des ersten Elements als Mittenelement */
/* in kleine und große Elemente aufgeteilt. Beide Teillisten werden rekursiv*/
/* sortiert und anschließend mit dem Mittenelement wieder zusammengefügt. */
q_sortiert([],[]).
q_sortiert([X|Xs],Ys):- zerlegt(Xs,X,Kleine,Grosse),
                        q_sortiert(Kleine,Kleine_s),
                        q_sortiert(Grosse,Grosse_s),
                        append(Kleine_s,[X|Grosse_s],Ys).
zerlegt([],X,[],[]).   /* zerlegt eine Liste in klein/groß mit X als Vgl-El.*/
zerlegt([Z|Zs],X,[Z|Ks],Gs):- Z=<X, zerlegt(Zs,X,Ks,Gs).
zerlegt([Z|Zs],X,Ks,[Z|Gs]):- Z>X, zerlegt(Zs,X,Ks,Gs).
```