Übersichtsblatt:
Merge Sort

Idee:

Der Mege Sort ist ein Sortieralgorithmus der nach dem Divide-And-Coquer Prinzip funktioniert. Diese Prinzip umfasst generell die 3 Teilschritte Divide, Conquer und Combine. Beim Merg Sort stellt sich das folgendermaßen da:

Divide: Ein zu sortierendes Array wird in zwei Teilarrays mit je n/2 Elementen zerlegt.
Conquer: Sortiere Die Subarrays mit Merg Sort.
Combine: Setzte die zwei sortierten Subarrays wieder zusammen.

Dem geneigten Leser ist also aufgefallen, dass wir hier eigentlich zwei Algorithmen benötigen. Einen zum Sortieren und einen zum Mischen, der bereits sortierten Felder.

Pseudocode:

Merg:

Eingabe: Zwei sortierte Folgen F_1 und F_2.
Ergebnis: Eine sortierte Folge F, die die Elemente von F_1 und F_2 enthält.

```
int merg(F1 ,F2){
   F = leere Folge;
   while( (F1 nicht leer) && (F2 nicht leer) ){
      if( Anfangselement F1 < Anfangselment F2 ){
         t = Anfangselement F1;
         lösche Anfangselement F1;
      }else{
         t = Anfangselement F2;
         lösche Anfangselement F2;
      }
      hänge t an F an;
   }
   hänge eine evtl. verbleibende Restfolge an F an;
   return F;
}
```

Merg-Sort:

Eingabe: Eine unsortierte Folge F.
Ergebnis: Eine sortierte Folge F', die alle Elemente von F enthält.

```
int mergSort(F){
   if( F hat genau ein Element ){
      return F;
   }else{
      halbiere F in F1 und F2;
      F1 = mergSort(F1);
      F2 = mergSort(F2);
      return merg(F1, F2);
   }
}
```

Analyse:

Da Merg Sort ein rekursiver Algorithmus ist, ist eine Analyse nicht ganz einfach, wir beschränken uns daher auf die Anzahl der Vergleiche von zwei Elmenten.

$$V(n) = 2 \cdot V(n/2) + n \quad n \geq 2$$
$$V(2^N) = 2V(2^{N-1}) + 2^N$$
$$\Leftrightarrow \frac{V(2^N)}{2^N} = \frac{V(2^{N-2})}{2^{N-1}} + 1$$

Nach einigen kleinen Umformungen ergibt sich hieraus eine Anzahl von Vergleichen $V(n) = n\log_2(n)$. Der Mergsort hat also eine Laufzeit von $O(n\log_2(n))$

Annahme:

n ist eine Zweierpotenz, so dass wir das Feld immer zwei gleichgroße Stücke aufteilen können.

$n = 2^N$