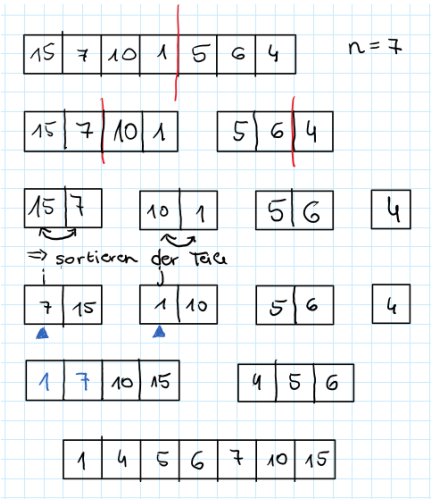
SWE Übung 2

Inhaltsverzeichnis

External-Mergesort

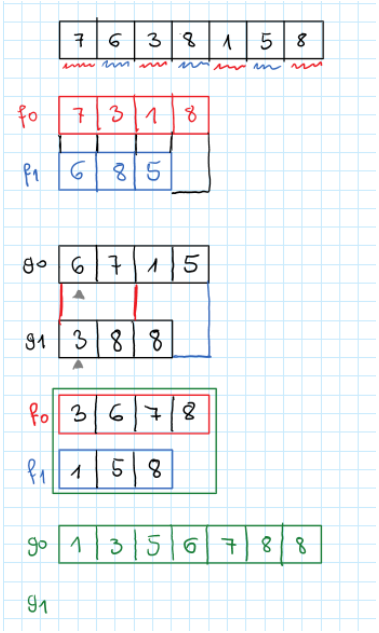
Lösungsidee

Der External Mergesort ist eine Abwandlung des Mergesort, bei dem größere Datenmengen sortiert werden können, da immer nur ein Teil einer großen Datei eingelesen, sortiert und in eine weitere Datei gespeichert wird.



Beim Mergesort wird ein Vektor in kleine Teile zerlegt, die dann sortiert und wieder zusammengefügt werden.
Im Beispiel links sieht man einen Vektor der Größe sieben, der rekursiv in immer kleinere Teile zerlegt wird, bis es sich um Vektorteile der Größe zwei, bzw. eins – wenn die Größe des Ursprungsvektors ungerade war – handelt.

Diese Vektorteile werden dann aufsteigend sortiert und schließlich werden zwei Teile zu einem größeren Teil zusammengefügt und wieder sortiert. Dass passiert so lange, bis die Teile wieder zu einem großen Teil der Größe des Ursprungsvektors zusammengefügt werden. Danach handelt es sich um einen aufsteigend sortierten Vektor.



Beim External Mergesort befinden sich die Daten in einer Datei, die dann Teilchenweise eingelesen werden. Bevor mit dem Sortieren begonnen werden kann, müssen die Daten der Ursprungsdatei in zwei Dateien aufgeteilt werden, und zwar so, dass die Daten immer abwechselnd in die eine Datei und in die andere Datei gespeichert werden, wie man im Beispiel links sieht.

Im Beispiel links wurden die Daten der Inputdatei auf die Dateien f0 und f1 aufgeteilt. Um die Daten sortieren zu können, werden zwei weitere Dateien gebraucht, im Beispiel links die Dateien g0 und g1.

Weiters wird noch eine Lauflänge benötigt, die angibt, wie viele Daten jeweils von den beiden Dateien eingelesen werden. Zu Beginn ist die Lauflänge eins, nach jedem Durchgang verdoppelt sich die Lauflänge, so lange, bis die Lauflänge größer oder gleich der Anzahl an Daten in der Ursprungsdatei sind.

Es werden immer genau so viele Daten von den beiden Dateien eingelesen, wie die Lauflänge vorgibt. Diese Daten werden dann miteinander verglichen und dementsprechend sortiert. Im Beispiel werden zuerst die Zahl 7 von der Datei f0 und die Zahl 6 von der Datei f1 eingelesen. Diese beiden Zahlen werden sortiert, sodass zuerst 6 und danach 7 kommt, und danach in die Datei g0 gespeichert. Danach werden die nächsten beiden Zahlen von den beiden Dateien eingelesen, sortiert und in die andere Datei g1 gespeichert. Das passiert so lange, bis das Ende der Dateien erreicht wurde. Danach werden die Input-Dateien zu Output-Dateien und umgekehrt, also im Beispiel wird nun von den Dateien g0 und g1 eingelesen und in die Dateien f0 und f1 geschrieben. Die Lauflänge wird erhöht und somit werden nun jeweils zwei Daten von den Dateien g0 und g1 eingelesen. Diese werden auch wieder sortiert und gespeichert. Das passiert so lange, bis alle Daten sortiert in einer Datei stehen und die andere leer ist, wie z.B. im Beispiel am Schluss alle Daten in der Datei g0 stehen und die Datei g1 leer ist.

Testfälle

zufällige Datei erstellen

siehe Datei *„test_partition.txt“*

Werte einer Datei alternierend aufteilen

siehe Dateien *„partition_0.txt“* und *„partition_1.txt“*

Datei kopieren

siehe Datei *„test_copy.txt“*

Datei drucken

```
c S a x L k I h m w
```

nicht existierende Datei

```
file doesn't exist
```

External Mergesort

Testfall 1

```
qP ah vi kw LO nS Hr mz FM br NK Ol KZ Ar Ry os VG JS he nR Zg Fx RC xa fV aP VO Hf bM UU kj Ux pn Oi iZ DR HV EN ln YZ
iC JH Jt VQ Zv tw to Mt uK UY XX RM wX GT nx Mk jt mI tK SM oP kZ Nu pW eu lW NA rq gq zR HV Si ZD Ca Vt go Az GH KY vC
Kx io bU EY um TL Gd oI VS lA fo Qh Jt Ae GF FM js Rh HZ va

Ae Ar Az Ca DR EN EY FM FM Fx GF GH GT Gd HV HV HZ Hf Hr JH JS Jt Jt KY KZ Kx LO Mk Mt NA NK Nu Oi Ol Qh RC RM Rh Ry SM
Si TL UU UY Ux VG VO VQ VS Vt XX YZ ZD Zg Zv aP ah bM bU br eu fV fo go gq he iC iZ io js jt kZ kj kw lA lW ln mI mz nR
nS nx oI oP os pW pn qP rq tK to tw uK um vC va vi wX xa zR
```

Oben sieht man alle Daten der Datei, bisher in unsortierter Version und unten sieht man diese Daten in sortierter Reihenfolge.

Testfall 2

```
c S a x L k I h m w

I L S a c h k m w x
```

Testfall 3

```
wds FST CsW YnN BeS JZD JYZ rII iQo ovP NxV YkU eey wNj esM WJz uGh SSU IJt szA sTg IFj qEM rum cjY VFQ cPK ajY vRg gVP
UcV UTh jUJ oyh Grr EzU ivy RTu LLS HVv uyN eUu Vhw THk VQi rLs xpr zlJ xLv qiG iVQ RzR iWI CYo Jed mXt ROx bqH dXR oyk
CmL AWA VDh rIG MHf MSM pRF NId pLl maB njP uFe jrd vDK GqS Ngi Eao BIT EES HDQ fcp BmN FIP Sjx DTA Xus qfp KoO JgC Dom
fvK waS QkW znW gpu hRC oFu weD Hby LSA

AWA BIT BeS BmN CYo CmL CsW DTA Dom EES Eao EzU FIP FST GqS Grr HDQ HVv Hby IFj IJt JYZ JZD Jed JgC KoO LLS LSA MHf MSM
NId Ngi NxV QkW ROx RTu RzR SSU Sjx THk UTh UcV VDh VFQ VQi Vhw WJz Xus YkU YnN ajY bqH cPK cjY dXR eUu eey esM fcp fvK
gVP gpu hRC iQo iVQ iWI ivy jUJ jrd mXt maB njP oFu ovP oyh oyk pLl pRF qEM qfp qiG rIG rII rLs rum sTg szA uFe uGh uyN
vDK vRg wNj waS wds weD xLv xpr zlJ znW
```

Testfall 4 (nicht existierende Datei)

```
file doesn't exist
```