2.)

Da in der Beispieldatei lediglich Zahlen gespeichert werden , besteht die Datei somit nur aus 12 verschiedenen Symbolen (0,...,9 , SPACE , Zeilenumbruch).

Die einfachste Möglichkeit :
Datei durchlaufen , Anzahl der verschiedenen Symbole bestimmen und mit der dementsprechenden minimal benötigten Bit-Anzahl codieren.
Statt ein Zeichen mit 8Bit zu codieren , reichen in der Beispieldatei 4 Bit (2^4=16).
D.h. mit dieser naivsten Vorgehensweise erzielt man bereits eine Kompressionsrate von 50% .

Mit der Huffman-Codierung auf den einzelnen Zeichen erziele ich eine Packgrösse von 1.204.254 Bytes , was einer Kompressionsrate von 41,3% entspricht.
(Mit einem Markov Modell höherer Ordnung wuerde man wahrscheinlich bessere Kompressionsraten erzielen , da das MM 0.Ordnung 12 Zustände besitzt , das MM 1.Ordnung 140 , das MM 2.Ordnung 962 ...usw)

D.h. z.B. für das MM 2.Ordnung : 962 Zustände sind mit 10 Bit codierbar (und ein Zustand entspricht drei aufeinanderfolgenden Bytes) , d.h. selbst bei angenommener , unrealistischerbei Gleichverteilung dieser Zustände beträgt die Kompressionsrate 10/24 = 41,7% .
Würde man die HuffmanCodierung oder arithmetische Codierung auf dieses MModell anwenden , käme sicherlich eine weit bessere Kompressionsrate zustande.

MarkovModell höherer Ordnung wären bei der gegebenen Beispieldatei möglich (da nur 12 Symbole) , im allgemeinen Fall jedoch würde dies jedoch einen exorbitanten Speicherbedarf bedeuten .
(256 Symbole MM 2.Ordnung : 256^3*sizeof(float) = 67MByte)

Da ich keine schon vorher bekannten Spezialstrukturen oder Besonderheiten der Datei der Dateien in den Codec einfliessen lassen will und den Speicherbedarf möglichst niedrig halten will , entscheide ich mich für einen allgemeinen Kompressions-Algorithmus , welcher dem sehr effizienten BZIP2 nachempfunden ist.

Aufbau :

1. Burrows-Wheeler Transformation , um die Datei für eine Lauflängencodierung vorzubereiten
2. Einen MoveToFront Encoder darüberlaufen lassen (häufigste Symbole nach vorn holen)
3. Das Ergebnis lauflängencodieren (1. & 2. haben hierfür Vorarbeit geleistet)
4. Und schliesslich mit einem adaptiven arithmetischen Codierer codieren

Bei der gegebenen Beispieldatei kommt mein Codec auf 844.271 Bytes , was einer Kompressionsrate von 28,9% entspricht.
In meinen Codec fliessen somit keine zusätzlichen Informationen über die Datei ein , die Eingabe besteht lediglich aus einer x-beliebigen Datei.

Der Codec besteht aus folgenden Dateien :

BWT.h	- Burrows-Wheeler Transfromation
MTF.h	- MoveToFront Codierer
RLE.h	- Lauflängen Codierer
ArithmeticCodec.h	- arithmetischer Codierer
decode.cpp	- Quellcode zum Packen
encode.cpp	- Quellcode zum Entpacken

Benutzung :

encode <inputFile> <outputFile> bzw.
decode <inputFile> <outputFile>

Beispiel :

encode testinput.txt pack.txt
decode pack.txt entpack.txt