Descrizione dell’algoritmo dell’Homework 4 obbligatorio :

Descriverò funzione per funzione in ordine di definizione.

Funzione Exctract\_string\_from\_file () :

Tale funzione (ricorsiva) prende in input una stringa contenete il nome del file da controllare per 2 volte (prende 2 stringe contenenti lo stesso valore, una di esse rappresenta il file name iniziale che durante ogni ricorsione non verrà variato), ed una lista di stringhe per riferimento.

Proseguendo, apriamo il file tramite la stringa passata in input, Lo inseriamo in una variabile alla quale applicheremo il metodo read(), applicando poi al risultato il metodo split(), ottenendo una lista temporanea contenete tutte le stringhe di quel suddetto file. Controlliamo poi che valore assume il primo elemento della lista temporanea (che sarebbe il filename del prossimo file da controllare), se esso è uguale al file name iniziale, vuol dire che siamo giunti all’ultimo file da controllare, allora sommiamo alla lista di stringhe presa per riferimento la lista temporanea contenete tutte le stringhe del file che stiamo controllando eccetto il primo elemento, chiamiamo poi il return per sancire la fine della funzione.

Diversamente, se il valore assume il primo elemento della lista temporanea è diverso dal file name originale, sommiamo alla lista di stringhe presa per riferimento la lista temporanea contenete tutte le stringhe del file che stiamo controllando eccetto il primo elemento, e richiamiamo poi ricorsivamente la funzione Funzione Exctract\_string\_from\_file (), passando come filename il primo elemento della lista temporanea, come file name iniziale lo stesso valore che ha assunto inizialmente, e come lista di stringhe la stessa alla quale stiamo continuando a sommare valori.

Tale funzione non ritorna nessun valore, ma modifica la lista di stringhe passata esternamente, aggiornandola con le stringhe controllate nei file.

Funzione stringList\_to\_finalString () :

Tale funzione prende in input una lista di stringhe appena generata tramite il metodo “Exctract\_string\_from\_file()”, viene calcolata poi la lunghezza della stringa più lunga contenuta all’interno della lista di stringhe, e viene inizializzata una lista di dizionari, con all’interno tanti dizionari (al momento vuoti) quanti sono i caratteri della stringa più lunga contenuta all’interno della lista di stringhe.

La lista di dizionari, sarà utilizzata in modo che ogni indice della lista, indica il dizionario relativo ai caratteri del medesimo indice nella stringa.

Proseguendo, si itera per ogni stringa della lista di stringhe, e all’interno di tale iterazione, si itera per ogni carattere della stringa, tenendo conto però dell’indice del carattere tramite il metodo enumerate().

Per ogni carattere si controlla se è all’interno del dizionario, nell’indice stesso a quello del carattere, è presente il carattere stesso, se no, al dizionario (quindi alla lista di dizionari con l’indice uguale a quello del carattere all’interno della stringa ) viene aggiunto un nuovo componente, che ha come chiave il carattere, e come valore 1, tale valore indica la frequenza per quell’indice, e si setta ad uno perché è la prima volta che tale carattere appare nel dizionario. De facto, se il carattere è già presente all’interno del dizionario, piuttosto che aggiungere il valore uno, si aggiunge il valore precedente ma sommato ad uno, così da far noto che quel carattere è apparso più volte in quella posizione.

Finite tutte le nostre iterazioni, ci troveremo una lista di dizionari, ogni indice della lista equivale alla posizione del carattere all’interno della stringa, ed ogni dizionario contiene come chiave il carattere e come valore la frequenza.

Una volta fatto ciò, controlliamo la nostra lista dizionari per ogni indice, e da ogni dizionario estrapoliamo il valore con più alta frequenza, inserendolo in una lista di caratteri, utilizzo una lista perché nel caso ci siano 2 caratteri con la stessa frequenza, utilizzerò il metodo sort() sulla lista.

Alla fine di ogni iterazione, aggiungo ad una stringa creata in precedenza il primo valore della lista di caratteri (così da aggiungere il carattere alfabeticamente ordinato nel caso ci siano due caratteri con la stessa frequenza.)

Alla fine, avrò come risultato la stringa che per ogni posizione, avrà il carattere più frequente all’interno di tutte le stringhe lette dai file.

Funzione most\_frequent\_chars () :

Questa è la funzione principale del nostro programma, prende come input il nome del file chiamato “filename”, crea una lista di stringhe vuota, ed insieme al “filename” viene passata come parametro al metodo “Exctract\_string\_from\_file()”(si ricordi che il “filename” viene passato due volte all’interno del metodo appena citato). Tale funzione riempirà la lista di stringhe con tutte le stringhe lette nei file, ed essa verrà poi passata a “stringList\_to\_finalString()”, ritornando tale valore, che rappresenta la stringa che per ogni posizione, avrà il carattere più frequente all’interno di tutte le stringhe lette dai file.