Ci abbiamo creduto e ci abbiamo provato. Ai posteri e al prof l’ardua sentenza.

Per realizzare la compressione di Snappy, algoritmo super veloce creato da Google ho tentato, partendo dall’alto, di scomporre il problema in piccoli compartimenti il più possibile indipendenti l’uno dall’altro concentrandomi sul mantenere il tutto il più semplice possibilie.

Per esempio: Snappy è un algoritmo che comprime a blocchi di massimo 64 kB. Questo permette di limitare fortemente la quantità di risorse allocate in memoria. Così mi sono concentrato unicamente su quest’aspetto di dividere il file originale in blocchi da comprimere separatamente e voilà.

* Scrivo dim varint. Questo permette al decompressore di conoscere quanto grande sarà il file una volta decompresso e quindi allocare la giusta quantità di risorse.
* …(leggo schema)
* Ad ogni iterazione le strutture dati (buffer e hash table) vengono svuotate

Da qui sono sceso di un livello e mi sono occupato della compressione del blocco vera e propria. Dopo molta sofferenza sono riuscito a riassumere tutto in questa singola funzione:

All’inizio di ogni blocco c’è sempre un literal, non avendo ancora una finestra di dati in cui cercare. Poi procedo fino alla fine del blocco cercando delle copie all’interno del blocco di dati. Per farlo, accedo all’hash table con l’hash code generato dai 4 byte correnti

Passando al flowchart della stessa funzione si capisce perfettamente che qui (punto all’if verde) sta la magia di Snappy. Accedendo all’hash table con l’hash code generato dai 4 byte correnti, posso determinare in tempo costante se ne esiste una copia o no.

**Heuristic**

Come vedete, un’altra caratteristica di Snappy è quella di non effettuare nessuna analisi per accertarsi che il file sia comprimibile. Come fare in modo quindi che il programma non perda tempo ad cercare una copia ad ogni byte. Con una soluzione un po’ sporca ma efficace. Semplicemente se analizzo 32 byte senza trovare una copia….

Blow up per file non comprimibili.

Da cui in poi si trattava semplicemente di risolvere i piccoli problemi come quello di scrivere correttamente le informazioni nel formato di Snappy o quello di aggiornare l’hash table.

A tal proposito: durante lo sviluppo mi sono trovato davanti al problema su come implementare l’hash table. Le due alternative principali erano….. Gestione collisioni. L’altra essendo una hash table aperta salva al suo interno semplicemente tutti i byte analizzati

Da un lato l’array di semplici permette una maggior velocità, dall’altro

Alla fine ho deciso di provare ad implementare entrambe e verificarne le prestazioni

Come ne faccio l’update

Test