L'algoritmo FAS, non può produrre soluzioni ottime nel caso in cui ad ogni attività sia associato un valore massimo, e bisogna selezionare un sottoinsieme di attività compatibili di valore massimo. ~~[illegible]~~ ~~[illegible]~~

ESERCIZIO 10 di GD

Utilizziamo il codice di Huffman per comprimere dati, normalmente utilizzando questo algoritmo possiamo portare a comprimere le dimensioni dei file fino al 50% della dimensione originale. Ci basiamo sul numero di frequenza delle lettere, per creare un ~~[illegible]~~ tabella di codice di lunghezza variabile, differente da quella utilizzata dall'ASCII, che usa blocchi di lunghezza fissa. Per ogni carattere dobbiamo calcolare la frequenza, cioè il numero di volte che si presenta all'interno del testo da comprimere.

Ecco l'algoritmo

Huffman $(C = \{c_1, c_2, \ldots, c_n\}, f(c_1), f(c_2), \ldots f(c_n))$

metti gli elementi di C in una coda Q

for $i \leftarrow 1$ to $|C|$ do

  alloca un nuovo nodo z

  left $[z] \leftarrow x \leftarrow$ EXTRACT_MIN (Q)

  right $[z] \leftarrow y \leftarrow$ EXTRACT_MIN (Q)

  $f[z] \leftarrow f[x] + f[y]$

  INSERT (Q, z)

La complessità dell'algoritmo Huffman (C, f) è $O(n \log n)$

Esiste sicuramente un albero ottimo T in cui due caratteri $x, y \in C$ di f minima appaiono nell'albero T alla profondità max, sono fratelli.

L'albero T' è ottenuto da T eliminando x e y rendendo il loro padre z una foglia con $f(z) = f(x) + f(y)$.

Definiamo $B(T) = \sum_{c \in C} f(c)\, d_T(c)$ e $B(T') = \sum_{c \in C'} f(c)\, d_{T'}(c)$