2.28 - Fusió ordenada de seqüències

Ja sabeu que fer la fusió ordenada de dues seqüències ordenades d'm i n elements, respectivament, comporta fer $m+n$ moviments de dades (penseu, per exemple, en un *merge* durant l'ordenació amb *mergesort* d'un vector). Però si hem de fer la fusió d'N seqüències, dos a dos, l'ordre en què es facin les fusions és rellevant. Imagineu que tenim tres seqüències A, B i C amb 30, 50 i 10 elements, respectivament. Si fusionem A amb B i després el resultat el fusionem amb C, farem $30+50=80$ moviments per a la primera fusió i $80+10=90$ per a la segona, amb un total de 170 moviments. En canvi, si fusionem primer A i C i el resultat el fusionem amb B farem un total de 130 moviments.

Dissenyeu un algorisme golafre (*greedy*) per fer les fusions i obtenir la seqüència final ordenada amb mínim nombre total de moviments. Justifiqueu la seva correctesa i calculeu-ne el cost temporal del vostre algorisme en funció del nombre de seqüències N.

Una solució

- L'algorisme *greedy* proposat utilitza la mateixa regla golafre que l'algoritme de Huffman: fusionar les dues llistes amb menor nombre d'elements. Després de fusionar les dues llistes l_1 i l_2, tenim una nova llista de mida $|l_1|+|l_2|$ amb els elements de les dues llistes ordenats. Per implementar aquest criteri farem servir una cua de prioritat Q on guardarem les llistes que encara s'han de fusionar (tal i com es fa a la implementació de l'algoritme de Huffman). La cua de prioritat ens permetrà accedir els elements en ordre creixent de mida.

  for all sequence s_i in $1 \leq i \leq N$ **do**
      Q.insert(s,$|s|$)
  while $|Q| \geq 2$ **do**
      $a \leftarrow$ Q.extract-min()
      $b \leftarrow$ Q.extract-min()
      $c \leftarrow$ Merge(a, b)
      Q.insert(c,$|c|$)
  return Q.extract-min()

- A cada iteració de l'algorisme, el nombre total de seqüències es redueix en una unitat; per tant el bucle farà $N-1$ iteracions. En total a la cua es faran $O(N)$ operacions d'extract-min i $O(N)$ operacions d'insert. Implementant la cua amb heaps el cost d'això serà $O(N \log N)$. Hem assumit que l'operació de merge té cost $O(1)$.

- Per tal de demostrar que l'algoritme greedy proporciona una solució amb el mínim nombre de moviments, observem que la solució calculada es pot representar (tal com passava a Huffman) com un arbre arrelat. Les fulles d'aquest arbre són les seqüències a fusionar i cada node intern representa una fusió entre dues seqüències.

  Observem que els elements d'una seqüència a participen a totes les fusions que es realitzen al camí des de la seva fulla fins a l'arrel de l'arbre. Si anomenem h_a l'alçada de la fulla de la seqüència a, aleshores el número de moviments que produeix a és $|a|h_a$.

  Suposem que, a una solució òptima, les dues seqüències a i b (amb $|a| \leq |b|$) amb menor nombre d'elements no són fulles contigües a la màxima profunditat possible ($h_{\max}$). Farem servir l'argument de l'intercanvi per arribar a una contradicció:

Considerem que les dues seqüències a màxima profunditat $h_{\max}$ són c i d (amb $|c| \leq |d|$). Si $c \neq a$ i intercanviem a amb c, tenim

$$|a|\ h_{\max} + |c|\ h_a \leq |a|\ h_a + |c|\ h_{\max},$$

perquè $|a| \leq |c|$ i $h_a \leq h_{\max}$. Després de fer l'intercanvi, la solució és òptima.

Si $b \neq d$, aleshores intercanviem a més b amb d i, pel mateix motiu, la solució és una solució òptima. Donat que sempre tenim una solució òptima en la què el mínim i el segon mínim es fusionen a màxima profunditat, l'algoritme és òptim.