Анализ на задача 5 - Чат

Задача „Чат” е пример за проблем, към който може да се подходи успешно по много различни начини. Разнообразието се поражда от факта, че задачата може да се раздели на две части – подреждане на думите и използване на макроси, които могат да се разглеждат като независими или като силно свързани.

Взети поотделно, подзадачите могат да бъдат решени много ефективно. Ако думите се разглеждат като върхове на граф, като всеки два върха са свързани посредством ребро с тежест, показваща колко изтривания (‘+’ или ‘-‘) са необходими, за да се сведе едната дума до другата, задачата за подреждането се свежда до прословутия *проблем за търговския пътник*[1]. Намирането на разстоянието между всеки две думи може да се направи лесно с динамично оптимиране за време от порядъка на $O(N^2 MaxWordLength^2)$, което за съжаление е твърде бавно за ограниченията в задачата. За да се постигне необходимата скорост, е нужно използването на хеширане или удачна структура като *Suffix Tree*[2]. Добър подход, когато не достига време, е всички низове да бъдат сортирани по азбучен ред. Така думи с общи префикси ще бъдат съседни в редицата и ще могат да бъдат казани доста икономично. Същият алгоритъм може да бъде приложен и като се сортират по азбучен ред обратните думи (прочетени отзад напред).

Втората подзадача – оптимално използване на макроси в даден низ – също може да бъде решавана доста ефективно, разглеждайки я като независима. Използването на динамично оптимиране и хеширане дава възможността низ с дължина L да бъде оптимално скъсен, използвайки M-те макроса, със сложност $O(LM)$. Възможно е съществуването и на допълнителна оптимизация, която този алгоритъм не използва. Това са ситуациите, когато можем да добавим и изтрием буква и да получим макрос. Например, ако имаме низ “`alabala`” и макрос ‘`A`’->“`alawww---bala`”. В такъв случай е добре макросите да бъдат нормирани, т.е. всеки макрос да се сведе до низа, който в действителност изписва. Тази оптимизация дава немалко предимство, защото именно този тип макроси заменят най-дълги поднизове, макар и всеки да може да бъде използван най-много по веднъж.

Оптималното решаване на двете части поотделно обаче не е най-доброто крайно решение. Причината за това е, че по-дълъг низ може да се окаже по-податлив на оптимизации с макроси и да доведе до по-добро крайно решение. В такъв случай, дори и подзадачите да се решават отделно, е добре, след решаването на първата, да се запазят няколко от най-добрите резултата, от които във втората фаза да бъде избран оптималния. Друг възможен подход е да се тръгне отзад напред – от макросите да бъдат построени фрази, които после активно да участват в построяването на резултата от първа фаза.

Оценяването беше извършено посредством 20 теста разделени в 4 групи от по 5: различни по размер балансирани тестове (с по 15 макроса); различни по размер тестове без макроси; различни по размер тестове с по 26 макроса; максимално тежки тестове.

Група	I					II					III					IV
Тест	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16-20
N	20	80	200	500	800	100	200	400	600	900	100	200	400	600	900	999
M	15	15	15	15	15	0	0	0	0	0	26	26	26	26	26	26
mLen	10	100	1000	5000	9000	100	500	2000	5000	9000	100	500	2000	5000	9000	9999

**mLen е максималната дължина на дума в тест.*

Линкове

[1] http://en.wikipedia.org/wiki/Traveling_salesman_problem
http://www.tsp.gatech.edu/
http://web.telia.com/~u85905224/tsp/TSP.htm

[2] http://en.wikipedia.org/wiki/Suffix_tree
http://mila.cs.technion.ac.il/~yona/suffix_tree/