Задача ВЕРИГА

Пояснение към решението

Построяваме граф с насочени дъги. Върховете на този граф са знаците, които се срещат в дадените низове и има дъга от знак c_1 към знак c_2, тогава и само тогава, когато съществува низ измежду дадените, който е от вида c_1c_2. В така построения граф може да има повече от една дъга от един връх към друг, както и дъги в различни посоки между два върха. Лесно се съобразява, че съществуването на търсената подредба на дадените низове е еквивалентно на задачата за съществуване на ойлеров цикъл в графа плюс изискването графът да е силно свързан, т.е., да можем да се придвижим от всеки връх до всеки друг, спазвайки посоката на дъгите.

Проверката дали графът е ойлеров се извършва като програмата преброява за всеки връх дали броят на влизащите дъги е равен на броя на излизащите. Построяването на графа се извършва от фунцията `create_graph()`. Графът се представя като масив от вектори

```
const int M=256;
vector<int> g[M];
```

Така векторите `g[0], g[1], …, g[M-1]` съдържат съседите, съответно на върховете `0, 1, …, M-1`.

Векторът `mark` служи за отбелязване кои знаци измежду знаците с номера от 0 до 255 са върхове на графа. Векторите `b1` и `b2`, служат за преброяване на влизащите и на излизащите дъги от върховете. Така за връх `c`, `b1[c]` дава броя на влизащите и `b2[c]` дава броя на излизащите. Проверката дали графът е ойлеров във функцията `create_graph()` става чрез

```
for (int i = 0; i < M; i++)
  if (b1[i] != b2[i]) {Euler=false; return;}
```

Проверката, дали графът е свързан се извършва чрез търсене в дълбочина, реализирано от функцията `void dfs(int u)`. Тя отбелязва посетените върхове, използвайки вектора `visit`. Графът не е свързан, когато има връх `i`, за който `mark[i]==true` и `visit[i]==false`.

Емил Келеведжиев