ЕСЕНЕН ТУРНИР ПО ИНФОРМАТИКА
Шумен, 10–12 ноември 2006 г.
Тема за група А (12 клас)

Задача А1. ИГРА

Даден е неориентиран граф с n върха и m ребра. Върховете са номерирани с числата от 1 до n. Дадени са и n пула, номерирани също с числата от 1 до n, като във всеки връх на графа е поставен по един пул. Разрешено е разместване на пуловете по един единствен начин: пул с номер 1 може да се размени с пул, който се намира в съседен връх на графа. Напишете програма **`move`**, която намира минималния брой ходове, за които пуловете могат да се разместят така, че всеки пул да се намира във връх със същия номер.

Вход: На първия ред на стандартния вход са дадени числата n и m. На следващите m реда са дадени по две числа, представляващи краищата на поредното ребро. На последния ред са дадени n числа, задаващи началното разположение на пуловете: i-тото число представлява номера на пула, намиращ се във връх i.

Изход: На стандартния изход да се изведе едно число – минималния брой ходове, с които може да се стигне до крайната позиция. Ако не е възможно да се достигне желаната позиция, да се изведе `-1`.

Ограничения: $n < 10$.

ПРИМЕР 1	**ПРИМЕР 2**
Вход	**Вход**

`3 2`
`1 3`
`2 3`
`3 1 2`
	`3 2`
	`1 2`
	`2 3`
	`3 1 2`

Изход	**Изход**
`2`	`-1`

Решение:

Да означим графа от условието на задачата с G. На всяко разположение на пуловете във върховете на G, съпоставяме пермутация $p = p_1, p_2, p_3, \ldots, p_n$, където p_i е номерът на пула, който се намира във връх i. Тъй като $n<10$, за представянето на различните пермутации ще използваме низове, например при n=5, низът `"23514"` означава, че във връх 1 се намира пул 2, във връх 2 – пул 3, във връх 3 – пул 5, във връх 4 – пул 1 и във връх 5 – пул 4.

Разглеждаме втори неориентиран граф P. Върховете на P са $n!$, като на всяка пермутация съответства връх от графа. Два върха са съседни, ако от едната пермутация може да се отиде в другата. Графът P е неориентиран, защото ходовете в играта са обратими. Например, ако в G върхове 2 и 4 са съседни, то от пермутацията `"23514"` може да се премине в `"21534"` и обратно.

Вместо да представяме графа P явно в програмата с някакви структури от данни, всеки път, когато трябва да намерим съседите на дадена пермутация-позиция извършваме следните пресмятания:

1) намираме върха u от графа G, в който се намира пул 1;

2) за всеки връх v, който е съсед на u (в графа G), получаваме съседна позиция в графа P, разменяйки пул 1 с пула от връх v.

При извършване на действията от точки 1) и 2), трябва да се съобразяваме с факта, че елементите на низовете са индексирани от 0.

В графа P трябва да намерим най-късия път (по брой ребра) от началната позиция до крайната позиция z="123…n". Ще използваме алгоритъма за търсене в ширина. За целта ни е необходима опашка, в която да бъдат включвани върховете (в нашия случай – позициите), които са достигнати за първи път.

Дефинираме и създаваме празна опашка по следния начин: `queue<string> Q;`

За всяка достигната позиция y, трябва да поддържаме броя ходове, с които може да бъде достигната. За целта ще използваме изображение (`map`), което на низове съпоставя числа, дефинирано така: `map<string,int> dist;`

Инициализираме стойностите на `dist` с `-1`, като за генерирането на всички пермутации използваме функцията от стандартната библиотека `next_permutation`.
Ако от текущата позиция x, може да се премине към съседна позиция y, за която `dist[y]` има стойност `-1`, то трябва да включим y в опашката: `Q.push(y)` и да установим `dist[y]=dist[x]+1`.

Броят на ребрата на графа P е равен на броя на ребрата на графа G. Следователно тялото на вътрешния цикъл в процедурата `bfs` се изпълнява общо m пъти. Едно отделно изпълнение зависи от реализацията на структурата `map`, която според спецификациите на стандартната библиотека осигурява ефективност $\log n!$. Така общо оценката за всички изпълнения на вътрешния цикъл е $O(m \log n!)$.

Всеки връх се посещава точно веднъж, което дава $O(n!)$ за операциите, свързани с опашката. Следователно за сложността на алгоритъма получаваме $O(n! + m \log n!)$ и като вземем пред вид, че $m \leq n(n-1)/2 \leq 36$, окончателно получаваме $O(n!)$ за сложността на алгоритъма по време. Очевидно и сложността по памет е $O(n!)$.

Стоян Капралов

Задача А2. СЛЪНЧОГЛЕДИ

Иванчо много обичал да ходи на гости на баба си Кунка. Тя имала къща с много дълъг балкон, на перваза на който били наредени саксии със слънчогледи. Противно на общоприетото схващане, те всъщност не гледали към слънцето. След като внимателно ги разгледал, Иванчо ограничил слънчогледите до четири типа – „северни“ (такива, които гледат на север), „южни“, „западни“ и „източни“. Установил още, че броят на северните слънчогледи е равен на броя на южните, както и, че броят на западните съвпада с броя на източните.

Баба Кунка размествала саксиите със слънчогледи всеки ден. При това не го правела безмозъчно, а с някаква странна логика и Иванчо скоро открил каква е тя: ако разгледаме всички саксии от някоя произволна позиция наляво, то броят на северните слънчогледи винаги е поне колкото на южните, както и броят на западните е поне колкото на източните. Изказано по-формално, горните правила звучат така:

n – брой саксии;
$SUM_N(i)$ – брой северни слънчогледи до саксия номер i ($1 <= i <= n$);
$SUM_S(i)$ – брой южни слънчогледи до саксия номер i ($1 <= i <= n$);
$SUM_W(i)$ – брой западни слънчогледи до саксия номер i ($1 <= i <= n$);
$SUM_E(i)$ – брой източни слънчогледи до саксия номер i ($1 <= i <= n$);

Изпълнено е, че: