Wstęp do programowania w języku C

Grupa MSz w czwartki

Lista 9 na zajęcia 15.12.2022

Zadanie 1. ***(15 punktów na pierwszej pracowni, 10 punktów na drugiej)***

Na wejściu dane są dodatnie liczby całkowite $w \leq 100$ (szerokość), $h \leq 100$ (wysokość) i $k \leq 100$ (ograniczenie liczby zmian kierunku). Nastepnie podanych jest h wierszy po w znaków, gdzie znak `'.'` określa pusty obszar, a znak `'#'` obszar niedostępny. Dokładnie dwa puste obszary są również oznaczone przez `'A'` i `'B'`; są to obszar początkowy i końcowy.

Napisz program który sprawdza czy można dojść od A do B zmieniając kierunek chodzenia co najwyżej k razy. Kierunki są cztery: lewo, prawo, góra, dół. Na początku nie mamy wybranego żadnego kierunku, więc wybranie go na początku liczy się już jako pierwsza jego zmiana.

- Przykład 1:

```
4 4 3
A...
##..
...#
...B
```

Odpowiedź: TAK.

- Przykład 2:

```
4 4 3
A.#B
#.#.
....
..#.
```

Odpowiedź: NIE (ale dla $k = 4$ byłoby TAK).

- Test wydajnościowy:

```
15 15 10
.#.#...#.#.#.#.
.....#......#..
.####.##.#.#.#.
........#.....#
.###.#.#.#B#.#.
.#........#.#..
.#.#.#.#.#.#.#.
...#...........
.#.#.#.#.#.#.#.
#.#.......#.#..
.#.#.#.#.#.#.#.
...............
.#.#.#.#.#.#.#.
...............
A#.#.#.#.#.#.#.
```

Na następnej stronie są wskazówki dla potrzebujących.

Ukryte testy: `https://drive.google.com/file/d/11LqYloS2x7V65BNC8y0EMg0BD3Lh1Vy3/`

Zadanie 2. *Przypominam, że do każdej listy w SKOSie jest jeszcze do zrobienia zadanie dla sprawdzaczki, które ma osobny termin.*

Wskazówki do zadania 1:

Wejście najłatwiej wczytać tak:

```
if (scanf("%u%u%u",&m,&n,&k) != 3) return 1;
for (unsigned int y = 0; y < n; y++)
  for (unsigned int x = 0; x < m; x++) {
    char c;
    if (scanf(" %c",&c) != 1) return 1;
    switch (c) {
    case '.': ...
    case '#': ...
    case 'A': ...
    case 'B': ...
    }
  }
```

(Spacja w formatowaniu `scanf` pomija wszystkie białe znaki, w szczególności koniec wiersza.)

Zadanie można rozwiązać używając DFSa lub BFSa. Można najpierw zrobić łatwiejszą wersję bez ograniczenia k.

W przypadku DFSa wygodna jest rekurencja. Na danym polu możemy podjąć decyzję aby podążać dalej prosto w wybranym wcześniej kierunku albo zmienić kierunek (o ile licznik zmian kierunku jest mniejszy od k). Aby program miał wielomianową złożoność nie możemy wielokrotnie odwiedzać tych samych miejsc mając ten sam kierunek i ten sam licznik zmian, trzeba więc zapamiętywać odwiedzone konfiguracje (tablica $n \times m \times k \times 4$).

BFS pozwala na trochę lepszą złożoność czasową i pamięciową – można zadbać o to, by przechodzić przez każdą parę pole $\times$ kierunek najwyżej raz, mając najmniejszy możliwy licznik zmian. Za to BFS wymaga kolejki, którą można zrobić wykorzystując odpowiednio dużą tablicę. Ale skoro już czytasz te wskazówki, to prawdopodobnie wolisz prostszy DFS.