Algorytmy dla Problemów Trudnych Obliczeniowo
Projekt 2025: Długi Kot

Problem

Długi Kot jest długi. Pewien kot właśnie obudził się na swoim legowisku i zorientował, że ktoś rozsypał po pokoju smaczki. Kot miałby ochotę zjeść ich jak najwięcej, ale wie, że jak tylko zejdzie z legowiska, to Zła Rumba zrobi z nim coś złego (do tej pory nie wiadomo, czy kot boi się o legowisko czy o siebie). Na szczęście długi kot jest długi i potrafi przejść po całym pokoju trzymając tylne łapki na legowisku. Utrudnienie stanowi fakt, że kot ma dużą inercję i jak już zacznie iść w jakimś kierunku, to idzie tak długo, aż natrafi na przeszkodę (co stanowi rzadką u kotów konsekwencję). Zadanie polega na zaplanowaniu trasy długiego kota tak, żeby zjadł co najmniej zadaną liczbę smaczków.

Wejście stanowi opis planszy (pokoju) o rozmiarach $W \times H$ pól (W to "szerokość", H to "wysokość" planszy). Na każdym polu znajduje się jeden znak oznaczający legowisko (takie pole jest tylko jedno i to na nim początkowo jest kot), wolne pole, nieprzesuwalną blokadę (można założyć, że blokady otaczają kota tak, że nie jest w stanie opuścić planszy) albo smaczek. W ramach jednego ruchu kot rozciąga się w danym kierunku (góra, dół, prawo lub lewo) na kolejne pola zajmując w ten sposób kolejne wolne pola tak długo, aż albo natrafi na blokadę, albo pole, które już sam wcześniej zajął. Pole z legowiskiem kota jest z definicji zajęte. Kot zjada smaczek, gdy wchodzi na pole, na którym ten smaczek się znajduje. Celem kota jest zjedzenie co najmniej S smaczków.

Wejście

Wejście składa się z jednego wiersza zawierającego po kolei liczby W, H, oraz S, po którym następuje ciąg H wierszy, gdzie każdy zawiera W znaków opisujących kolejne pola planszy, o następującym znaczeniu:

1. `"O"` – pole z legowiskiem (początkowa lokalizacja kota),
2. `"."` – wolne pole,

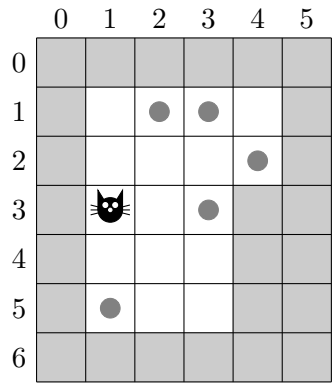
3. `"*"` – pole ze smaczkiem,

4. `"#"` – pole z blokadą (ściana, mebel itp.).

Przykład. Poniżej znajduje się opis przykładowej planszy:

```
6 7 4
######
#.**.#
#...*#
#O.*##
#...##
#*..##
######
```

Na tej planszy znajduje się 5 smaczków, z których kot chce zjeść co najmniej 4:



Wyjście

Wyjście składa się z pojedynczego wiersza zawierającego sekwencję ruchów kota. Każdy ruch opisany jest pojedynczą literą:

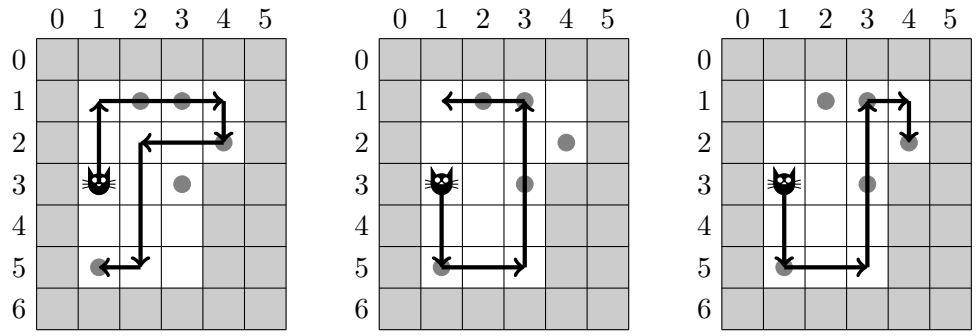
`"L"` – ruch w lewo, `"R"` – ruch w prawo, `"G"` – ruch w górę, `"D"` – ruch w dół.

Litery opisujące ruchy kota następują bezpośrednio po sobie, bez odstępów.

Przykład. Przykładowe rozwiązania powyższej planszy to:

(a) `GPDLDL`, (b) `DPGL`, (c) `DPGPD`.

Poniżej przedstawiamy wędrówkę kota dla każdego z tych rozwiązań:



(a) (b) (c)

Zadanie

Proszę zaimplementować program w języku C/C++, który wczytuje ze standardowego wejścia opis planszy oraz wypisuje na standardowe wyjście opis rozwiązania (można założyć, że rozwiązanie zawsze istnieje). Program powinien spełniać następujące warunki:

1. Program jest jednowątkowy i jednoprocesowy (nie wykorzystuje w żaden sposób mechanizmów wielowątkowości).
2. Program nie odwołuje się do żadnych operacji wejścia/wyjścia (w tym dostępu do sieci i plików) innych niż czytanie ze standardowego wejścia i zapisywanie na standardowe wyjście.

Punkty kontrolne (Checkpointy)

W ramach projektu odbędą się cztery punkty kontrolne. Przed końcem dnia każdego punktu kontrolnego należy wysłać program do systemu OIOIOI (link zostanie podany). Za każdym razem może to być inny program. Programy będą testowane na wejściach spełniających następujące warunki:

1. Checkpoint A (31 III 2025): Plansze o rozmiarach maksymalnie 8×8.
2. Checkpoint B (28 IV 2025): Plansze o rozmiarach maksymalnie 20×20, zawierające 1, 2, lub 3 smaczki (kot chce zjeść wszystkie z nich).
3. Checkpoint C (26 V 2025): Plansze o rozmiarach maksymalnie 20×20, gdzie każde wolne pole zawiera smaczek i kot chce zjeść wszystkie smaczki.
4. Checkpoint D (23 VI 2025): Plansze o rozmiarach maksymalnie 100×100.