Розв'язання задачі «Рахунок гри»

Автор: Ілля Порубльов

Основні ідеї кількома словами: найбільші області знаходяться за допомогою правила правої руки (у варіанті «з поверненнями»); площі обчислюються в процесі обходу по межі (додаванням орієнтованих (знакових) площ трапецій, бічні сторони котрих — ребро та його x-проекція).

1. Алгоритм виділення області

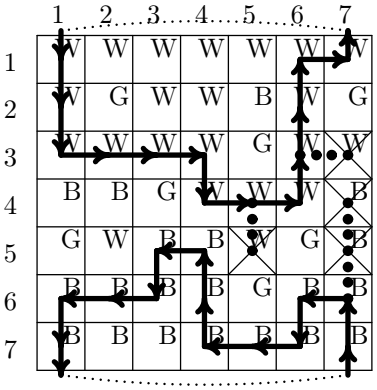

Рис. 1. Приклад з умови

Заради уникнення двозначностей, будемо користуватися словами «ліворуч / праворуч» *лише* у їхньому «відносному» розумінні (тобто, «ліворуч» означає «напрям, що знаходиться проти годинникової стрілки відносно поточного напрямку»). «Абсолютні» ж напрями будемо називати «західним», «східним», «північним» та «південним».

Припустимо, що межам приписаний напрям, вказаний на рис. 1. Тоді межа *характеризується* тим, що ліворуч від неї знаходиться (*уся*) внутрішність своєї області, праворуч — решта поля.

А це означає, що коли при побудові межі з'являється вибір (є різні сусідні клітинки потрібного кольору), то *в першу чергу* слід розглядати *найправіший* напрям. Адже правіше межі (рахуючи серед клітинок потрібного кольору) можуть бути тільки тупики (див. далі), котрі неважко розпізнати, відкинути й перейти до розгляду наступного (по порядку справа наліво) варіанта шляху. А от якщо піти «занадто ліворуч», можна отримати шлях, котрий насправді перетинає потрібну область (проходячи «строго всере́дині» неї), а помітити цей факт значно важче.

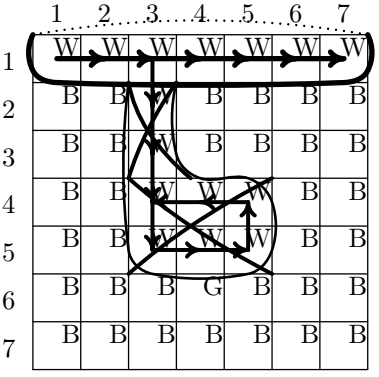

Рис. 2.

З'ясуємо, коли саме потрібно вважати, що потрапили в тупик, і «відступати» у попередню клітинку. Очевидно, тупиками будуть клітинки, у яких нема сусідів потрібного кольору (крім клітинки, звідки щойно прийшли). Але, як видно з рис. 2, вимога, щоб клітинки не повторювалися, порушується не лише при продовженні шляха через клітинку, де побували щойно, а і через клітинку, де взагалі коли-небудь побували.

Нарешті, якщо вже розглянули можливі продовження шляху потрібного кольору і вони *всі* виявилися тупиковими, то теж потрібно «відступати».

Тепер оглянемо основні ідеї *реалізації*.

Щоб не перевіряти, чи є у даної клітинки всі сусіди чи вона крайня, доцільно ввести в масив додаткові 0-ий та $(N+1)$-ий стовпчики, заповнивши їх зеленими клітинками.

Закодуємо «абсолютні» напрямки відповідно до рис. 3 (0 — «на південь», 1 — «на схід», 2 — «на північ», 3 — «на захід»). Крім того, введемо масиви-константи `dx:array[0..3] of integer = (0,1,0,-1)` та `dy:array[0..3] of integer = (1,0,-1,0)`.

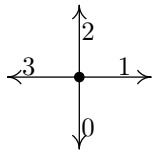

Рис. 3. Коди напрямків

Тоді отримувати зміни «абсолютних» координат взагалі легко (зміщення на одну клітинку в напрямі `d` означає збільшення `x`-координати на `dx[d]` та `y`-координати на `dy[d]`), та й з поворотами не складно: вираз `(d+1) mod 4` задає напрямок «ліворуч відносно `d`»; для поворота праворуч доводиться писати "`(d+3) mod 4`" (бо більш очевидне "`(d-1) mod 4`" не працює при `d` = 0); значить, перебрати напрямки у порядку справа наліво можна циклом

«`for dd:=3 to 5 do` дослідити напрям "`(d+dd) mod 4`"».

Попередні міркування щодо продовжень шляха та «відступань» найлегше реалізуються рекурсивним алгоритмом: перехід до наступної клітинки межі — рекурсивний виклик, «відступання» — вихід на один рівень назовні, при досягенння кінцевої[1] клітинки слід порахувати площу й вийти з рекурсії.

Але рекурсивна реалізація створює кілька незручностей.

1. Дійшовши до кінцевої клітинки, дуже бажано «зовсім вийти», а не вийти на кілька рівнів і зануритися в інші гілки рекурсії, безцільно перебираючи шляхи всере́дині області.
2. Глибина рекурсії визначається довжиною межі, котра в найгіршому випадку може сягати майже $\frac{2}{3}N^2$.
3. Для обчислення площі (за алгоритмом з рис. 6) потрібно знати саму межу як послідовність напрямків переходів, а рекурсивна реалізація (хоча й будує її неявно у програмному стеку) не дозволяє легко і просто доступатися до неї.

Усі ці незручності можна (при використанні 32-бітного компілятора) подолати, але цілком розумним буде інший варіант: «розгорнути» рекурсію в ітерацію, по аналогії з нерекурсивною версією алгоритма пошуку в глибину. Це буде не набагато складніше для написання, але надійніше і ефективніше за часом роботи програми.

[1] північно-східної при дослідженні білої області, південно-західної для чорної

2. Підрахунок кількостей клітинок у області

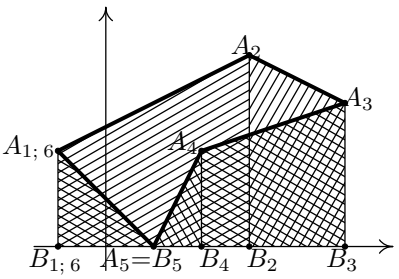

Рис. 4. До формули (1)

З обчислювальної геометрії відома формула площі простого[2] многокутника

$$S = \left| \sum_{i=1}^{n} \frac{(y_{i+1} + y_i) \cdot (x_{i+1} - x_i)}{2} \right|, \qquad (1)$$

де (x_i, y_i) — координати i-ої вершини многокутника в порядку обходу, n — кількість вершин, 1-а вершина умовно вважається також і $(n{+}1)$-ою.

Доведення формули опустимо, але відзначимо, що у прикладі з рис. 4 площі трапецій $A_1A_2B_2B_1$ та $A_2A_3B_3B_2$ додаються, трапецій $A_3A_4B_4B_3$, $A_4A_5B_5B_4$ та $A_5A_6B_6B_5$ — віднімаються, і в результаті виходить якраз те, що треба.

Звичайно, щоб застосувати (1) до даної задачі, формулу треба видозмінити. З одного боку, ситуація спрощується, бо тепер кожне ребро або вертикальне (тоді $x_{i+1} - x_i = 0$, отже доданок нульовий), або горизонтальне (тоді $\frac{y_{i+1}+y_i}{2}$ дорівнює поточній y-координаті).

Але ситуація ускладнюється тим, що номери клітинок — не зовсім координати. На рис. 5-(а) межа білої області виділена лінією, що проходить по центрам клітинок (тоді номери клітинок можна вважати координатами). На рис. 5-(б) ця межа паралельно перенесена на лінії між клітинками. В результаті частина клітинок межі опинилися ззовні області (на рис. 5-(б) виділені сірим кольором). Неважко переконатися, що такими будуть *в точності*:

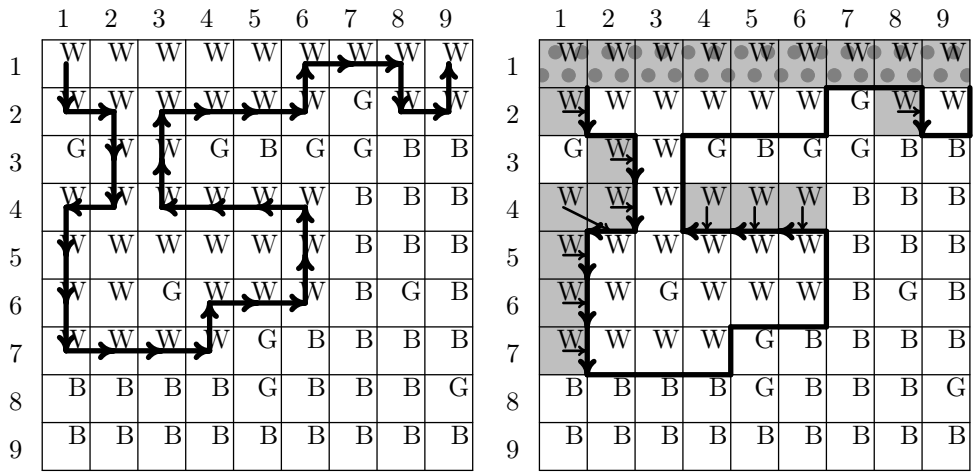

Рис. 5. Приклад складної межі

1. клітинки крайнього північного рядка, котрий за умовою має особливий статус;
2. клітинки, через котрі межа направлялася або на південь, або на захід.

Нам відома кількість ребер межі T; відомо також, що межа задає («сумарне») переміщення на $N{-}1$ клітинок на схід. Значить, десь у межі є $N{-}1$ ребер, направлених на схід, а решта $T - (N{-}1)$ дають нульове сумарне переміщення. А це означає, що сумарна кількість ребер, направлених або на південь, або на захід, дорівнює $\frac{T-(N-1)}{2}$.

```
res:=N+(T-(N-1)) div 2;  the_y:=0;
for i:=1 to T do begin
  the_y:=the_y+dy[BORDER_DIR[i]];
  res:=res + the_y*dx[BORDER_DIR[i]]
end;
```
Рис. 6. Алгоритм обчислення площі області (за відомою межею)

Остаточно, приходимо до алгоритма, наведеного на рис. 6. Масив `BORDER_DIR` зберігає напрями ребер межі згідно кодування з рис. 3. В алгоритмі (на відміну від формули (1)) нема «замикання» області, але це не впливає на відповідь, бо на початку та наприкінці межі `the_y` $= 0$.

3. Оцінки часу роботи та об'єму пам'яті

Кількість дій алгоритма виділення області становить $\theta(M_1) = O(N^2)$ (де M_1 — сумарна кількість клітинок у межі та тупикових клітинок); алгоритма обчислення площі області — $\theta(M) = O(N^2)$ (де M — кількість клітинок у межі). Втім, ці чудові оцінки перекриваються оцінкою $\theta(N^2)$ на читання вхідних даних.

Більш того, алгоритм виділення межі вимагає зберігати всі вхідні дані в масиві, тобто об'єм пам'яті теж становить $\theta(N^2)$. Як відомо, за таких умов «вузьким місцем» є об'єм пам'яті...

[2] можливо не опуклого і можливо не монотонного, але гарантовано несамоперетинного