Marian M. Kędzierski	Marian M. Kędzierski, Filip Wolski	Filip Wolski
Treść zadania	Opracowanie	Program

OI, Etap II, dzień drugi, 8.02.2008

Ucieczka

Al Bajtone, znany na całym świecie złodziej, planuje napad na bank. Wie, że jak tylko obrabuje bank, rozpocznie się policyjny pościg. Al Bajtone jest niestety kiepskim kierowcą i skręcanie w lewo sprawia mu kłopoty. Chce więc tak zaplanować ucieczkę, aby przez każde skrzyżowanie przejeżdżać na wprost lub skręcać w prawo. Wie też, że jeżeli przez jakieś skrzyżowanie raz przejedzie, to policja będzie tam już na niego czyhać. Nie może więc dwa razy przejeżdżać przez to samo skrzyżowanie. Dodatkowo, na niektórych skrzyżowaniach zawsze stoją patrole policji, więc takie skrzyżowania musi omijać (na skrzyżowaniach, przy których znajdują się bank i kryjówka, nie ma patroli policji).

Al Bajtone planuje trasę ucieczki z banku do swojej kryjówki. Nieoczekiwanie złożył Ci wizytę i przedstawił „propozycję nie do odrzucenia": musisz policzyć, ile jest różnych tras ucieczki prowadzących z banku do kryjówki, spełniających podane warunki.

Ulice Bajtogrodu tworzą regularną prostokątną siatkę. Wszystkie ulice prowadzą w kierunku północ-południe lub wschód-zachód, a na przecięciu każdych dwóch ulic znajduje się skrzyżowanie. Bank znajduje się na południe od skrzyżowania wysuniętego najbardziej na południowy zachód. Al Bajtone rozpocznie ucieczkę w kierunku północnym.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia położenie kryjówki, opis skrzyżowań, przy których stoją patrole policji, oraz pewną dodatnią liczbę całkowitą k,*
- *wyznaczy liczbę wszystkich tras ucieczki z banku do kryjówki, które spełniają podane warunki,*
- *wypisze na standardowe wyjście resztę z dzielenia wyniku przez k.*

Wejście

W pierwszym wierszu standardowego wejścia znajdują się trzy liczby całkowite n, m i k ($1 \leqslant n,m \leqslant 100$, $1 \leqslant k \leqslant 10^9$). Liczby n i m to odpowiednio liczby ulic, które prowadzą w kierunku wschód-zachód oraz północ-południe. W drugim wierszu znajdują się dwie liczby całkowite x i y ($1 \leqslant x \leqslant m$, $1 \leqslant y \leqslant n$). Reprezentują one położenie kryjówki — znajduje się ona przy skrzyżowaniu x-tej ulicy prowadzącej w kierunku północ-południe i y-tej ulicy prowadzącej w kierunku wschód-zachód. Ulice są numerowane z zachodu na wschód i z północy na południe, odpowiednio od 1 do m i od 1 do n.

W każdym z następnych n wierszy znajduje się m znaków „$*$" i/lub „$+$". Jest to mapa miasta. Znak w i-tym wierszu i j-tej kolumnie tej mapy określa skrzyżowanie i-tej ulicy prowadzącej z zachodu na wschód z j-tą ulicą prowadzącą z północy na południe, przy czym „$*$" oznacza, że na skrzyżowaniu stoi patrol policji, a „$+$" oznacza, że przez skrzyżowanie może prowadzić trasa ucieczki.

Al Bajtone rozpoczyna ucieczkę, wjeżdżając na skrzyżowanie o współrzędnych $(1,n)$ z kierunku południowego, tj. z nieistniejącego skrzyżowania $(1,n+1)$.

Wyjście

Twój program powinien wypisać w pierwszym i jedynym wierszu resztę z dzielenia liczby wszystkich możliwych tras ucieczki przez k.

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*

`3 5 10`
`4 2`
`+++++`
`++*++`
`++++*`
	`2`

Rozwiązanie

Wprowadzenie

Na II etapie II Olimpiady Informatycznej pojawiło się zadanie „Klub Prawoskrętnych Kierowców". W zadaniu tym należało stwierdzić dla wczytanej prostokątnej mapy pól (spośród których niektóre są zajęte, zaś inne wolne), jaka jest najkrótsza trasa z pewnego pola A do pewnego innego pola B, pod warunkiem, że nie dopuszczamy skrętów w lewo oraz jazdy przez zajęte pola.

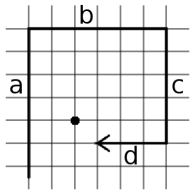
W naturalny sposób można zapytać o liczbę możliwych „prawoskrętnych" tras na takiej mapie. To pytanie stało się inspiracją do stworzenia tego zadania.

Rozwiązanie prawie wzorcowe — jeszcze nieoptymalne

Kluczowym warunkiem w zadaniu jest założenie, że zliczamy tylko trasy bez samoprzecięć. Sprawia to, że dozwolone są jedynie trasy szczególnej postaci.

Podzielmy drogę Ala Bajtone na części, z których każda jest prostym odcinkiem pomiędzy dwoma skrętami w prawo. Oznaczmy cztery pierwsze odcinki literami: *a*, *b*, *c* i *d* (gdy będzie nam wygodnie, to takimi samymi literami będziemy oznaczać także ulice, na których leżą odcinki). Przykład drogi z podziałem na odcinki jest przedstawiony na rys. 1.

Zauważmy, że skoro *a* zaczyna się w południowo-zachodnim końcu miasta, a *d* jest na pewno na południe od *b*, to przedłużenie odcinka *d* przecina odcinek *a*. W takim razie Al Bajtone musi zjechać z ulicy *d* (w prawo, oczywiście) przed dojechaniem do ulicy *a*. Oznaczmy kolejny odcinek trasy Ala Bajtone literą *e*. Łatwo zauważyć, że biegnie on w tym samym kierunku, co odcinek *a*, i całkowicie zawiera się w prostokącie ograniczonym ulicami *a*, *b*, *c* i *d*. Równie łatwo można dowieść metodą indukcji, że jeśli literami *u*, *v*, *w*, *x*, *y* i *z* oznaczymy sześć kolejnych (niekoniecznie początkowych) odcinków trasy Ala Bajtone, to:



Rys. 1: Początek trasy Ala Bajtone.

- ostatni odcinek z szóstki (z) jest w całości zawarty w prostokącie ograniczonym ulicami v, w, x i y;
- pierwszy odcinek szóstki (u) leży w całości poza prostokątem ograniczonym ulicami v, w, x i y.

Trasa ucieczki Ala Bajtone ma więc kształt spirali skręcającej w prawo. Ponadto ucieczka kończy się w kryjówce, stąd do powyższych trzeba dodać jeszcze jeden warunek:

- kryjówka musi należeć do każdego z prostokątów, utworzonych przez cztery kolejne odcinki ucieczki.

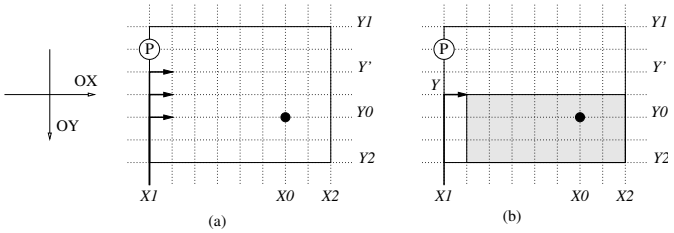
Z powyższych własności widać, że trasę ucieczki Ala Bajtone można przedstawić jako prosty odcinek zakończony skrętem i następnie trasę całkowicie zawartą w mniejszym prostokącie. To oznacza, że problem można spróbować rozwiązać techniką programowania dynamicznego — wyznaczając liczbę możliwych tras dla wszystkich prostokątów mapy zawierających kryjówkę Ala Bajtone. Prostokąty trzeba przy tym rozważać w takiej kolejności, aby wynik dla każdego prostokąta można było łatwo wyznaczyć na podstawie wyników dla prostokątów wcześniej przeanalizowanych.

Weźmy więc dowolny prostokąt $P = [x_1, x_2] \times [y_1, y_2]$, ograniczony ulicami x_1 i x_2 (pionowymi) oraz y_1 i y_2 (poziomymi). Załóżmy, że kryjówka Ala Bajtone leży na tym obszarze w punkcie (x_0, y_0), czyli $x_1 \leqslant x_0 \leqslant x_2$ i $y_1 \leqslant y_0 \leqslant y_2$. Przypomnijmy, że osie układu współrzędnych są zgodnie z treścią zadania skierowane odpowiednio na wschód i na południe. Chcemy obliczyć liczbę tras spełniających warunki zadania, wchodzących do P od południa w lewym dolnym rogu, czyli w punkcie (x_1, y_2).

Możliwe są dwa przypadki.

1. Kryjówka leży na ulicy x_1, czyli $x_0 = x_1$. Wtedy Al Bajtone może do niej dojechać tylko jedną drogą i to tylko wówczas, gdy na odcinku od skrzyżowania (x_1, y_2) do skrzyżowania (x_1, y_0) nie ma żadnych patroli policyjnych.

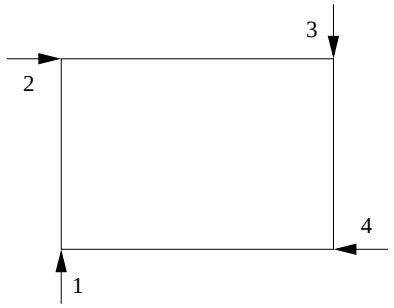
2. Kryjówka znajduje się na wschód od ulicy x_1, czyli $x_0 > x_1$. Wtedy Al Bajtone musi wykonać przynajmniej jeden skręt w prawo. Ponieważ zakładamy, że będzie poruszał się tylko w prostokącie P, to może skręcić na wschód w ulicę y dla $y_1 \leqslant y \leqslant y_2$. Dodatkowo $y' \leqslant y \leqslant y_0$, gdzie $y' - 1$ jest najdalej wysuniętym na południe skrzyżowaniem na ulicy x_1 pomiędzy ulicami y_1 i y_2, na którym stoi policja; jeżeli na odcinku od (x_1, y_1) do (x_1, y_2) nie ma policji, to przyjmujemy zamiast tego $y' = y_1$ (patrz też rys. 2a).



Rys. 2: Możliwości ruchu Ala Bajtone w prostokącie P (a) i podprostokąt P_y, w którym odbywa się dalsza ucieczka, po wyborze określonego y (b).

W pierwszym przypadku mamy rozwiązanie, równe zero lub jeden, gdy tylko sprawdzimy obecność patroli policyjnych na odcinku od (x_1, y_0) do (x_1, y_2). Poszukując rozwiązania w nietrywialnym, drugim przypadku, możemy przeglądać wartości $y = y_2, y_2 - 1, \ldots, y_1$, dopóki nie napotkamy pierwszego patrolu — na tej podstawie wyznaczamy y'. Dla wartości y z przedziału $[y', y_0]$ sumujemy wyniki dla prostokątów $P_y = [x_1 + 1, x_2] \times [y, y_2]$, zakładając, że wjeżdżamy do takiego prostokąta od strony zachodniej w lewym górnym rogu $(x_1 + 1, y)$ (patrz także rys. 2b).

W tym momencie widzimy, że informacja o liczbie tras wchodzących do prostokąta w lewym dolnym rogu od południa nie jest wystarczająca do obliczeń dynamicznych. Potrzebne są także wyniki w przypadku, gdy Al Bajtone wjeżdża przez lewy górny narożnik od zachodu. Powtarzając poprzednie rozumowanie dla takich przypadków, zauważymy szybko, że trzeba rozważyć także przypadki, gdy Al Bajtone wjeżdża w prostokątny obszar w prawym górnym rogu od północy i w prawym dolnym rogu od wschodu. Przypadki te oznaczymy: (1), (2), (3) i (4) — są one przedstawione na rys. 3.



Rys. 3: Konieczne do rozważenia sposoby rozpoczęcia trasy wewnątrz prostokąta.

Prosta implementacja przedstawionego pomysłu przebiega następująco. Rozpoczynamy od prostokąta $[x_0, x_0] \times [y_0, y_0]$ składającego się jedynie ze skrzyżowania, przy którym jest kryjówka — liczba tras we wszystkich czterech przypadkach jest równa 1. Następnie, dla kolejnych prostokątów o coraz większym polu (można je przeglądać, na przykład, zwiększając w każdym etapie jeden z wymiarów prostokąta o jeden) obliczamy każdy z czterech wyników. W pesymistycznym przypadku obliczenia dla jednego prostokąta wymagają przejrzenia wszystkich skrzyżowań znajdujących się na jego obwodzie.

Ponieważ na mapie mamy w sumie $O(n^2 \cdot m^2)$ prostokątów (dlaczego?) i dla każdego z nich należy rozważyć $O(n + m)$ potencjalnych miejsc skrętu, to złożoność czasowa

zaproponowanego rozwiązania wynosi $O(n^2m^2(n+m))$, jest więc zbyt duża jak na ograniczenia z zadania. Również konieczność pamiętania $O(n^2m^2)$ wyników dla wszystkich prostokątów przekracza dostępne zasoby pamięci.

Rozwiązanie wzorcowe — po poprawkach

Przedstawione rozwiązanie możemy jednak poprawić. Uważny Czytelnik dostrzeże bowiem, że nie ma konieczności sumowania $O(n+m)$ wyników dla mniejszych prostokątów, aby otrzymać wynik dla większego. Powróćmy do prostokąta $P=[x_1,x_2]\times[y_1,y_2]$ i obliczania dla niego wyniku w przypadku (1) — w poprzednim rozwiązaniu wymagało to dodawania wielu wyników typu (2) dla prostokątów postaci $[x_1+1,x_2]\times[y,y_2]$ dla $y' \leqslant y \leqslant y_0$. Zauważmy jednak, że wszystkie rozważane tu trasy możemy podzielić na dwie grupy:

- skręcające w prawo przed dojechaniem do ulicy y_1;
- skręcające w prawo na skrzyżowaniu (x_1,y_1).

Liczba tras z pierwszej grupy to w rzeczywistości liczba tras typu (1) dla prostokąta $[x_1,x_2]\times[y_1+1,y_2]$. Natomiast liczba tras z drugiej grupy to liczba tras typu (2) dla prostokąta $[x_1+1,x_2]\times[y_1,y_2]$, o ile tylko cały odcinek od (x_1,y_1) do (x_1,y_2) jest wolny od policji. Analogicznie można uprościć obliczenia we wszystkich czterech przypadkach — za każdym razem wynik dla prostokąta i jego wybranego rogu będzie wyznaczany w czasie stałym. Pozostaje więc tylko wymyślić metodę rozpoznawania w czasie stałym, czy dowolny odcinek jest wolny od policji, i otrzymamy rozwiązanie zadania działające w czasie $O(n^2m^2)$.

Wszystkie „bezpieczne" dla Ala Bajtone odcinki ulic możemy wyznaczyć na początku działania algorytmu, ponownie stosując programowanie dynamiczne. Wystarczy, że dla każdego punktu na mapie obliczymy, jak daleko na południe i na wschód ciągnie się bezpieczna strefa. Wyznaczając wartość (na przykład w kierunku wschodnim) dla skrzyżowania (x,y), wystarczy skorzystać z wartości dla skrzyżowania $(x+1,y)$. Łatwo zauważyć, że wszystkie wartości można wyliczyć w czasie $O(nm)$ i zapisać w pamięci rozmiaru $O(nm)$. Natomiast odpowiadając na pytanie, czy odcinek od (x_1,y_1) do (x_2,y_1), dla $x_1 \leqslant x_2$, jest bezpieczny, wystarczy porównać rozmiar bezpiecznej strefy w kierunku wschodnim dla skrzyżowania (x_1,y_1) z wartością x_2-x_1+1.

Uzyskaliśmy już satysfakcjonującą złożoność czasową. Niestety, złożoność pamięciowa całego algorytmu nadal jest rzędu $O(n^2m^2)$, co przekracza limit pamięciowy w zadaniu, który wynosi 64 MB. Problem stanowi tu duża liczba prostokątów, dla których zapisujemy wyniki częściowe. Oczywiście ostatecznie interesuje nas tylko wynik typu (1) dla prostokąta $[1,m]\times[1,n]$, ale wcześniej wyliczamy i tablicujemy wszystkie wartości typu (1)–(4) dla prostokątów $[x_1,x_2]\times[y_1,y_2]$, gdzie $1 \leqslant x_1 \leqslant x_2 \leqslant m$ i $1 \leqslant y_1 \leqslant y_2 \leqslant n$. Aby zredukować liczbę pamiętanych wartości, można obliczać wyniki dla prostokątów w takiej kolejności, by w każdym momencie korzystać tylko ze stosunkowo niewielu wyników wcześniejszych. Wówczas wszystkie wyniki starsze będzie można zapomnieć i do obliczeń wystarczy nam mniejsza pamięć. Znalezienie odpowiedniej kolejności przetwarzania prostokątów wymaga pewnej pomysłowości, gdyż nie jest ona najbardziej typowa.

Zauważmy, że obliczając liczbę tras (na przykład typu (1)) dla prostokąta $P=[x_1,x_2]\times[y_1,y_2]$, korzystamy z wyników dla prostokątów $[x_1,x_2]\times[y_1+1,y_2]$

i $[x_1+1, x_2] \times [y_1, y_2]$. Są to co prawda prostokąty o polach znacznie mniejszych niż badany, ale każdy z nich ma obwód mniejszy dokładnie o 2 od prostokąta P! Wystarczy więc pogrupować prostokąty ze względu na obwód i wyznaczać wyniki kolejno dla grup $G_0, G_2, G_4, \ldots, G_{2(n+m)}$, gdzie G_k to wszystkie prostokąty o obwodzie k. Widać, że obliczenia dla grupy G_k wymagają odwołania się jedynie do wartości z grupy G_{k-2} i każda grupa ma najwyżej $O((n+m)nm)$ elementów, co mieści się już w limicie pamięciowym dla zadania. Jako ćwiczenie pozostawiamy Czytelnikowi opracowanie efektywnej metody wyznaczenia i poindeksowania wszystkich prostokątów należących do ustalonej grupy G_k.

Opisane rozwiązanie wzorcowe zostało zaimplementowane w plikach `uci.cpp`, `uci1.pas` oraz `uci2.java`.

Rozwiązania nieoptymalne

Oprócz rozwiązania wzorcowego istnieją również algorytmy o gorszej złożoności.

Jednym z nich jest opisany jako pierwszy algorytm o złożoności czasowej $O(n^2m^2(n+m))$, który został zaimplementowany w plikach `ucib2.cpp`, `ucib3.pas` oraz `ucib4.java`. Przechodzi on testy 1–10.

Inne rozwiązanie to algorytm siłowy, czyli bezpośrednio wyszukujący wszystkie możliwe ścieżki ucieczki Ala Bajtone. Został on zaimplementowany w plikach `ucis3.cpp`, `ucis4.pas`, `ucis5.java`. Przechodzi on testy 1–7.

Testy

Rozwiązania zawodników były sprawdzane na następujących zestawach testowych:

Nazwa	n	m	Opis
uci1.in	5	5	mały test poprawnościowy z jednym posterunkiem policji
uci2a.in	5	5	mały test poprawnościowy z jedną trasą ucieczki
uci2b.in	5	5	mały test poprawnościowy z jedną trasą ucieczki
uci2c.in	5	5	mały test poprawnościowy bez trasy ucieczki
uci3a.in	10	10	średni test poprawnościowy — miasto bez policji
uci3b.in	10	10	średni test poprawnościowy — miasto z jednym patrolem policji
uci4.in	10	10	średni test poprawnościowy — kilka patroli, kryjówka blisko południowo-zachodniego krańca
uci5.in	10	10	średni test poprawnościowy — kryjówka blisko centrum
uci6.in	20	20	średni test poprawnościowo-wydajnościowy — dużo policji w mieście, kryjówka tuż obok banku
uci7a.in	20	20	średni test poprawnościowo-wydajnościowy
uci7b.in	20	10	średni test poprawnościowy — miasto puste

Nazwa	n	m	Opis
uci8.in	40	40	średni test poprawnościowo-wydajnościowy
uci9.in	40	40	średni test poprawnościowo-wydajnościowy
uci10.in	40	40	średni test poprawnościowo-wydajnościowy
uci11.in	80	80	duży test wydajnościowy
uci12.in	80	80	duży test wydajnościowy
uci13.in	100	100	duży test wydajnościowy
uci14.in	100	100	duży test wydajnościowy
uci15.in	100	100	duży test wydajnościowy
uci16.in	100	100	duży test wydajnościowy
uci17.in	100	100	maksymalny możliwy test, z maksymalną możliwą odpowiedzią

Zawody III stopnia

opracowania zadań