Jakub Wojtaszczyk
Treść zadania, Opracowanie

Wojciech Śmietanka
Program

Dostępna pamięć: 256 MB.

OI, Etap III, dzień pierwszy, 14.04.2010

Latarnia

Przy wejściu do budynku, w którym mieszka Bajtazar, Bitocy w środku nocy włączył latarnię. Światło latarni nie daje spać Bajtazarowi. Co prawda, latarnia nie świeci bezpośrednio w okno Bajtazara, ale jej światło odbija się od innych okien i wpada w okno Bajtazara. Bajtazar nie może zasnąć i denerwuje się. Żeby czymś zająć umysł, zamiast się denerwować, Bajtazar wygląda przez okno i zastanawia się, w które jeszcze okna świeci latarnia. Problem ten wciągnął go na tyle, że zapomniał o śnie i zadzwonił do Ciebie, prosząc o pomoc. Znasz Bajtazara nie od dziś i wiesz, że póki nie napiszesz dla niego programu rozwiązującego ten problem, sam(a) też nie pójdziesz spać.

Bajtazar mieszka w budynku B, w którym jest n okien. Latarnia znajduje się na samym dole budynku, na ścianie. Naprzeciwko budynku B stoi budynek C, oddalony o 10 metrów. Jego ściana jest równoległa do ściany budynku Bajtazara. Budynek C ma m okien.

Światło latarni rozchodzi się po liniach prostych, chyba że trafi w okno — wtedy mówimy, że **latarnia świeci w to okno***, a światło latarni odbija się od niego (zgodnie z zasadą „kąt padania równa się kątowi odbicia").*

Na ścianach budynków mamy określone układy współrzędnych — oś X jest pozioma, oś Y jest pionowa, osie na obu ścianach mają zgodne zwroty, a punkty $(0, 0)$ na obu ścianach leżą dokładnie naprzeciwko siebie. Okna to prostokąty położone na ścianach o bokach równoległych do osi układu współrzędnych. Światło nie odbija się od brzegów okien. Ponadto wiadomo, że okna w obrębie jednego budynku mają parami rozłączne wnętrza. Latarnia znajduje się na ścianie budynku B w punkcie $(0, 0)$ i nie znajduje się we wnętrzu ani na brzegu żadnego z okien.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite n oraz m ($1 \leqslant n, m \leqslant 600$), oddzielone pojedynczym odstępem, oznaczające liczbę okien w pierwszym i w drugim budynku. W kolejnych n wierszach opisane są okna w budynku Bajtazara (budynek B), po jednym w wierszu.

W $(i + 1)$-szym wierszu (dla $1 \leqslant i \leqslant n$) znajdują się cztery liczby całkowite $x_{1,i}$, $y_{1,i}$, $x_{2,i}$, $y_{2,i}$ ($-1\,000 \leqslant x_{1,i} < x_{2,i} \leqslant 1\,000$, $0 \leqslant y_{1,i} < y_{2,i} \leqslant 1\,000$), pooddzielane pojedynczymi odstępami. Taka czwórka oznacza, że i-te okno znajdujące się w budynku B stanowi prostokąt na ścianie budynku, którego lewy dolny i prawy górny róg mają współrzędne $(x_{1,i}, y_{1,i})$ i $(x_{2,i}, y_{2,i})$, wyrażone w metrach.

Następnie w kolejnych m wierszach, w tym samym formacie, opisane są okna budynku C.

Wyjście

Twój program powinien wypisać w pierwszym wierszu standardowego wyjścia liczbę okien budynku B, w które świeci latarnia. Możesz założyć, że dla każdych danych wejściowych będzie istniało co najmniej jedno takie okno.

W drugim wierszu należy wypisać numery tych okien, podane w kolejności rosnącej, poodzielane pojedynczymi odstępami (okna numerujemy od 1).

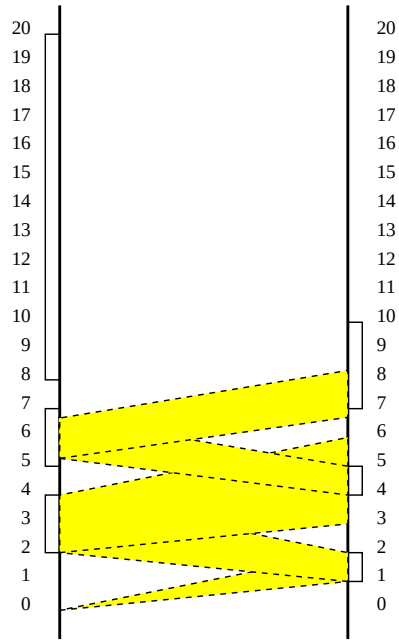
Przykład

Dla danych wejściowych:

```
3 3
-1 2 1 4
-1 5 1 7
-3 8 -2 20
-1 1 1 2
-1 4 1 5
-1 7 1 10
```

poprawnym wynikiem jest:

```
2
1 2
```



Wyjaśnienie do przykładu: *Do pierwszego okna budynku B światło dolatuje, odbiwszy się od pierwszego okna w budynku C (np. odbija się w punkcie $(0, 1.5)$ budynku C i trafia w punkt $(0, 3)$ budynku B). Żeby trafić w drugie okno budynku B, światło latarni musi odbić się trzy razy — ten sam promień trafia w punkt $(0, 4.5)$ w drugim oknie budynku C, a potem w $(0, 6)$ w drugim oknie budynku B. Światło nie trafia w trzecie okno budynku B, tylko obok niego. Należy dodać, że światło latarni oświetla cały budynek C, na rysunku natomiast zaznaczono tylko te promienie świetlne, które po odbiciu oświetlają okna budynku B.*

Rozwiązanie

Wprowadzenie

Powyższe zadanie jest dość trudne (czego dowód stanowi choćby fakt, że podczas zawodów nikt nie otrzymał za nie maksymalnej liczby punktów). Dlatego będziemy do rozwiązania przymierzać się krok po kroku, po drodze analizując i odrzucając różne możliwości.

Oznaczmy $M = \max(n, m)$, czyli M to maksymalna liczba okien na jednej ścianie budynku. Zaczniemy od zrozumienia, jak zachowuje się światło odbite od pojedynczego punktu na budynku C. Zauważmy, że jeśli światło latarni pada na budynek C w punkcie (x, y) zawartym w jakimś oknie, to następnie odbija się i pada na budynek B w punkcie $(2x, 2y)$. Jeśli tam trafi w okno, to odbija się dalej i trafia w budynek C w punkcie $(3x, 3y)$, i ogólnie, odbija się, trafiając w punkty (kx, ky), aż w końcu nie trafi w żadne okno. Aby uprościć nieco notację, dla dowolnego punktu $P = (x, y)$ oraz dowolnej liczby c przez cP oznaczymy punkt (cx, cy); podobnie, dla dowolnego prostokąta R o narożnikach A i B przez cR oznaczymy prostokąt o narożnikach cA i cB.

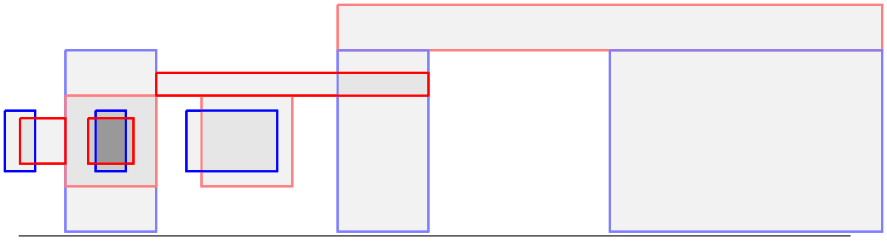
Widzimy teraz, że światło trafiające w okno opisane prostokątem O na budynku C po odbiciu oświetli na budynku B prostokąt $2O$. Jeśli teraz pewne okno na budynku B przecina się z prostokątem $2O$, dając prostokąt O', to po odbiciu od prostokąta O' światło oświetli na budynku C prostokąt $\frac{3}{2}O'$.

To oczywiście nasuwa nam pierwszy pomysł na rozwiązanie — dla kolejnych k pamiętamy prostokąty oświetlone na odpowiednim budynku po k odbiciach, przenosimy je na przeciwległy budynek, przecinamy z oknami, które tam są, dostajemy nowe prostokąty, i tak aż do skutku (tj. aż w końcu, po pewnej liczbie odbić, już nic nie będzie oświetlone). To rozwiązanie jednak zapewne będzie zbyt wolne — jeśli, przykładowo, na budynku B znajduje się 500 okien poziomych o wymiarach 2000×1, a na budynku C jest 600 okien pionowych o wymiarach 1×1000, to po dwóch odbiciach na ścianie budynku C oświetlamy już około 300 000 różnych prostokątów — a to dopiero dwa pierwsze odbicia! Teraz musielibyśmy przeciąć otrzymane prostokąty (po powiększeniu o czynnik 3/2) z 600 prostokątami na ścianie budynku C, i taką operację powtarzać tak długo, jak długo jakikolwiek promień będzie odbijał się od okien. Nawet jeśli uwierzymy, że liczba prostokątów nie będzie przyrastać ponad 300 000, to ciągle — wykonując przecięcia metodą „każdy z każdym" — dostajemy $180\,000\,000\,R$ operacji przecięcia prostokąta z prostokątem, przy czym R to maksymalna liczba odbić. Takie rozwiązanie było zatem zbyt wolne, by uzyskać na zawodach akceptowalną liczbę punktów.

Przy okazji nasuwa się pytanie: jak wiele razy może odbić się promień, zanim trafi w ścianę (nie w okno) budynku? (albo, innymi słowy, jak duże może być R z poprzedniego szacowania?) Otóż zauważmy, że aby odbijać się dalej, w drugim odbiciu musi trafić w okno na budynku B. Tymczasem każdy punkt znajdujący się wewnątrz okna na budynku B ma albo odciętą, albo rzędną różną od zera (bowiem w treści zadania znajduje się informacja, że latarnia nie leży na brzegu ani we wnętrzu żadnego okna, a współrzędne okien są całkowitoliczbowe). Zatem po dwóch odbiciach promień przesunął się przynajmniej o 1 wzdłuż którejś współrzędnej, czyli po $2k$ odbiciach przesunie się wzdłuż którejś współrzędnej przynajmniej o k. Stąd — skoro współrzędne okien są ograniczone co do wartości bezwzględnej przez 1000 — po co najwyżej 2001 odbiciach promień na pewno trafi w ścianę. Ta informacja będzie nam potrzebna do badania jakości konstruowanych rozwiązań.

Przecinanie prostokątów

Teraz zajmiemy się szybszym sprawdzaniem, które okna są oświetlone po k odbiciach.



Rys. 1: Ilustracja poniższego rozumowania dla testu przykładowego i $k = 4$. Najciemniejszym kolorem został zaznaczony obszar, od którego światło odbija się wciąż po czterech odbiciach — jest to prostokąt o narożnikach $(-1, 16/3)$ i $(1, 20/3)$.

Załóżmy, że chcemy wiedzieć, czy pewien punkt P jest oświetlony po k odbiciach. Jest tylko jeden promień, który po k odbiciach ma szansę trafić w ten punkt — to promień, który bezpośrednio na ścianę budynku C trafia w punkcie $\frac{1}{k+1}P$. Aby tam się odbił, w punkcie $\frac{1}{k+1}P$ na ścianie budynku C musi być okno. Dalej, aby promień nie zginął na ścianie budynku B po pierwszym odbiciu, w punkcie $\frac{2}{k+1}P$ na ścianie budynku B także musi być okno. I dalej, okna muszą być w punktach postaci $\frac{2r}{k+1}P$ na ścianie budynku B oraz w punktach postaci $\frac{2r+1}{k+1}P$ na ścianie budynku C.

Zauważmy, że można te wszystkie warunki przenieść na wspólną płaszczyznę. Zamiast mówić, że punkt αP leży na którymś z budynków w oknie O, można równoważnie powiedzieć, że punkt P leży w prostokącie $\frac{1}{\alpha}O$. Wykonajmy zatem następujący rysunek: dla każdej liczby całkowitej r, $1 \leqslant r < (k+1)/2$, oraz dla każdego okna O na budynku B rysujemy prostokąt $\frac{k+1}{2r}O$. Podobnie, dla każdego r', $0 \leqslant r' < k/2$, oraz każdego okna O' na budynku C rysujemy prostokąt $\frac{k+1}{2r'+1}O'$. Punkty, które są zawarte w którymś z prostokątów dla każdego r oraz dla każdego r', to dokładnie te punkty, które są oświetlone po k odbiciach (na odpowiednim budynku — w zależności od parzystości liczby k). Zauważmy jeszcze, że skoro okna na ścianie każdego z budynków były rozłączne, to prostokąty rysowane dla ustalonego r lub r' są rozłączne — a zatem zamiast sprawdzać, które punkty są zawarte w jakimś prostokącie dla każdego r i r', wystarczy sprawdzić, które punkty są zawarte w dokładnie k prostokątach. Dodajmy jeszcze, że chcemy tylko wiedzieć, które okna na budynku B będą oświetlone — czyli, w szczególności, wystarczy rozpatrywać problem tylko dla nieparzystych liczb k, gdyż dla parzystych k promień po k odbiciach trafia w budynek C.

Zamiatanie

Mamy zatem do rozwiązania następujący podproblem: dane jest k zbiorów prostokątów (każdy z tych zbiorów to przeskalowane, ze wspólną skalą, okna z któregoś z budynków). Każdy zbiór składa się z M parami rozłącznych prostokątów. Mamy dodatkowo M parami rozłącznych wyróżnionych prostokątów — to są okna na budynku B. Chcemy dowiedzieć się, które z tych M prostokątów przecinają się w jakimś punkcie z przynajmniej k innymi prostokątami. W naszym wypadku $M \leqslant 600$ oraz $k \leqslant 2000$.

Zauważmy, że jeśli znajdziemy pewien punkt, w którym przecina się dokładnie $k+1$ prostokątów, to każdy z nich pochodzi z innego zbioru, a zatem jest wśród nich wyróżniony prostokąt. Możemy zatem usunąć ten wyróżniony prostokąt ze zbioru (zaznaczając, że jest on częścią wyniku) i ponownie szukać punktu, w którym przecina się $k+1$ prostokątów. Jako że wyróżnione prostokąty nie przecinają się wzajemnie, usuwając już trafione prostokąty, nie psujemy wyniku dla tych, których jeszcze nie znaleźliśmy.

Chcemy zatem znaleźć punkt, w którym przecina się przynajmniej $k+1$ prostokątów. Do tego możemy użyć techniki znanej jako *zamiatanie*[1]. Wyobraźmy sobie prostą poziomą (będziemy ją nazywać *miotłą*) leżącą poniżej wszystkich prostokątów, którą następnie będziemy powoli przesuwać w górę. W każdym momencie chcemy dla każdego punktu miotły wiedzieć, ile prostokątów przecina się w tym punkcie. Za każdym razem, gdy w pewnym punkcie miotły przecina się $k+1$ prostokątów, znajdujemy wśród nich wyróżniony i usuwamy go (zmniejszając odpowiednie wartości na miotle oraz dodając usunięty prostokąt do rozwiązania), a następnie kontynuujemy zamiatanie. Technika zamiatania jest bardzo pożyteczna w wielu zadaniach geometrycznych, dlatego też dokładnie omówimy jej zastosowanie w tym zadaniu.

Prosta struktura danych

Przy zamiataniu zazwyczaj przechowujemy pewną strukturę danych, która opisuje obecny stan miotły (czyli, w naszym wypadku, ile prostokątów przecina który punkt miotły), oraz *zdarzenia*, czyli informacje o tym, w których momentach stan miotły będzie się zmieniał.

Wpierw poświęćmy chwilę uwagi strukturze danych. Przez *wartość* punktu będziemy rozumieć liczbę prostokątów, we wnętrzu których ten punkt jest zawarty. Oczywiście, skoro punktów na prostej jest nieskończenie wiele, nie chcemy przechowywać wartości wszystkich punktów. Zauważmy jednak, że wartość ta może zmieniać się tylko w punktach, których odcięta jest równa odciętej jakiegoś boku prostokąta — te punkty miotły nazwiemy *punktami kluczowymi*. Wystarczy zatem, dla ustalonego odcinka między dwoma sąsiednimi punktami kluczowymi, pamiętać wartość dowolnego punktu z tego odcinka, przykładowo środka (tu uwaga: w ogólnym przypadku należałoby jeszcze pamiętać wartości w samych punktach kluczowych, ale w tym konkretnym zadaniu przecinamy tylko wnętrza prostokątów, przez co w żadnym punkcie kluczowym nie przecina się $k+1$ prostokątów). Możemy, wobec tego, zacząć od posortowania wszystkich punktów kluczowych i wyznaczenia środków odcinków między sąsiednimi punktami — te środki odcinków nazwiemy *punktami znaczącymi*. Najprostszą strukturą danych, której możemy tu użyć, jest zwykła tablica, w której na i-tej pozycji trzymamy wartość i-tego punktu znaczącego. Ta struktura okaże się niedługo zbyt powolna, ale na chwilę obecną zależy nam na zrozumieniu samej idei zamiatania — gdy już skonstruujemy algorytm zamiatający, będziemy poprawiać tę strukturę.

[1] O zamiataniu można poczytać w książkach o geometrii obliczeniowej, np. [22], a także posłuchać na stronie `http://was.zaa.mimuw.edu.pl?q=node/37`. Technika ta była wykorzystywana w rozwiązaniach wielu zadań olimpijskich, jak np. *Straż pożarna* z XVI Olimpiady Informatycznej [16] lub *Trójkąty* z XV Olimpiady Informatycznej [15].

Zdarzenia i miotła

Na początku nasza struktura danych zawiera same zera — faktycznie, dla dostatecznie małych y (np. ujemnych) miotła nie przecina żadnych prostokątów. Sytuacja na miotle zmienia się dla wartości y, które są rzędnymi pewnego wierzchołka prostokąta. Jeżeli na wysokości y znajduje się dolny bok pewnego prostokąta o narożnikach (x_1, y_1) i (x_2, y_2), to wartości punktów znaczących zawartych na odcinku (x_1, x_2) zwiększamy o 1. Jeśli na wysokości y znajduje się górny bok, to wartości punktów na odcinku (x_1, x_2) zmniejszamy o 1.

Tworzymy zatem *zdarzenia* — czwórki (y, x_1, x_2, z), gdzie y to wysokość, na której zdarzenie ma miejsce, x_1 i x_2 to końce odcinka, którego zdarzenie dotyczy, zaś z to znak, czyli plus lub minus jedynka, w zależności od typu zdarzenia. Nasz algorytm zamiatający wygląda następująco:

```
1:  Algorytm miotły
2:  Wejście: k+1 zbiorów po M rozłącznych prostokątów każdy.
3:  Wyjście: w tablicy T na pozycji i znajduje się jedynka, jeśli i-ty wyróżniony
4:           prostokąt zawiera punkt, w którym przecina się k+1 prostokątów.
5:    for i := 1 to M do
6:      for j := 0 to k do begin
7:        kluczowe.wstaw(x1[i][j]);
8:        { x1[i][j] to odpowiednia współrzędna i-tego prostokąta z j-tego zbioru }
9:        kluczowe.wstaw(x2[i][j]);
10:     end
11:   kluczowe.sortuj();
12:   kluczowe.usunPowtorzenia();
13:   for x := 1 to rozmiar(kluczowe) - 1 do
14:     znaczace[x] := (kluczowe[x] + kluczowe[x + 1])/2;
15:   for x := 1 to rozmiar(znaczace) do wartosc[i] := 0;
16:   { Pole wartosc[i] będzie przechowywać wartość punktu znaczace[i]. }
17:   for i := 1 to M do
18:     for j := 0 to k do begin
19:       zdarzenia.wstaw(y1[i][j], x1[i][j], x2[i][j], +1);
20:       zdarzenia.wstaw(y2[i][j], x1[i][j], x2[i][j], −1);
21:     end
22:   zdarzenia.sortuj();  { leksykograficznie po współrzędnych }
23:   for j := 1 to rozmiar(zdarzenia) do begin
24:     for i := 1 to rozmiar(znaczace) do
25:       if (znaczace[i] > zdarzenia[j].x1) and
26:          (znaczace[i] < zdarzenia[j].x2) then
27:       begin
28:         if zdarzenia[j].z = +1 then wartosc[i] := wartosc[i] + 1
29:         else wartosc[i] := wartosc[i] − 1;
30:       end
31:     for i := 1 to rozmiar(znaczace) do
32:       if wartosc[i] = k + 1 then begin
33:         O := prostokąt wyróżniony zawierający punkt znaczace[i];
```

```
34:          T[O.numer] := 1;
35:          for z := 1 to rozmiar(znaczace) do
36:            if (O.x1 < znaczace[z]) and (znaczace[z] < O.x2) then
37:              wartosc[z] := wartosc[z] − 1;
38:          Usuń z listy zdarzeń zdarzenie związane z górnym brzegiem O;
39:        end
40:    end
41:    return T;
```

W powyższym algorytmie nie jest sprecyzowany sposób znajdowania wyróżnionego prostokąta zawierającego dany punkt oraz usuwania związanych z nim zdarzeń. To jednak nie jest istotnym problemem. Aby usuwać zdarzenia, wystarczy, przykładowo, dla każdego zdarzenia pamiętać, od którego prostokąta pochodzi, i przed obsłużeniem sprawdzać, czy ten prostokąt nie został już usunięty. Ze znajdowaniem prostokątów jest jeszcze prościej: otóż wystarczy nachalnie przejrzeć wszystkie, których jeszcze nie usunęliśmy, co dodaje do kosztu czasowego składnik $O(M^2)$.

Złożoność czasowa tego algorytmu to $O(k^2M^2)$ — dla każdego z $(k+1)M$ prostokątów przeglądamy listę wszystkich punktów znaczących, których jest $O(kM)$. Jest to zatem algorytm zbyt wolny. Widać, że winna jest tu mało wyrafinowana struktura danych reprezentująca miotłę. Poszukamy zatem czegoś lepszego.

Efektywna implementacja miotły: drzewo przedziałowe

Poszukiwania odpowiedniej struktury danych zacznijmy, metodycznie, od wypisania listy operacji, jakich wykonywanie musi ona umożliwiać. Załóżmy, że miotła zawiera N punktów znaczących, ponumerowanych od 1 do N. Wówczas każdemu punktowi znaczącemu i odpowiada parametr $wartosc[i]$. Musimy umieć obsłużyć pięć operacji:

(a) Ustaw wszystkie wartości na zero.

(b) Dodaj 1 do wszystkich wartości na przedziale $[a, b]$.

(c) Odejmij 1 na przedziale $[a, b]$, do którego wcześniej dodano 1.

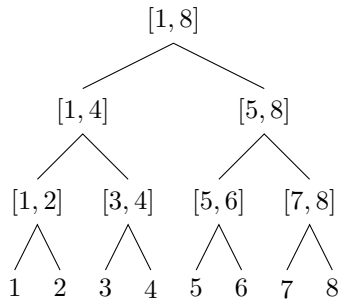
(d) Wyznacz największą wartość w miotle.

(e) Znajdź dowolne wystąpienie tej wartości.

Jeżeli będziemy umieli wykonać każdą z tych operacji w czasie $O(\log N)$, to będziemy w stanie zaimplementować algorytm miotły w czasie $O(kM \log(kM) + M^2)$. Faktycznie, wszystko poza powyższymi operacjami (w tym dwukrotne sortowanie) możemy wykonać w takim właśnie czasie, natomiast do wykonania operacji (b) i (c) potrzeba dodatkowo możliwości przeliczania współrzędnych punktów na miotle na numery najbliższych punktów znaczących, co można wykonywać w czasie $O(\log(kM))$ za pomocą wyszukiwania binarnego w tablicy *znaczace*.

Operacje (a)-(e) można zrealizować efektywnie, wykorzystując odpowiednio dostosowane *drzewo przedziałowe*[2]. Załóżmy, że N jest potęgą dwójki o całkowitym wykładniku (jeśli tak nie jest, to zwiększmy N tak, aby stało się potęgą dwójki). Wówczas *drzewo przedziałowe* to pełne drzewo binarne, którego liście odpowiadają liczbom od 1 do N, a węzły wewnętrzne reprezentują niektóre podprzedziały przedziału $[1, N]$, zwane *przedziałami bazowymi*, patrz rys. 2. W przedziałach interesują nas tylko liczby całkowite w nich zawarte. Przypomnijmy podstawowe własności tej struktury danych:

- Drzewo przedziałowe zawiera $O(N)$ węzłów, ma głębokość $O(\log N)$ i można je zaimplementować za pomocą jednej tablicy, tak aby dało się poruszać wzdłuż jego krawędzi w czasie stałym.
- Wszystkie przedziały bazowe zawierające dane $i \in \{1, 2, \ldots, N\}$ stanowią ścieżkę od liścia odpowiadającego i do korzenia.
- Każdy przedział $[a, b]$ można rozłożyć na sumę $O(\log N)$ parami rozłącznych przedziałów bazowych. Zarówno złożoność czasowa tego algorytmu, jak i liczba przedziałów bazowych będących przodkami przedziałów z otrzymanego rozkładu są rzędu $O(\log N)$.



Rys. 2: Przykład drzewa przedziałowego dla $N = 8$.

Obsługa operacji (a)-(c) stanowi jedno z najlepiej znanych, a zarazem najprostszych zastosowań drzew przedziałowych. Z każdym węzłem v drzewa wiążemy wartość $w[v]$, początkowo równą zeru. Liczby $w[v]$ mają spełniać tę własność, że dla każdego $i \in \{1, 2, \ldots, N\}$, $val[i]$ to suma wartości $w[v]$ po wszystkich przedziałach bazowych zawierających i (czyli na ścieżce od liścia odpowiadającego i do korzenia). Aby tak było, w operacjach (b) i (c) zwiększamy, odpowiednio o $+1$ lub -1, wartości $w[v]$ we wszystkich przedziałach bazowych z rozkładu przedziału $[a, b]$.

[2]W tym opracowaniu przedstawiamy jedynie skrócony opis drzew przedziałowych. Więcej na ich temat można przeczytać np. w opracowaniu zadania *Tetris 3D* z XIII Olimpiady Informatycznej [13] albo na stronie `http://was.zaa.mimuw.edu.pl/?q=node/8`. Sposób dostosowania (formalnie: wzbogacenia) drzew przedziałowych przedstawiony dalej jest podobny jak w przypadku problemu obliczania pola sumy (teoriomnogościowej) pewnej liczby prostokątów na płaszczyźnie, o którym to problemie można poczytać w książce [27] lub posłuchać na wspomnianej już wcześniej stronie `http://was.zaa.mimuw.edu.pl?q=node/37`. Wzbogacanie drzew przedziałowych pojawiło się także stosunkowo niedawno na Olimpiadzie: zadanie *Łyżwy*, XVI OI [16].

Obsłużenie operacji (d) oraz (e) wymaga dodatkowego *wzbogacenia* drzewa, czyli dodania we wszystkich węzłach jeszcze jednego, pomocniczego parametru, który oznaczymy przez W. Wartość $W[v]$ będzie reprezentować maksymalną wartość liścia w poddrzewie ukorzenionym w v, ale tak, jakby to poddrzewo było osobnym drzewem przedziałowym. Innymi słowy, jest to maksimum z sum wartości w na ścieżkach od v do wszystkich liści w poddrzewie v. Następujące równanie rekurencyjne:

$$W[v] = w[v] + \max(W[lewy\_syn(v)],\ W[prawy\_syn(v)]), \tag{1}$$

zachodzące dla wszystkich węzłów wewnętrznych v drzewa przedziałowego, pozwala w standardowy sposób aktualizować wartości W w drzewie po wykonaniu operacji (b) lub (c). Ów standardowy sposób polega na przejściu przez wszystkie węzły będące przodkami węzłów z rozkładu przedziału $[a, b]$, w kierunku od liści do korzenia, i zastosowaniu tego wzoru do obliczenia $W[v]$ we wszystkich węzłach v niebędących liśćmi. Koszt czasowy wciąż logarytmiczny.

Jeśli mamy do dyspozycji wartości parametru W, operację (d) obsługujemy w czasie stałym, zwracając tę wartość w korzeniu. Z operacją (e) także pójdzie łatwo: wystarczy przejść ścieżką od korzenia drzewa do liścia odpowiadającego wartości zwracanej w operacji (d). W tym celu, będąc w danym momencie w węźle v, przechodzimy do tego spośród jego synów, który realizuje maksimum we wzorze (1). Koszt czasowy takiego przejścia to, oczywiście, $O(\log N)$.

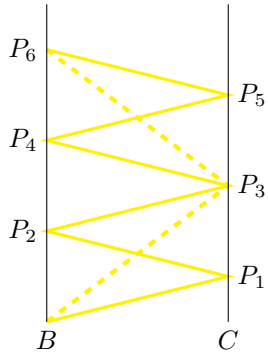
Wykończenie i algorytm wzorcowy

Omówiliśmy już technikę zamiatania wraz z drzewem przedziałowym. Umiemy w czasie $O(kM \log(kM) + M^2)$ odpowiedzieć na pytanie, które z okien na budynku B są oświetlone przez promienie światła odbijające się dokładnie k razy. Na tym etapie rozwiązywania zadania może wydawać się, że już nie musimy analizować naszego problemu, potrzebujemy tylko jakiejś technicznej sztuczki, aby osiągnąć akceptowalną złożoność. W końcu nie widać wyraźnych przeszkód, które uniemożliwiałyby sprytne połączenie, niezależnych w naszym podejściu, obliczeń dla różnych wartości k.

Istnieje jednak inne możliwe źródło ulepszeń. Fakt, że nie jest od razu widoczne, a samo jego dostrzeżenie stanowi tylko połowę sukcesu, stanowi o dużym stopniu trudności zadania. Spróbujmy cofnąć się na sam początek rozważań, do momentu, w którym myśleliśmy jeszcze o dwóch budynkach stojących naprzeciw. W ten sposób możemy dostrzec kluczowy fakt:

Obserwacja 1. Jeżeli promień z latarni może pokonać trasę do pewnego okna w $2^l(2m+1)-1$ odbiciach, to może pokonać tę samą trasę w $2^l - 1$ odbiciach.

Zaiste — rozważmy trasę, którą pokonuje promień, wykonując $2^l(2m+1)-1$ odbić, i przez $P_1, P_2, \ldots, P_{2^l(2m+1)-1}$ oznaczmy kolejne punkty odbić, a przez $P_{2^l(2m+1)}$ — docelowy punkt naszego promienia. Rozważmy teraz promień, który po raz pierwszy trafia w budynek C w punkcie P_{2m+1}. Jego kolejne odbicia wystąpią w punktach $P_{2(2m+1)}$, $P_{3(2m+1)}$, aż do $P_{2^l(2m+1)}$. Mamy gwarancję, że wszystkie punkty odbić są we wnętrzach okien, gdyż są to punkty, w których odbijał się oryginalny promień



Rys. 3: Pięciokrotne odbicie można zastąpić jednokrotnym.

(łatwo sprawdzić, że występują na tych samych budynkach). Zatem faktycznie nasz promień dotarł do celu w $2^l - 1$ odbiciach.

Warto tu zwrócić uwagę, że ten argument nie pozwala zejść z $2^l - 1$ odbić do $2^{l-1} - 1$ odbić — parzyste odbicia zawsze występują na ścianie budynku B, i nie gwarantują tego, że symetryczne punkty na budynku C znajdują się wewnątrz okien.

Co daje nam ta obserwacja? Otóż, jeśli jakieś okno będzie oświetlone, to będzie oświetlone w $2^l - 1$ odbiciach dla pewnego l. Wobec tego możemy rozważać istotnie mniej wartości parametru k — zamiast wszystkich liczb nieparzystych z przedziału $[1, 2000]$, wystarczy rozważać liczby postaci $2^l - 1$. Niech L będzie największą liczbą, dla której $2^L \leqslant R$ (w naszym przypadku $L = 10$, ogólnie $L = O(\log R)$). Stosując omówioną już metodę zamiatania, otrzymujemy algorytm o złożoności czasowej

$$O\left(M^2 + \sum_{l=1}^{L}\left(2^l M \log\left(2^l M\right)\right)\right) = O\left(M^2 + M\log\left(2^L M\right)\sum_{l=1}^{L} 2^l\right) =$$
$$= O\left(M\log\left(2^L M\right)2^{L+1} + M^2\right) = O\left(2^L M(L + \log M) + M^2\right).$$

W powyższych przekształceniach wartość M^2 umieściliśmy poza obliczaną sumą, gdyż odpowiada ona identyfikacji okna zawierającego dany punkt znaczący, które wykreślamy wówczas z dalszych rozważań, uznając je za oświetlone. Dla $L = 10$ i $M \leqslant 600$ takie rozwiązanie powinno bez większych problemów zmieścić się w limicie czasowym.

To podejście zostało zaimplementowane w plikach `lat.cpp` i `lat2.pas`.

Inne rozwiązania

Rozwiązania niepoprawne

W pliku `latb1.cpp` zaimplementowano rozwiązanie takie jak wzorcowe, które ogranicza się do liczby odbić nie większej niż 300. Taki program nie dostanie więcej niż 60 pkt. — jest możliwe, że do oświetlenia pewnego okna będą konieczne 1023 odbicia.

W pliku `latb2.cpp` jest rozwiązanie działające niemalże jak program `lat.cpp`. Jedyna różnica polega na obsłudze zdarzenia, w którym przychodzi przedział $[a, b]$

powodujący, że w miotle jest obszar pokryty przez $k+1$ prostokątów. W tym podejściu usuwamy wszystkie wyróżnione prostokąty, które przecinają się niepusto z przedziałem $[a, b]$. Oczywiście, jest to rozwiązanie niepoprawne.

W pliku `latb3.cpp` znajduje się rozwiązanie, które rozważa kilkaset losowych punktów z wnętrza każdego z okien i sprawdza, w które z nich może trafić promień światła. W tym celu przeglądane są kolejne liczby odbić postaci $2^l - 1$. Przy ustalonej liczbie odbić żądane sprawdzenie można wykonać w czasie $O(2^l)$. Wynika to z faktu, że po odpowiednich obliczeniach wstępnych, sprawdzenie, czy punkt (x, y) leży w oknie na danej ścianie, można wykonać w czasie stałym.

W pliku `latb4.cpp` znajduje się rozwiązanie, które działa prawie tak samo jak poprzednie. Różnica polega na tym, że nie korzystamy tutaj z Obserwacji 1 i sprawdzamy kilka losowych punktów dla kolejnych liczb odbić od 2 aż do 1000.

Rozwiązanie nieoptymalne

Po każdym odbiciu utrzymujemy oświetlony obszar, jako zbiór parami rozłącznych prostokątów. Po początkowych odbiciach zbiór prostokątów ma rozmiar rzędu $O(nm)$, co przy próbie przecinania każdego prostokąta z naszego zbioru z każdym napotkanym oknem daje koszt symulacji jednego odbicia $O(nm(n+m))$. Takie rozwiązanie zostało zaimplementowane w pliku `lats1.cpp`. W zależności od jakości implementacji, za tego typu rozwiązania można było uzyskać od 20 do 40 punktów.

Testy

Testy zostały podzielone na pięć grup, w każdej występuje kilka różnych testów.

Nazwa	n + m	Opis
lat1[a-d].in	$[41, 110]$	testy poprawnościowe
lat2[a-h].in	$[4, 320]$	testy poprawnościowe, dużo odbić
lat3[a-h].in	$[2, 580]$	testy poprawnościowe, dużo odbić
lat4[a-g].in	$[630, 960]$	testy wydajnościowe, dużo odbić
lat5[a-h].in	$[1\,050, 1\,200]$	testy wydajnościowe, dużo odbić