Karol Pokorski
Treść zadania, Opracowanie

Lech Duraj
Program

Dostępna pamięć: 128 MB.

OI, etap III, dzień drugi, 14.03.2013

Laser

Kapitan Bajtazar poluje na odcinkowce na planecie Tumulum VI. Jego statek stoi w miejscu lądowania (które będziemy uważać za początek układu współrzędnych) i wyposażony jest w laser ogłuszający. Laserem tym można obracać, tak aby promień lasera był skierowany pod dowolnym kątem. Zasilania statku wystarczy na co najwyżej k strzałów, z których każdy może być oddany pod dowolnie wybranym kątem. W momencie, gdy laser jest włączony, nie można nim obracać.

Na planecie znajduje się n odcinkowców – każdy z nich jest istotą jednowymiarową (odcinkiem) o końcach w punktach o współrzędnych dodatnich i całkowitych. Celem Bajtazara jest trafienie promieniem lasera jak największej liczby odcinkowców, przy czym zabronione jest trafienie jakiegoś odcinkowca więcej niż raz – kapitan chce je sprzedać z dobrym zyskiem, a do tego muszą być w nienagannym stanie fizycznym i psychicznym. Promień lasera rozchodzi się wzdłuż prostej, a gdy trafia odcinek, przenika przez niego i biegnie dalej. Jeśli promień lasera przejdzie przez sam koniec lub wzdłuż odcinkowca, to również jest on trafiony.

Napisz program, który wyznaczy maksymalną liczbę odcinkowców, które można trafić promieniem lasera, zgodnie z podanymi powyżej zasadami.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite k i n ($1 \leqslant k \leqslant 100$, $1 \leqslant n \leqslant 500\,000$), oddzielone pojedynczym odstępem. W kolejnych n wierszach opisane są odcinkowce, po jednym w wierszu. W każdym z tych wierszy znajdują się po cztery dodatnie liczby całkowite x_1, y_1, x_2, y_2 ($1 \leqslant x_1, y_1, x_2, y_2 \leqslant 1\,000\,000$), rozdzielone pojedynczymi odstępami. Liczby takie reprezentują odcinkowca o końcach (x_1, y_1) i (x_2, y_2).

W testach wartych 36% punktów zachodzi dodatkowy warunek $k \leqslant 2$, w testach wartych 45% punktów zachodzi $n \leqslant 2000$ i $k \leqslant 30$, natomiast w testach wartych 81% punktów zachodzi $n \leqslant 200\,000$ i $k \leqslant 50$.

Wyjście

Twój program powinien wypisać w pierwszym (i jedynym) wierszu standardowego wyjścia dokładnie jedną liczbę całkowitą: maksymalną liczbę odcinkowców, które można trafić promieniem lasera (każdego dokładnie raz), wykonując co najwyżej k strzałów.

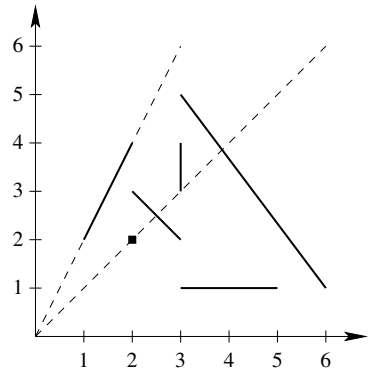
Przykład

Dla danych wejściowych:

```
3 6
1 2 2 4
3 1 5 1
3 2 2 3
3 3 3 4
2 2 2 2
6 1 3 5
```

poprawnym wynikiem jest:

```
5
```



Wyjaśnienie do przykładu: *Wystarczy uruchomić laser dwukrotnie. Promień lasera zaznaczono na rysunku linią przerywaną.*

Testy „ocen”:

`1ocen`: $k = 4$, $n = 5$, *mały test poprawnościowy;*

`2ocen`: $k = 2$, $n = 5$, *tylko odcinkowce zerowej długości;*

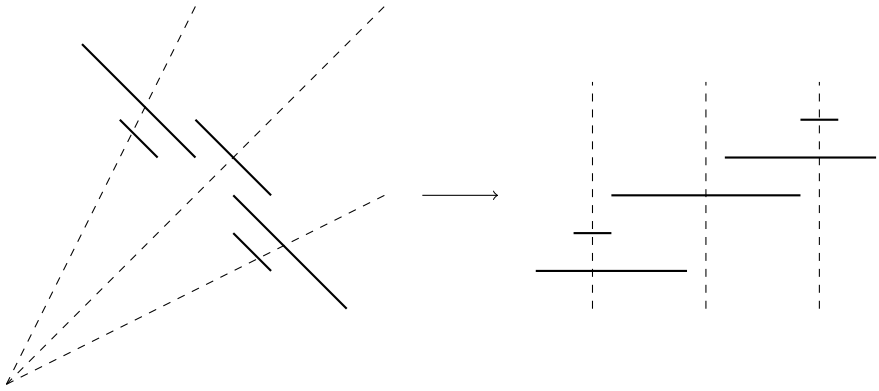
`3ocen`: $k = 2$, $n = 3$, *złośliwy test poprawnościowy;*

`4ocen`: $k = 3$, $n = 500\,000$, *test maksymalnego rozmiaru.*

Rozwiązanie

Niezbyt długą historyjkę z zadania *Laser* można jeszcze skrócić i powiedzieć, że sprowadza się ono do wyznaczenia k półprostych (zaczepionych w początku układu współrzędnych) w taki sposób, aby przecięły łącznie jak największą liczbę zadanych odcinków, przy czym żadnego odcinka nie wolno przeciąć dwa razy.

Geometryczna otoczka tego zadania może trochę odstraszać, szczęśliwie jednak okazuje się, że stanowi ona jedynie szczegół implementacyjny. Łatwo przekonać się, że nie potrzebujemy znać dokładnych współrzędnych końców odcinka, a jedynie kąty, pod jakimi należy ustawić laser, aby jego promień przechodził przez te końce. Wówczas każdy odcinek możemy utożsamić z przedziałem kąta lasera, który trafia w ów odcinek (rys. 1). Dzięki tej obserwacji do rozwiązania mamy nieco łatwiejsze zadanie: *Danych jest n odcinków na prostej, które należy przeciąć co najwyżej k pionowymi liniami, aby jak najwięcej odcinków zostało przeciętych, a żaden nie został przecięty dwukrotnie.* Zauważmy, że korzystamy tutaj z faktu, że wszystkie odcinki są zawarte w jednej ćwiartce układu współrzędnych. Od tego momentu możemy zapomnieć o oryginalnym zadaniu i skupić się na jego nowej wersji.



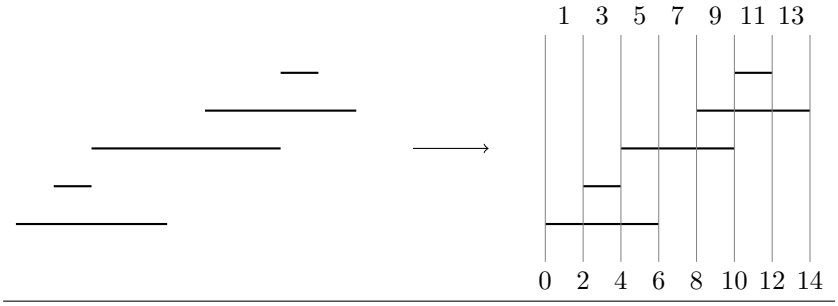
Rys. 1: Sprowadzenie problemu dwuwymiarowego do jednowymiarowego.

Rozwiązanie wzorcowe

Ponieważ kolejność wykonywania strzałów nie ma znaczenia, będziemy je rozpatrywać w kolejności rosnących współrzędnych x. To pozwoli nam wprowadzić do problemu przestrzeń stanów i myśleć w kategoriach programowania dynamicznego. Niech $r[p, x]$ oznacza maksymalną liczbę odcinków, które można przeciąć (nie przecinając żadnego dwukrotnie), wykorzystując dokładnie p strzałów, jeśli ostatni strzał nastąpił nie później niż na współrzędnej x.

Przy takiej definicji stanów jest nieskończenie wiele. Zauważmy jednak, że jeśli między współrzędnymi x_1 i x_2 nie zaczyna się ani nie kończy żaden odcinek – można strzelić w dowolną z nich, a wynik pozostanie niezmieniony. Takie współrzędne można więc bezpiecznie utożsamić (a nawet przeskalować tak, aby były, na przykład, niedużymi liczbami całkowitymi; patrz rys. 2).

Co ważne, nie można sobie pozwolić tylko na analizę punktów pomiędzy krańcami odcinków (na rysunku: punktów o współrzędnych nieparzystych), gdyż w jednej współrzędnej x mogą jednocześnie zaczynać i kończyć się różne odcinki (na rysunku



Rys. 2: Przeskalowanie współrzędnych. Współrzędne parzyste odpowiadają początkom i końcom odcinków, a współrzędne nieparzyste – maksymalnym przedziałom, które nie zawierają krańców odcinków.

dzieje się tak np. dla $x = 10$). Z tego samego powodu oraz ze względu na konieczność co najwyżej jednokrotnego trafienia każdego odcinka, nie wystarczy także analizowanie tylko początków i końców odcinków (na rysunku: punktów o współrzędnych parzystych). Można to także zilustrować na naszym prostym przykładzie: gdyby chcieć wykonać dokładnie trzy strzały, możliwe jest trafienie wszystkich odcinków z powyższego rysunku – jednak aby tego dokonać, niezbędne jest dokonanie strzału w $x = 7$.

Od tego momentu rozważana przestrzeń stanów jest skończona, a nawet dość mała – zawiera bowiem $\Theta(nk)$ stanów. Pozostaje nadal pytanie, jak obliczać wartości $r[p, x]$. Załóżmy, że znane są już wyniki dla wszystkich stanów dla $p-1$ strzałów oraz wyniki dla stanów $r[p, x']$, dla $x' < x$. Wówczas rozpatrujemy dwa przypadki, w zależności od tego, czy wykonujemy strzał we współrzędnej x.

Jeśli nie wykonujemy strzału w x, to wszystkie p strzałów musimy wykonać (zgodnie z zasadami zadania) przed współrzędną x, zatem będzie ich $r[p, x-1]$.

W drugim przypadku sytuacja jest bardziej skomplikowana. Musimy zadbać o to, aby żaden z odcinków przeciętych w strzałach wcześniejszych nie został przecięty w p-tym strzale. Biorąc pod uwagę kolejność obliczeń, wystarczy zapewnić, aby zbiory odcinków przeciętych w $(p-1)$-szym i p-tym strzale były rozłączne. Zatem współrzędna $(p-1)$-szego strzału musi być mniejsza od współrzędnych wszystkich początków odcinków zawierających współrzędną x.

Aby móc zaimplementować wydajne obliczanie wyników dla poszczególnych stanów, warto najpierw wyznaczyć kilka pomocniczych wartości:

- $w[x]$ – liczba odcinków, które trafia strzał wykonany w x,
- $l[x]$ – najmniejsza współrzędna początku odcinka trafionego strzałem w x.

W drugim przypadku zestrzelimy więc $r[p, l[x]-1] + w[x]$ odcinków. Ostatecznie rekurencja (pomijając warunki brzegowe) ma postać:

$$r[p, x] = \max(r[p, x-1], r[p, l[x]-1] + w[x]).$$

Wartości $w[x]$ można obliczyć, rozbijając każdy odcinek na dwa zdarzenia: początek (który zwiększa liczbę przeciętych w danym miejscu odcinków o jeden) oraz koniec odcinka (który tę wartość zmniejsza). Takie zdarzenia można posortować względem współrzędnej x, remisy rozstrzygając na korzyść punktów rozpoczynających odcinki. Następnie wystarczy już tylko rozpatrzyć te zdarzenia zgodnie z nowym posortowaniem i zapisać dla kolejnych x liczbę aktualnie przeciętych odcinków.

Wartości $l[x]$ można obliczyć analogicznie, najłatwiej będzie jednak analizować zdarzenia względem malejących wartości x. Utrzymujemy minimalną współrzędną dla początków odcinków przecinających x i, jeśli napotkamy koniec nowego odcinka, sprawdzamy, czy jego początek jest wcześniej niż aktualnie znane minimum.

W ten sposób dostajemy rozwiązanie o złożoności obliczeniowej $O(nk)$ i takiej samej złożoności pamięciowej. Tę drugą można łatwo zredukować, zauważając, że aby obliczyć wyniki dla p strzałów, wystarczy pamiętać tylko wyniki dla $p-1$ strzałów. Zastosowanie tej optymalizacji pozwala zredukować złożoność pamięciową do $O(n+k)$. Wyżej opisane rozwiązanie zostało zaimplementowane w plikach `las.cpp`, `las1.pas` i `las2.cpp` i jest wystarczające do uzyskania maksymalnej punktacji. Rozwiązania bez optymalizacji pamięci pozwalały uzyskać na zawodach około 80% maksymalnej punktacji (plik: `lasb1.cpp`).

Rozwiązanie alternatywne

Na problem można także spojrzeć z innej perspektywy i spróbować zastosować programowanie dynamiczne inaczej. Zamiast dla ustalonego x (oraz liczby strzałów) pamiętać maksymalną liczbę odcinków możliwych do przecięcia, można zapytać odwrotnie: jaka jest najmniejsza możliwa współrzędna końca odcinka, który został trafiony, przy założeniu, że użyto p strzałów oraz łącznie zestrzelono t odcinków. Można o tym myśleć jako o minimalizacji kąta ostatniego, p-tego strzału lasera dla wyniku t. Rozwiązanie to ma gorszą złożoność czasową: $O(nk \log(nk))$, z uwagi na konieczność zastosowania wyszukiwania binarnego, aby wyszukiwać najmniejszy możliwy kąt zapewniający zadany wynik. Przy prawidłowej implementacji rozwiązanie tego typu miało szansę uzyskać około 80% maksymalnej punktacji.

Rozwiązania naiwne oraz błędne

W przypadku braku pomysłu na rozwiązanie szybkie i poprawne, zawodnicy czasami podejmują próby uzyskania częściowej punktacji. Przykładowe rozwiązanie, które jest poprawne, ale niezbyt szybkie, opiera się na metodzie przeszukiwania z nawrotami: spośród możliwych pozycji potencjalnych strzałów wystarczy wybrać co najwyżej k – można spróbować albo przetestować je wszystkie, albo pewien ich podzbiór (i liczyć na to, że rozwiązanie optymalne zostanie znalezione). Takie rozwiązania miały szansę uzyskać około 20–30% punktów i zostały zaimplementowane w plikach `lass1.cpp` oraz `lass2.pas`.

Można było także zadanie próbować rozwiązywać zachłannie: na przykład pierwszy strzał wybrać tak, aby przeciął największą liczbę odcinków, kolejny, aby przeciął największą liczbę pozostałych itd. Takie rozwiązania (błędne algorytmicznie) warte były najczęściej 0 punktów. Rozwiązania tego typu zaimplementowane są w plikach `lasb2.cpp` oraz `lasb3.cpp`.

Testy

Rozwiązania były oceniane za pomocą 11 grup testów, z których większość zawierała po trzy testy:

- Testy z grupy a były testami losowymi; zadbano jednak, aby laser przecinał pewne zbiory końców odcinków pod tym samym kątem.
- Testy z grupy b zawierały odcinki krótkich długości; możliwe było przestrzelenie prawie wszystkich z nich i bezpieczne dokonanie k strzałów (aby żadnego odcinka nie przeciąć dwa razy).
- Testy z grupy c zostały przygotowane specjalnie w celu obalenia rozwiązań zachłannych – strzał w największą liczbę odcinków (lub jedno z kilku największych) uniemożliwiał osiągnięcie najlepszego wyniku.