Analiza algorytmów - Paweł Młyniec

Problem

Dane jest pomieszczenie, które w rzucie z góry jest wielokątem opisanym jako ciąg rzeczywisto liczbowych współrzędnych jego wierzchołków. Wewnątrz pomieszczenia umieszczonych jest n kamer, z których i-ta kamera umieszczona jest w punkcie x_i,y_i, skierowana jest w kierunku opisanym przez b_i oraz obserwuje wycinek koła o promieniu r_i oraz kącie środkowym α_i. Zadaniem jest opracować metodę, która obliczy jaki odsetek powierzchni obserwowany jest przez dokładnie n kamer.

Rozwiązanie ogólne:

Rozwiązanie ogólne:
1. Wczytuję dane i umieszczam je w strukturach
2. Sprawdzam czy kamery znajdują się w pomieszczeniu
3. n razy losuję punkt w przestrzeni
4. sprawdzam czy punkt jest w wielokącie
5. jeśli jest to sprawdzam czy jest w zasięgu wszystkich
6. Liczę stosunek punktów w zasięgu kamer do tych poza nim

Rozwiązanie szczegółowe:

Program podzieliłem na trzy części: część która wylicza stosunek pól wielokąta, klasę “areaCounter”. Klasę generującą wielokąt z kamerami, “polygonGenerator” oraz klasę odpowiedzialną za wyświetlanie “view”.
W klasie “areaCounter” ciekawe są pod względem algorytmicznym dwie funkcje:

```
isPointInPolygon(Punkt p)
dla każdej krawędzi:
  sprawdź czy prosta poprowadzona z punktu p do (p.y,nieskończoność) przecina się z krawędzią
  sprawdź z której strony x się przecina
  jeśli z lewej
    kontynuuj
  jeśli z prawej
    dodaj do zmiennej cross jeden
```

Jeśli cross%2 jest parzyste to punkt leży poza pokojem w przeciwnym wypadku leży w pokoju.

Sprawdzenie przecięć obliczam z przyrównania funkcji linowej y = ax + b przechodzącej przez wierzchołki oraz położenia x przecięcia, wstawiając p.y do równania prostej.

isPointInAllCameraView(polygon,punkt):
dla każdej kamery:
  przyrównaj odległość od punktu od kamery do długości promienia
  jeśli mniejsza
    policz kąt prostej przechodzącej przez punkt oraz kamerę do pionu
    jeśli pomiędzy b oraz b + alfa (czyli w kącie widzialności kamery)
      sprawdź czy odcinek punkt-kamera przecina się z dowolną krawędzią
      jeśli nie, dodaj do punktów obserwowanych przez kamery
  w przeciwnych wypadkach dodaj do punktów nieobserwowanych przez kamery

Przykład:
dla p = (2,0) oraz wielokąta a=(0,2) b=(4,2) c=(2,4) algorytm postąpi następująco:
 weźmie krawędź ab
 policzy x przecięcia z wzoru przecięcie.x = a.x + (p.y - a.y)*(a.x - b.x)/(a.y - b.y)
   przecięcie.x = 2 + (2 - 0)*(2 - 2)/(0 - 4) = 2
 przecięcie.x leży pomiędzy (a.x i b.x> oraz p.x < przecięcie.x
 dodaje jeden do zmiennej cross

weźmie krawędź bc
policzy x przecięcia
    przecięcie.x = 2 + (2 - 4)*(2 - 4)/(4 - 2) = 4
przecięcie x leży na (b.x i c.x> oraz p.x < przecięcie.x
dodaje jeden do zmiennej cross

weźmie krawędź ca
policzy x przecięcia
    przecięcie.x = 4 + (2 - 2)*(4 - 2)/(2 - 0) = 4
przecięcie x nie leży na (c.x i a.x> oraz p.x < przecięcie.x
nie dodaje jeden do zmiennej cross

cross%2 = 0 => punkt leży poza wielokątem

Algorytm użyty do tworzenia wielokąta:
1.Wygeneruj dwie list losowych współrzędnych x i y
2.Posortuj je
3.Wyizoluj ekstrema
4.Losowo podziel punkty na dwie części
5.Zbuduj z punktów losowe wektory które następnie dobierz w pary
6.Posortuj wektory względem kąta
7.Ułóż je w wielokąt który przenieś do punktów początkowych układu

Złożoność algorytmu

W moim programie spodziewam się złożoności O(n) względem liczby krawędzi. Wynika to z faktu,
Że algorytm sprawdza po kolei przecięcia tworzonych prostych z każdą krawędzią. Jeśli natomiast liczyć względem kamer dążących do nieskończoności jak i krawędzi to złożoność pesymistyczna wyszłaby O(n*m) gdzie n to liczba krawędzi a m liczba kamer. Wynika to z faktu, że dla punktu wylosowanego algorytm sprawdza czy nie jest w zasięgu każdej z kamer których jest m przy czym takie sprawdzenie ma też złożoność liniową. Można także dodać jako złożoność liczbę punktów monte carlo co powoduje wymnożenie złożoności p krotnie bo p razy wykonujemy sprawdzenie punktu.
Finalnie spodziewam się złożoności O(n*m*p) przy wszystkich parametrach dążących do nieskończoności lub O(n) zakładając że liczba punktów monte carlo i liczba kamer jest stała.

Dla wielokąta z małym pokryciem kamer:

	Vertices	Cameras	Result	PointsNr	Time	q(n)
0	4	50	0	1000	0.04811692238	0.06099531073
1	8	50	0	1000	0.02604818344	0.2253444382
2	16	50	0	1000	0.01687431335	0.6957098805
3	32	50	0	1000	0.02743625641	0.855774662
4	64	50	0	1000	0.05299425125	0.8861056624
5	128	50	0	1000	0.09480786324	0.9906036167
6	256	50	0	1000	0.1820108891	1.031993335
7	512	50	0	1000	0.3756680489	1
8	1024	50	0	1000	0.7449345589	1.008593424
9	2048	50	0	1000	1.497616053	1.003376128
10	4096	50	0	1000	2.961431742	1.014828182
11	8192	50	0	1000	5.996341467	1.002392678
12	16384	50	0	1000	12.10487199	0.993102411

						6
13	32768	50	0	1000	24.34124756	0.9877371761
14	65536	50	0	1000	51.19084787	0.9393380311
15	100000	50	0	1000	89.46648955	0.8201133872

Widać, że złożoność jest prawie liniowa poza przypadkiem na samym początku. Dzieje się tak dlatego, że podczas sprawdzania czy punkt jest w zasięgu kamer, jeśli sprawdzi, że nie jest w pierwszej przerywa sprawdzanie. Powoduje to od 0 dom-krotnego zwiększenia złożoności. Dalej pozostaje liniowa ale widać różnice w asymptocie O(T(n)). Dla bardzo dużych instancji problemu także widoczne jest odchylenie od założonej asymptoty, a dzieje się to dlatego, iż dane przestają się mieścić w najszybszych kieszeniach pamięci.

Dla wielokąta z różnym pokryciem kamer (z większą ilością i średnim kątem):

	Vertices	Cameras	Result	PointsNr	Time	
0	4	500	0.3004115226	1000	1.276622772	0.018793979
1	8	500	0.2316476346	1000	1.30455327	0.03678319946
2	16	500	0.1796407186	1000	0.3081440926	0.3114493791
3	32	500	0.0490797546	1000	0.1528978348	1.255364884
4	64	500	0.004784688995	1000	0.2938845158	1.306244884
5	128	500	0.01578947368	1000	0.4234573841	1.813099309
6	256	500	0.03529411765	1000	0.9677855968	1.586653682
7	512	500	0.1208791209	1000	3.071081161	1
8	1024	500	0.08235294118	1000	3.105618954	1.977757869
9	2048	500	0	1000	5.669877052	2.166594537
10	4096	500	0	1000	10.45163679	2.350698726

11	8192	500	0	1000	28.08767867	1.749425403
12	16384	500	0	1000	49.95182395	1.967387563
13	32768	500	0	1000	86.27943349	2.278053835
14	65536	500	0	1000	126.0154681	3.119445529
15	100000	500	0	1000	84.49349928	7.09901406

Widzimy tutaj, że asymptota zależy od wyniku czyli liczby kamer w których punkty są wudzialne.