Dokumentacja projektu

Michał Padzik
padzikm@student.mini.pw.edu.pl

Albert Wolant
wolanta@student.mini.pw.edu.pl

8 kwietnia 2015

1 Opis problemu

Golf Na polu golfowym znajduje się n piłeczek oraz n dołków na piłeczki. Golfiści chcą jednocześnie, każdy swoją piłeczkę, umieścić, w którymś z dołków. W tym celu ustalają między sobą, który celuje do którego dołka, ale w taki sposób, by tory ich piłeczek się nie przecinały, co gwarantuje brak zderzeń piłeczek. Załóżmy, że piłeczki i dołki są punktami na płaszczyźnie oraz, że żadne trzy z tych punktów nie są współliniowe i że tory piłeczek są odcinkami prostej. Przedstaw działający w czasie $O(n^2 log(n))$ algorytm przydzielania piłeczek do dołków tak, aby żadne dwa tory piłeczek nie przecinały się.

2 Format danych

W poniższej sekcji znajduje się proponowany format danych wejściowych i wyjściowych razem z przykładami.

2.1 Dane wejściowe

Dane wejściowe mają format pliku tekstowego. Struktura tego pliku jest następująca:

1. Liczba par (golfista, dołek) w nowej linii.
2. Współrzędne x i y golfisty, oddzielone spacją, w nowej linii. Ta linia występuje tyle razy, ile jest golfistów.
3. Pusta liniia.
4. Współrzędne x i y dołka, oddzielone spacją, w nowej linii. Ta linia występuje tyle razy, ile jest golfistów.

2.2 Dane wyjściowe

Dane wyjściowe mają format pliku tekstowego. Plik ten składa się z sekwencji następującej postaci:

1. Współrzędne x i y golfisty, oddzielone spacją, w nowej linii.
2. Współrzędne x i y dołka, do którego strzela powyższy golfista, oddzielone spacją, w nowej linii.
3. Pusta linia.

W pliku występuje tyle sekwencji przedstawionej wyżej postaci, ile jest par (golfista, dołek). Sekwencje są umieszczone jedna po drugiej.

2.3 Przykład

2.3.1 Przykładowy plik wejściowy

```
3
0 0
2.5 3.5
3.5 4.5

1 0
10.45 20.5
4 2
```

Plik zawiera dane 3 golfistów o współrzędnych $(0;0)$, $(2.5;3.5)$ i $(3.5;4.5)$ oraz 3 dołków o współrzędnych $(1;0)$, $(10.45;20.5)$ i $(4;2)$.

2.3.2 Przykładowy plik wyjściowy

```
0 0
1 0

2.5 3.5
4 2

3.5 4.5
10.45 20.5
```

Plik zawiera 3 dopasowane pary. Golfista o współrzędnych $(0;0)$ strzela do dołka o współrzędnych $(1;0)$, golfista o współrzędnych $(2.5;3.5)$ strzela do dołka o współrzędnych $(4;2)$, a golfista o współrzędnych $(3.5;4.5)$ strzela do dołka o współrzędnych $(10.45;20.5)$.

3 Opis rozwiązania

Poniżej zamieszczamy proponowany algorytm rozwiązania problemu w postaci opisowego pseudokodu:

Dane

- Lista G punktów oznaczających pozycje golfistów.
- Lista D punktów oznaczających pozycje dołków.

Algorytm

1. $s :=$ punkt o najmniejszej współrzędnej y w listach G i D (jeżeli jest kilka takich punktów to weź ten z najmnjiejszą współrzędną x).
2. Złącz pozostałe punkty w listę C.
3. Posortuj listę C kątowo, przeciwnie do ruchu wskazówek zegara, względem punktu s.
4. Idąc od początku listy C stwórz sumę S.
   (a) $S := 0$.
   (b) $c :=$ pierwszy punkt listy C.
   (c) Dopóki $S \neq -1$
      i. Jeżeli s jest dołkiem i c jest golfistą to $S - -$
      ii. Jeżeli s jest dołkiem i c jest dołkiem to $S + +$
      iii. Jeżeli s jest golfistą i c jest golfistą to $S + +$
      iv. Jeżeli s jest golfistą i c jest dołkiem to $S - -$
      v. $c :=$ kolejny element na licie C.
5. Połącz s z punktem poprzedzającym c na liście C. Niech ten punkt to p.
6. Rekurencyjnie wywołaj algorytm dla punktów poprzedzających p na liście C oraz dla punktów następujących po p na liście C.

4 Analiza poprawności

Poniżej zamieszczamy dowód, że dopasowanie punktów zdefiniowane w zadaniu zawsze istnieje oraz przedstawiony powyżej algorytm znajduje takie dopasowanie. Będzie to dowód indukcyjny ze względu na liczbę par (golfista, dołek). Oznaczmy liczbę tych par w zadaniu przez n.

1. Jeśli $n = 1$, dopasowanie zawsze istnieje w sposób oczywisty poprzez połączenie jedynych dwóch punktów zadania. Algorytm znajdzie takie dopasowanie, ponieważ wybierze jeden z punktów jako s i dołączy do niego drugi jako jedyny punkt listy C.

2. Dla $n > 1$ zakładamy, że dopasowanie istnieje dla każdego $k < n$ par i algorytm je znajduje.
   Wykonajmy teraz przedstawiony algorytm bez zejść rekurenycjnych z punktu 6. Po pierwsze, suma S zawsze osiągnie wartość -1. Wynika to z faktu, że początkowo w zbiorze jest tyle samo golfistów i dołków. Po wyznaczeniu i wyjęciu ze zbioru punktu s, punktów jednej z tych grup jest o jeden więcej niż drugiej. Dodatkowo wśród pozostałych punktów liczniejsze będą punkty innej kategorii niż punkt s (np. jeśli s jest dołkiem to będzie więcej golfistów). W takiej sytuacji, mając na uwadze zasady konstruowania sumy S widać, że zawsze osiągnie ona wartość -1, być może po przejżeniu wszystkich punktów. Zatem zawsze wyznaczymy punkt p i otrzymamy 2 zbiory punktów oddzielone prostą sp. W każdym z tych zbiorów jest tyle samo dołków i golfistów i jest tam mniej niż n par. Zwróćmy uwagę, że wartość S nigdy nie jest mniejsza od -1, ponieważ w chwili osiągnięcia -1 przeglądanie punktów jest przerywane. Wynika stąd, że przed analizą punktu p suma S była równa 0. Znaczy to, że wśród odwiedzonych do tej pory punktów jest tyle samo dołków co golfistów. Dodatkowo, ponieważ początkowo w zbiorze grupy golfistów i dołków były równoliczne i punkty s i p należą do różnych grup wiemy, że wśród punktów jeszcze nie odwiedzonych tyle samo jest dołków i golfistów. Oczywiste jest też, że w każdym z tych podzbiorów jest mniej niż n par. Zatem dla każdego z tych zbiorów istnieje odpowiednie dopasowanie i przytoczony algorytm je znajduje. Wynika to z założenia indukcyjnego. Dodatkowo rozwiązania można połączyć bez straty poprawnoci, bo leżą na różnych półpłaszczyznach. Po dodaniu do tego rozwiązania odcinka sp mamy rozwiązanie problemu rozmiaru n i znalezione zostało przedstawionym algorytmem.

3. Z zasady indukcji matematycznej wynika, że przedstawiony algorytm jest poprawny.

5 Analiza złożonoci

Niech n oznacza liczbę par (golfista, dołek) w zadaniu.

- Punkt 1. algorytmu można wykonać w czasie $O(n)$.
- Punkt 2. algorytmu można wykonać w czasie $O(1)$ przy odpowiedniej implementacji.
- Sortowanie kątowe w punkcie 3. można wykonać w czasie $O(nlog(n))$.
- W pesymistycznym wypadku w punkcie 4. trzeba będzie przejżeć całą listę. Trwa to $O(n)$.
- Dodanie nowej pary do rozwiązania można wykonać w czasie stałym.
- W pesymistycznym przypadku, czyli wtedy gdy punkt p jest ostatnim na liście C, zawsze będzie wykonywane tylko jedno wywołanie rekurencyjne.

W takim przypadku podzbiór, dla którego przeprowadzimy to zejście rekurenycjne, ma jedną parę mniej. Zatem pesymistycznie wykonamy $O(n)$ zejść rekurencyjnych.

Podsumowując każde wywołanie rekurencyjne ma złożoność $O(nlog(n))$. W przypadku pesymistym wykonanych będzie $O(n)$ takich wywołań. Zatem pesymistyczna złożoność czasowa całego algorytmu wynosi $O(n^2log(n))$.