

[AAL] Convex hull

Piotr Frątczak
(300207)

Bartosz Świtalski
(300279)

27 listopada 2020

1 Problem

Dany jest zbiór punktów w przestrzeni trójwymiarowej. Wyznaczyć zadaną dokładnością jego powłokę wypukłą (tzn. tak, aby żaden z punktów nie wystawał ponad znaną powłokę bardziej niż zostanie to dopuszczone). Porównać czas obliczeń i wyniki różnych metod.

2 Decyzje projektowe

Wybrany język programowania to C++. Do projektu zostanie dołączony plik `doc/manual.pdf` z instrukcją uruchamiania programu.

Zakładamy, że nie wiemy, które z podanych punktów są punktami powierzchniowymi¹.

3 Implementowane algorytmy

Opis

- **Algorytm naiwny** Naiwne sprawdzanie każdej trójki punktów czy jest ścianą zewnętrzną powłoki. Wykonają się 4 pętle, każda po n razy.
- **Gift Wrapping** Wykorzystuje fakt, że znamy już punkty należące do powłoki wypukłej i, mając krawędź k należącą do takiej ściany, znajdziemy kolejną ścianę przez znalezienie punktu po drugiej stronie krawędzi niż znana już ściana. Dzięki temu nie musimy przeglądać wszystkich trójek, lecz dobieramy kolejny punkt do krawędzi znanej ściany zewnętrznej tak, aby nowa ściana także była zewnętrzną.
- **Incremental** Polega na dołączaniu punktów leżących na powłoce do znanych już punktów leżących na powłoce.
- **Dziel i zwyciężaj** Klasyczny podział na podproblemy z rekursywnym wywołaniem z dobraną funkcją skalania.

¹leżą na powierzchni ściany powłoki wypukłej

Działanie

Algorithm 1 Algorytm naiwny

```
1: procedure NAIVE(punkty)
2:   for all  $i \in \textit{punkty}$  do
3:     for all  $j \in \textit{punkty} \setminus \{i\}$  do
4:       for all  $k \in \textit{punkty} \setminus \{i, j\}$  do
5:         dodaj  $(i, j, k)$  do zbioru ścian powłoki zewnętrznej
6:         for all  $l \in \textit{punkty} \setminus \{i, j, k\}$  do
7:           if  $l$  jest po zewnętrznej stronie płaszczyzny opisanej przez  $(i, j, k)$  then
8:             usuń  $(i, j, k)$  ze zbioru ścian powłoki zewnętrznej
9:           end if
10:        end for
11:      end for
12:    end for
13:  end for
14:  return zbiór ścian powłoki zewnętrznej
15: end procedure
```

Algorithm 2 Gift Wrapping

```
1: procedure GIFTWRAPPING(punkty)
2:   while nie znaleziono pierwszej ściany do
3:     algorytmNaiwny(punkty) ▷ do momentu znalezienia pierwszej ściany
4:   end while
5:   for all krawędź  $k \in$  ściany ze zbioru  $s$  ścian powłoki zewnętrznej do
6:     if  $k$  jest jedną z krawędzi dokładnie jednej ściany w  $s$  then
7:       for all  $p \in \textit{punkty}$  do
8:         if  $p \notin$  żadnej ściany ze zbioru ścian powłoki zewnętrznej then
9:           ▷ ściślej:  $p \notin$  żadnej ściany powłoki zewnętrznej
10:          if  $(k, p)$  opisuje ścianę zewnętrzną then
11:            dodaj ścianę  $(k, p)$  do  $s$ 
12:          end if
13:        end if
14:      end for
15:    end if
16:  end for
17: end procedure
```

Algorithm 3 Incremental

```
1: procedure INCREMENTAL(punkty)
2:   znajdź pierwsze 4 punkty leżące na powłoce zgodnie z algorytmem Gift Wrapping
3:   for  $i \leftarrow 5, \text{sizeof}(\textit{punkty})$  do
4:     for all  $s \in$  zbioru ścian powłoki zewnętrznej do
5:       if  $p[i]$  leży po zewnętrznej stronie  $s$  then
6:         połącz  $p[i]$  krawędziami z powłoką zewnętrzną
7:       else
8:         odrzuć  $p[i]$ 
9:       end if
10:    end for
11:  end for
12: end procedure
```

Algorithm 4 Dziel i zwyciężaj

```
1: procedure DZIELIZWYCIEZAJ(punkty)
2:    $p1, p2 \leftarrow$  podziel  $p$  na podproblemy
3:    $pz1 \leftarrow$  dzielIZwyciezaj( $p1$ ) ▷ powłoka zewnętrzna 1
4:    $pz2 \leftarrow$  dzielIZwyciezaj( $p2$ ) ▷ powłoka zewnętrzna 2
5:   merge( $pz1, pz2$ )
6: end procedure
7: procedure MERGE( $pz1, pz2$ )
8:   połącz trójkątnymi ścianami  $pz1$  i  $pz2$  tworząc  $p$  ▷ tymczasowa powłoka
9:   uzyskaj  $pz$  poprzez usunięcie ścian wewnętrznych w  $p$ 
10: end procedure
```

Teoretyczna pesymistyczna złożoność

Algorytm	$O(T(n))$
Algorytm naiwny	$O(n^4)$
Gift Wrapping	$O(n^2)$
Incremental	$O(n^2)$
Dziel i zwyciężaj	$O(n \cdot \log(n))$

4 Użycie

Dane z pliku wejściowego

```
./chull << [plik_wejscowy] >> [plik_wyjscowy]
```

Automatyczna generacja danych na podstawie parametrów:

```
./chull -n [liczba_punktow] -d [dokladnosc]  
-seed [seed] >> [plik_wyjscowy]
```

Testowanie z generacją danych, pomiarem czasu i prezentacją wyników

```
./chull -n [liczba_punktow] -d [dokladnosc]  
-p [liczba_problemov] -step [krok]  
-r [liczba_instancji_problemu]
```

5 Konwencje

Wejście

Parametry `-n`, `-p`, `-step`, `-r` powinny być liczbami całkowitymi. Parametry `-d` oraz `-seed` powinny być niewielką liczbą rzeczywistą² z kropką jako separatorem dziesiętnym.

Każdy wiersz pliku wejściowego składa się z 3 liczb rzeczywistych opisujących współrzędne punktu na osiach x, y, z , oddzielonych średnikiem, z kropką jako separatorem dziesiętnym. Wiersze kończą się znakiem nowej linii. Kolejność punktów jest dowolna.

w Przykładowe wejście:

```
./chull -n 1000 -d 0.1 -p 3 -step 500 -r 5
```

Pomiar czasu i prezentacja wyników dla 3 problemów o wielkościach 1000, 1500 i 2000. Dla każdej wielkości losowanych 5 instancji problemu.

Przykładowy plik `input.txt`:

```
2.15;0.333;4.01  
4.665;2.643;9.06
```

Wyjście

Format pliku wyjściowego jest podobny do wejściowego, lecz w tym wypadku każde 3 kolejne punkty (w trzech kolejnych wierszach) opisują ścianę powłoki wypukłej. Dodatkowo pierwszy punkt i dwa ostatnie oraz dwa pierwsze i ostatni punkt również opisują ścianę powłoki

² ≤ 1

wypukłej.

Przykładowe wyjście 1 :

2.15;0.333;4.01
4.665;2.643;9.06
5.11;7.45;7.88
1.1;0.34;5.999

Przykładowe wyjście 2 :

Algorytm z asymptotą $O(T(n))$ ³		
n ⁴	$t(n)$ ⁵	$q(n)$ ⁶
n_1		.
n_2		.
...
n_{mediana}	$t(n_{\text{mediana}})$	1.0
...
n_{k-1}	$t(n_{k-1})$.
n_k	$t(n_k)$.

6 Wyniki

Reprezentacja graficzna

Dla danych z pliku wejściowego oraz automatycznej generacji na podstawie parametrów zostanie wygenerowany graf 3D z powłoką wypukłą będącą rozwiązaniem dla danego zbioru wejściowego.

³analiza teoretyczna

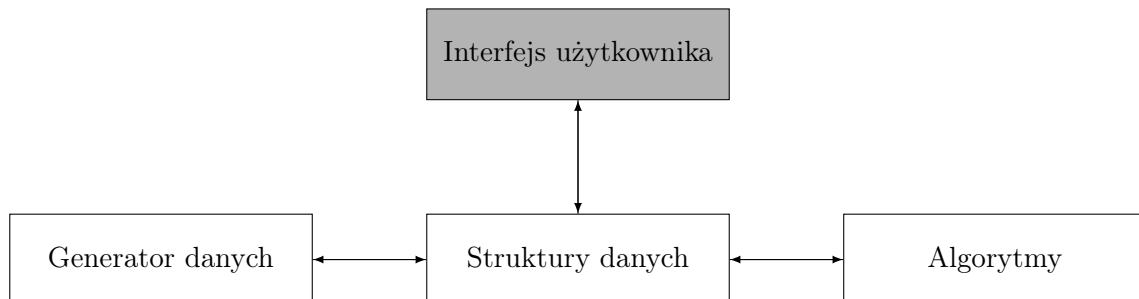
⁴rozmiar problemu

⁵średni czas wykonania kilkunastu (kilkudziesięciu) uruchomień dla wartości n

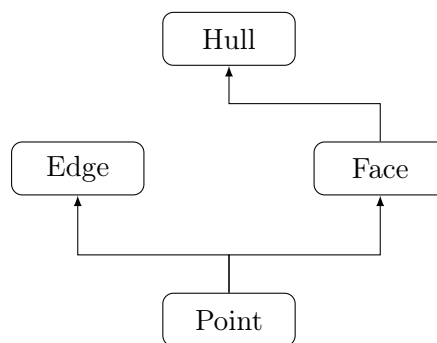
⁶ $t(n)/O(T(n))$

7 Struktura programu


Podział na moduły



Hierarchia klas



8 Powiązane linki

 [Repozytorium projektowe](#)