Înfăşurătoarea convexă

Capitolul 18

- **Un algoritm ineficient**
- **Scanarea Graham**
- **Potrivirea Jarvis**
- **Rezumat**
- **Implementări sugerate**
- **Probleme propuse**
- **Soluțiile problemelor**

În cadrul acestui capitol vom prezenta noțiunea de înfăşurătoare convexă şi modul în care aceasta poate fi determinată. Vom prezenta mai întâi un algoritm simplu, ineficient, care poate fi utilizat pentru a determina înfăşurătoarea convexă, iar apoi vom descrie doi algoritmi mai performanți.

Intuitiv, putem considera mulțimea de puncte ca fiind o colecție de cuie. În acest caz înfăşurătoarea convexă poate fi asemuită cu un fir de ață (cu grosime neglijabilă) care înconjoară toate cuiele.

Determinarea înfăşurătorii convexe este o problemă destul de des întâlnită. Mulți algoritmi de geometrie analitică necesită, ca pas intermediar, determinarea unei astfel de înfăşurători.

Uneori, în locul denumirii de înfăşurătoare convexă se mai utilizează şi termenul de învelitoare convexă.

18.1. Un algoritm ineficient

Vom începe studiul învelitorilor convexe prin prezentarea unui algoritm simplu, dar ineficient care poate fi utilizat pentru determinarea lor.

Se observă faptul că o învelitoare convexă poate fi descrisă fie prin mulțimea vârfurilor sale, fie prin mulțimea laturilor sale. Evident, mulțimea vârfurilor poate fi determinată pe baza mulțimii laturilor şi invers.

18.1.1. Prezentarea algoritmului

Pentru ca un poligon să fie convex, trebuie ca toate celelalte vârfuri ale sale aflate de aceeaşi parte a unei laturi. Mai mult, pentru ca poligonul respectiv să fie o înfăşurătoare convexă a unei mulțimi de puncte va trebui ca şi celelalte puncte aflate de aceeaşi parte a dreptei suport a laturii.

Aşadar, pentru a determina laturile înfăşurătorii convexe va trebui să verificăm, pentru fiecare latură, dacă toate celelalte puncte se află de aceeaşi parte a ei.

Ca urmare, vom considera toate perechile de puncte şi vom determina, pentru fiecare pereche, dreapta suport a segmentului care uneşte cele două puncte care fac parte din perechea respectivă. După determinarea acestei drepte, vom verifica dacă celelalte puncte se află de aceeaşi parte a ei.

Vom considera că trebuie determinată înfăşurătoarea convexă a unei mulţimi de N puncte ale căror coordonate sunt cunoscute. Aşadar, vom avea $N \cdot (N - 1) / 2$ perechi de puncte şi pentru fiecare dintre acestea va trebui să verificăm dacă celelalte $N - 2$ puncte (care nu fac parte din pereche) se află de aceeaşi parte a segmentului care uneşte punctele din pereche.

18.1.2. Analiza complexităţii

Se observă imediat că, pentru a verifica dacă $N - 2$ puncte se află de aceeaşi parte a unei drepte, va trebui să alegem un punct P care nu se află pe dreapta respectivă şi apoi, pentru fiecare dintre celelalte $N - 3$ puncte să verificăm dacă se află de aceeaşi parte a dreptei ca şi punctul P. Aşadar, pentru fiecare pereche de puncte, ordinul de complexitate al verificării faptului dacă perechea determină o latură a înfăşurătorii convexe se realizează într-un timp de ordinul $O(N)$.

Determinarea ecuaţiei unei drepte pentru care se cunosc coordonatele a două puncte se realizează pe baza unor operaţii matematice simple, deci operaţia se va realiza în timp constant.

Datorită faptului că vom verifica $N \cdot (N - 1) / 2$ perechi de puncte, ordinul de complexitate al întregului algoritm devine $O(N^3)$ şi, aşa cum vom vedea, acest ordin indică faptul că algoritmul nu este deloc eficient.

18.2. Scanarea Graham

Vom descrie acum un algoritm eficient care poate fi utilizat pentru a determina înfăşurătoarea convexă a unei mulţimi de puncte. Această metodă se bazează pe determinarea unor unghiuri polare în jurul unui punct, motiv pentru care vom începe cu definirea acestei noţiuni. Ulterior, vom arăta modul în care poate fi identificată înfăşurătoarea convexă pe baza unghiurilor polare.

18.2.1. Unghiuri polare

Pentru un punct P dat, unghiul polar pe care îl formează acesta cu un alt punct Q este dat de unghiul format de dreapta orizontală care trece prin P şi dreapta suport a segmentului care uneşte punctul P de punctul Q.

Dacă se cunosc coordonatele punctelor P şi Q putem determina foarte uşor distanţa de la P la Q (pe baza formulei care furnizează distanţa dintre două puncte date), precum şi distanţa de la punctul P la proiecţia punctului Q pe dreapta orizontală care trece

prin P (această distanță este dată de diferența coordonatelor orizontale ale celor două puncte).

Cu ajutorul acestor două valori putem determina foarte simplu cosinusul unghiului polar şi, dacă dorim, prin aplicarea funcției `arccos`, a valorii exacte a unghiului.

18.2.2. Prezentarea algoritmului

Scanarea *Graham* rezolvă problema înfăşurătorii convexe prin păstrarea unei stive (aceasta poate fi simulată folosind orice tehnică) care conține puncte. Fiecare vârf dat este inserat în stivă o singură dată, iar punctele care nu fac parte din înfăşurătoare sunt eliminate pe parcurs. Aşadar, în final stiva va conține vârfurile înfăşurătorii convexe în ordine trigonometrică.

Algoritmul este foarte simplu şi necesită parcurgerea a patru paşi. Inițial se determină vârful p_0 care are coordonata verticală cea mai mică şi, în cazul în care există mai multe astfel de vârfuri se alege cel care are coordonata orizontală cea mai mică.

La al doilea pas vom sorta celelalte vârfuri în ordinea unghiurilor polare în jurul vârfului p_0. Dacă există mai multe vârfuri cu acelaşi unghi polar, atunci se păstrează doar cel care se află la distanța maximă față de p_0. Pentru aceasta va trebui utilizat un algoritm de sortare eficient, al cărui ordin de complexitate să fie $O(N \cdot \log N)$. Vom identifica vârfurile sortate prin $p_1, p_2, \ldots, p_m$.

Pasul al treilea este extrem de simplu şi constă doar în inserarea în stivă a vârfurilor p_0, p_1 şi p_2.

La al patrulea pas vom considera pe rând, nodurile $p_3, p_4, \ldots, p_m$. Vom nota nodul curent prin p_i, nodul care se află în vârful stivei prin p_{i-1} şi următorul nod din stivă prin p_{i-2}. Pentru fiecare nod p_i vom verifica dacă unghiul format din vârfurile p_{i-2}, p_{i-1} şi p_i nu formează un unghi care să anuleze convexitatea poligonului. Practic acest unghi trebuie să realizeze o "întoarcere" spre stânga. Acest lucru poate fi verificat foarte simplu cu ajutorul determinantului folosit pentru calculul ariei sau verificarea coliniarității. Pentru a se păstra conexitatea, determinantul trebuie să fie nagativ. În cazul în care nu se păstrează conexitatea, vârful p_{i-1} este eliminat din stivă; aşadar p_{i-2} devine p_{i-1}, şi se consideră următorul nod din stivă ca fiind p_{i-2}. Verificarea continuă (cu eliminări succesive ale nodurilor) până în momentul în care unghiul format nu indică pierderea convexității. În final, stiva va conține doar vârfurile înfăşurătorii convexe, în ordine trigonometrică.

O demonstrație intuitivă a acestui algoritm se bazează pe faptul că, prin verificarea unghiurilor şi a eliminărilor se îndepărtează concavitățile. Aşadar, în final vom avea un poligon convex. Este evident faptul că, datorită modului în care se alege vârful p_0, acesta va trebui să facă parte obligatoriu din înfăşurătoare. De asemenea, putem observa că nodul p_m nu poate fi eliminat în nici o situație. Totuşi, este foarte uşor de observat că, datorită modului în care a fost determinat acesta, şi el va face parte întotdeauna din înfăşurătoare.

18.2.3. Analiza complexității

Primul pas al algoritmului se realizează în timp liniar deoarece constă într-o singură parcurgere (necesară pentru determinarea unui minim).

Cel de-al doilea pas implică o sortare a unor unghiuri polare, operație ce necesită un timp de ordinul $O(N \cdot \log N)$. Determinarea unui unghi polar implică doar calcule matematice simple, deci se realizează în timp constant. Aşadar, pentru determinarea tuturor unghiurilor polare vom avea nevoie de un timp al cărui ordin de complexitate este $O(N)$.

Cel de-al treilea pas necesită un timp de execuție constant, el constând în simpla inserare a trei elemente într-o stivă.

Cel de-al patrulea pas se execută într-un timp liniar deoarece fiecare nod este inserat în stivă o singură dată şi este eliminat cel mult o dată. Fiecare inserare necesită verificarea unui unghi, operație ce se poate realiza în timp constant, iar fiecare eliminare necesită o verificare suplimentară a unui unghi (realizabilă tot în timp constant). Deoarece avem cel mult N inserări şi cel mult $N - 3$ eliminări (înfăşurătoarea conține cel puțin trei puncte), ordinul de complexitate al operațiilor efectuate este $O(N)$.

În concluzie ordinul de complexitate al algoritmului de determinare a înfăşurătorii convexe pe baza scanării *Graham* este $O(N \cdot \log N)$. Practic toate operațiile, cu excepția sortării, se realizează în timp liniar.

18.3. Potrivirea Jarvis

Tehnica folosită pentru determinarea înfăşurătorii convexe pentru potrivirea Jarvis poartă denumirea de *împachetarea pachetului* (sau împachetarea cadoului).

Intuitiv, se simulează modul de înfăşurare a unei benzi de hărtie în jurul mulțimii de puncte. La fel ca şi în cazul scanării *Graham*, vom începe cu punctul cel mai de jos şi, dacă există mai multe astfel de puncte, cu cel mai din stânga (coordonată verticală minimă şi, în caz de egalitate, coordonată orizontală minimă).

Acum vom întinde hărtia spre dreapta şi apoi o deplasăm în sus până în momentul în care atinge un punct (cui). Acesta va face şi el parte din înfăşurătoarea convexă. Păstrând hârtia întinsă, continuăm în acelaşi mod până în momentul în care ajungem la punctul de pornire.

18.3.1. Prezentarea algoritmului

Practic, prin potrivirea *Jarvis* se construieşte o secvență a punctelor de pe înfăşurătoarea convexă. Se începe din punctul p_0 şi următorul punct este dat de cel care are cel mai mic unghi polar în raport cu el. Dacă există mai multe astfel de puncte, se alege cel mai îndepărtat punct față de p_0. Vom continua în acelaşi mod până în momentul în care vom ajunge la cel mai de sus punct (este uşor de observat că acesta va face parte din înfăşurătoare). În acest moment am construit "partea dreaptă a înfăşurătorii". Pentru a construi partea stângă vom porni de la cel mai de sus punct şi vom alege punctele

pe baza unghiurilor polare, inversând sensul axei *Ox*. În final vom ajunge din nou la punctul p_0.

Potrivirea *Jarvis* poate fi implementată şi ca o deplasare în jurul înfăşurătorii (fără a mai fi necesară construirea separată a celor două părți), dar în această situație va trebui ca la fiecare pas să determinăm unghiurile formate cu dreapta suport a ultimei laturi şi nu cu o dreaptă orizontală.

18.3.2. Analiza complexității

Practic, la fiecare pas va trebui să determinăm valorile a $O(N)$ unghiuri. Numărul paşilor efectuați va depinde de numărul h al nodurilor de pe înfăşurătoarea convexă. Aşadar, ordinul de complexitate al algoritmului va fi $O(N \cdot h)$.

Din nefericire, numărul nodurilor de pe înfăşurătoare poate fi chiar N (sau foarte apropiat de N), caz în care ordinul de complexitate devine $O(N^2)$. Totuşi, în cazul în care numărul nodurilor este foarte mic, ordinul de complexitate ajunge să fie, practic, $O(N)$.

Se observă faptul că, în cazul în care numărul nodurilor de pe înfăşurătoare are ordinul $O(\log N)$, cele două metode au acelaşi ordin de complexitate. Totuşi, potrivirea *Jarvis*, datorită detaliilor de implementare, va fi mai rapidă.

Aşadar, se recomandă utilizarea scanării *Graham* doar în situațiile în care numărul nodurilor de pe înfăşurătoarea convexă este relativ mare. În celelalte situații, potrivirea *Jarvis* reprezintă o alegere mai bună.

Avantajul principal al potrivirii *Jarvis* îl constituie faptul că punctele nu mai trebuie sortate în vederea determinării înfăşurătorii.

18.4. Rezumat

În cadrul acestui capitol am prezentat noțiunea de înfăşurătoare convexă şi am descris trei algoritmi care pot fi utilizați pentru determinarea acesteia.

Primul dintre ei este foarte uşor de implementat, dar timpul de execuție este inacceptabil de mare.

Următorii doi algoritmi, scanarea *Graham* şi potrivirea *Jarvis*, au timpi de execuție mult mai mici şi oricare dintre ei poate fi utilizat cu succes.

În final am conluzionat că, exceptând cazul în care suntem siguri că o mare parte dintre punctele date vor face parte din înfăşurătoare, este recomandabilă utilizarea potrivirii *Jarvis*.

18.5. Implementări sugerate

Pentru a vă obişnui cu modul în care va trebui să implementați algoritmii de determinare a înfăşurătorii convexe în diferite sutuații, vă sugerăm să scrieți programe pentru:

1. determinarea înfăşurătorii convexe folosind algoritmul ineficient;

2. determinarea înfăşurătorii convexe folosind scanarea *Graham*;
3. determinarea înfăşurătorii convexe folosind potrivirea *Jarvis*, determinând separat partea dreaptă şi partea stângă a înfăşurătorii;
4. determinarea înfăşurătorii convexe folosind potrivirea *Jarvis* fără a construi separat partea dreaptă şi partea stângă a înfăşurătorii.

18.6. Probleme propuse

În continuare vom prezenta enunţurile câtorva probleme pe care vi le propunem spre rezolvare. Toate aceste probleme pot fi rezolvate folosind informaţiile prezentate în cadrul acestui capitol. Cunoştinţele suplimentare necesare sunt minime.

18.6.1. Elfi

Descrierea problemei

În urma conflictului dintre *elfi* şi *orci* sistemul de apărare al elfilor a fost serios afectat. Din acest motiv se doreşte construirea unui zid magic de apărare care să cuprindă în interiorul său toate locuinţele elfilor.

Deoarece mulţi dintre elfii cu capacităţi magice au fost ucişi în luptă, se doreşte ca lungimea totală a zidului să fie minimă. Aşadar, zidul va avea forma unui poligon convex care va conţine toate casele elfilor în vârfuri, pe laturi sau în interior.

Date de intrare

Prima linie a fişierului de intrare `ELFI.IN` conţine numărul N al locuinţelor elfilor. Următoarele N linii conţin perechi de numere, separate printr-un spaţiu, reprezentând coordonatele unei locuinţe.

Date de ieşire

Fişierul de ieşire `ELFI.OUT` va conţine pe prima linie numărul K al vârfurilor poligonului care reprezintă zidul magic. Pe fiecare dintre următoarele K linii se vor afla două numere reprezentând coordonatele unui vârf al poligonului. Vârfurile poligonului vor fi scrise în sens trigonometric.

Restricţii şi precizări

- $1 \leq N \leq 5000$;
- nu pot exista două locuinţe la aceleaşi coordonate;
- există posibilitatea ca trei sau mai multe locuinţe să aibă coordonatele situate pe aceeaşi linie;
- coordonatele sunt numere întregi cuprinse între 1 şi 5000.

Exemplu

ELFI.IN	**ELFI.OUT**
5	4
10 10	10 10
10 20	10 20
15 15	20 20
20 10	20 10
20 20	

Timp de execuție: 1 secundă/test

18.6.2. Teritoriu

Descrierea problemei

Teritoriul controlat de celebra regină *Catherine a Erathiei* este determinat de poligonul de perimetru minim care conține în interior, pe laturi sau în vârfuri toate orașele regatului său. Ea dorește să afle care este suprafața exactă a teritoriului pe care îl controlează.

Date de intrare

Prima linie a fișierului de intrare **ARIE.IN** conține numărul N al orașelor din Erathia. Următoarele N linii conțin perechi de numere, separate printr-un spațiu, reprezentând coordonatele unui oraș.

Date de ieșire

Fișierul de ieșire **ARIE.OUT** va conține o singură linie pe care se va afla un număr, reprezentând suprafața teritoriului controlat de regină.

Restricții și precizări

- $1 \leq N \leq 5000$;
- nu pot exista două orașe la aceleași coordonate;
- există posibilitatea ca trei sau mai multe orașe să aibă coordonatele situate pe aceeași linie;
- coordonatele sunt numere întregi cuprinse între 1 și 5000;
- valoarea ariei va fi scrisă cu două zecimale exacte.

Exemplu

ARIE.IN	**ARIE.OUT**
5	100.00
10 10	
10 20	

```
15 15
20 10
20 20
```

Timp de execuţie: 1 secundă/test

18.6.3. Fred Flinstone

Descrierea problemei
Într-o placă sunt înfipte N cuie cu dimensiuni neglijabile astfel încât nu există trei cuie ale căror coordonate sunt puncte coliniare. *Fred Flinstone* i-a o aţă de o anumită culoare şi înconjoară cuiele, astfel încât lungimea totală a aţei să fie minimă. Apoi el elimină cuiele atinse de aţă şi repetă procedeul folosind o aţă de o altă culoare. După ce elimină din nou cuiele atinse de aţă, alege o altă culoare şi procedeul se repetă până în momentul în care nu mai rămâne nici un cui. Va trebui să determinaţi, pentru fiecare culoare în parte, cuiele atinse de aţa având culoarea respectivă.

Date de intrare
Prima linie a fişierului de intrare **`FRED.IN`** conţine numărul N al cuielor de pe tablă. Fiecare dintre următoarele N linii va conţine o pereche de numere reale reprezentând coordonatele unui cui, separate prin câte un spaţiu.

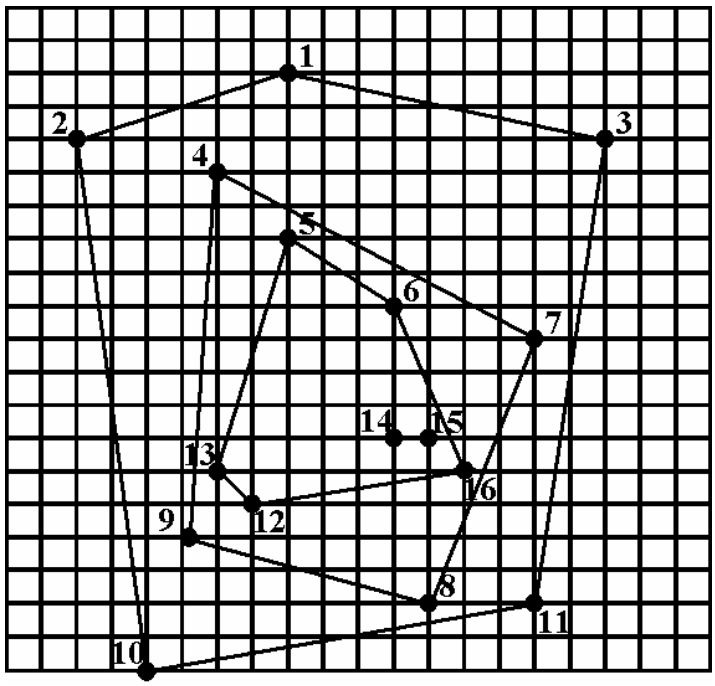
Date de ieşire
Prima linie a fişierului de ieşire **`FRED.OUT`** va conţine numărul K al culorilor folosite de *Fred Flinstone*. Fiecare dintre următoarele K linii corespunde unei anumite culori. Primul număr de pe o astfel de linie reprezintă numărul C al cuielor atinse de aţa având culoarea respectivă, iar următoarele C numere reprezintă numerele de ordine ale celor C cuie. Numerele de pe o linie vor fi separate prin spaţii.

Restricţii şi precizări
- $1 \le N \le 5000$;
- cuiele sunt identificate prin numere naturale cuprinse între 1 şi N.
- coordonatele cuielor sunt numere reale cu cel mult trei zecimale exacte, cuprinse între 0 şi 1000;
- există posibilitatea ca ultima aţă să atingă doar unul sau două cuie.

Exemplu

FRED.IN

```
16
8 2
2 4
17 4
6 5
8 7
11 9
15 10
12 18
5.1 16
4 20
15 18
7 15
6 14
11 13
12 13
13 14
```



FRED.OUT

```
4
5 1 2 3 10 11
4 4 7 8 9
5 5 6 12 13 16
2 14 15
```

Timp de execuție: 3 secunde/test

18.7. Soluțiile problemelor

Vom prezenta acum soluțiile problemelor propuse în cadrul secțiunii precedente. Pentru fiecare dintre acestea va fi descrisă metoda de rezolvare şi va fi analizată complexitatea algoritmului prezentat.

18.7.1. Elfi

Problema se reduce la a determina înfăşurătoarea convexă a punctelor care reprezintă coordonatele locuințelor elfilor.

Datorită faptului că este posibil ca toate punctele să se afle pe această înfăşurătoare convexă vom utiliza scanarea *Graham*, deoarece potrivirea *Jarvis* ar putea avea ordinul de complexitate $O(N^2)$, iar limita de timp admisă pentru execuția programului va fi depăşită.

După determinarea înfăşurătorii convexe vom scrie în fişierul de ieşire punctele care o determină, în ordine trigonometrică.

Analiza complexității

Citirea datelor de intrare implică citirea celor N perechi de coordonate ale locuințelor elfilor, aşadar ordinul de complexitate al operației de citire a datelor de intrare este **O(N)**.

Pentru cele N puncte vom determina înfăşurătoarea convexă folosind scanarea *Graham*, operație al cărei ordin de complexitate este **O(N · log N)**.

Scrierea datelor în fişierul de ieşire se realizează în timp liniar deoarece vom scrie cel mult N numere. Aşadar, ordinul de complexitate al acestei operații este **O(N)**.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N) + O(N \cdot \log N) + O(N) =$ **O(N · log N)**.

18.7.2. Teritoriu

Problema se reduce la a determina înfăşurătoarea convexă a punctelor care reprezintă coordonatele oraşelor şi determinarea ariei poligonului obținut.

Datorită faptului că este posibil ca toate punctele să se afle pe această înfăşurătoare convexă vom utiliza scanarea *Graham*, deoarece potrivirea *Jarvis* ar putea avea ordinul de complexitate $O(N^2)$, iar limita de timp admisă pentru execuția programului va fi depăşită.

După determinarea înfăşurătorii convexe vom determina aria poligonului convex care reprezintă înfăşurătoarea. Pentru aceasta vom împărți poligonul în triunghiuri "trasând" toate diagonalele care pornesc dintr-un anumit punct. Vom obține astfel $N - 2$ triunghiuri a căror arie poate fi calculată folosind formula cunoscută. Vom însuma apoi ariile şi vom scrie rezultatul în fişierul de ieşire.

Analiza complexității

Citirea datelor de intrare implică citirea celor N perechi de coordonate ale oraşelor din *Erathia*, aşadar ordinul de complexitate al operației de citire a datelor de intrare este **O(N)**.

Pentru cele N puncte vom determina înfăşurătoarea convexă folosind scanarea Graham, operație al cărei ordin de complexitate este **O(N · log N)**.

Pentru poligonul obținut vom calcula ariile celor $N - 2$ triunghiuri obținute prin trasarea diagonalelor care pornesc dintr-un vârf. Ordinul de complexitate al acestei operații este **O(N)**.

Pe măsură ce ariile sunt determinate, acestea vor fi însumate, deci pentru a scrie rezultatul în fişierul de ieşire vom avea nevoie doar de un timp de ordinul $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N) + O(N \cdot \log N) + O(N) + O(1) =$ **O(N · log N)**.

18.7.3. Fred Flinstone

Această problemă se reduce la determinarea succesivă a unor înfăşurători convexe. Vom determina o înfăşurătoare pentru toate cele *N* puncte, apoi o alta pentru punctele rămase după eliminarea celor care se află pe prima înfăşurătoare şi vom continua până în momentul în care vor fi eliminate toate punctele.

Datorită faptului că toate cele *N* puncte se vor afla pe exact o înfăşurătoare, este recomandabil să utilizăm potrivirea *Jarvis*. Pentru a determina punctele de pe a *i*-a înfăşurătoare ordinul de complexitate va fi $O(N \cdot h_i)$, unde h_i este numărul de puncte de pe a *i*-a înfăşurătoare.

Pentru toate cele *K* înfăşurători vom avea ordinul de complexitate:

$$\sum_{i=1}^{K} O(N \cdot h_i) = O(N \cdot N) = O(N^2),$$

deoarece avem $h_1 + h_2 + ... + h_K = N$.

Dacă utilizăm scanarea *Graham*, există posibilitatea să obţinem ordinul de complexitate $O(N^2 \cdot \log N)$, deoarece este posibil ca fiecare înfăşurătoare să conţină doar trei puncte, deci putem avea până la $[N / 3] + 1$ înfăşurători şi pentru fiecare dintre acestea ordinul de complexitate este $O(N \cdot \log N)$.

Pe măsura determinării înfăşurătorilor vom scrie în fişierul de ieşire numerele de ordine ale punctelor care le determină.

Analiza complexității

Citirea datelor de intrare implică citirea celor *N* perechi de coordonate ale cuielor de pe tablă, aşadar ordinul de complexitate al operației de citire a datelor de intrare este **O(N)**.

Pentru cele *N* puncte vom determina o serie de înfăşurători convexe, operație al cărei ordin de complexitate este $\mathbf{O(N^2)}$, dacă se utilizează potrivirea *Jarvis*.

Metoda prin care a fost estimat acest ordin (în cadrul prezentării soluției) nu este corectă în întregime din punct de vedere logic. Practic, pentru a *i*-a înfăşurătoare numărul de puncte pentru care aceasta este determinată nu este *N*, ci $N - h_1 - h_2 - ... - h_{i-1}$.

Obținem astfel, pentru a *i*-a înfăşurătoare următoarea formulă pentru calculul ordinului de complexitate:

$$O\left(\left(N - \sum_{j=1}^{i-1} h_j\right) \cdot h_i\right).$$

Ordinul de complexitate al întregii operații ar deveni:

$$\sum_{i=1}^{K} O\left(\left(N - \sum_{j=1}^{i-1} h_j\right) \cdot h_i\right).$$

Cu toate acestea, dacă efectuăm calculele, rezultatul final va fi, în cazul cel mai defavorabil, tot $O(N^2)$.

Scrierea datelor în fişierul de ieşire se realizează pe parcursul determinării înfăşurătorilor convexe, deci nu se consumă timp suplimentar pentru această operaţie.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N) + O(N^2) = \mathbf{O(N^2)}$.