
! " #
)

\$

% " ! & # " ' % " ! & (# ' % " !



*
" ! ! " " ! & # " ' \$ \$! " " ! & (# ' #)

* # # ! # ! % # % + ! # %
, (# -)



! # , -% ! # , -)
+ % % ! %#)

" ! \$ " # \$ # . /
" " " # \$ # \$ \$ % ") # ! ! /
! , ! / ! " (-% % (! ! ! #
/ (#)
O" ! " \$! " # " . " # #
" " #)

1

! % # " ! + \$, ! "
-)





! " # \$)
234523678 . " # \$)
- " ! (# ! ! " # % # ' " " ! \$ # # \$ %
! " ! \$ # #)



[REDACTED]

9)

[REDACTED]

* # ! \$! % ! # , # -)

[REDACTED]



! , " " - ! ! " \$# , # -)



* ! (! " " ! " # , # -)



7 :
! & # " # '%&# ! ('%& '%&"
' . # " # ! / # ! # , "
" # -)



; # " # . " #
" !\$# # \$
" !\$# #
(" \$! \$



! (. " #
" !

[REDACTED]

< # . " #
! # !

[REDACTED]

[REDACTED]

* # . " #
" / " " \$ # =)

[REDACTED]

" # ! " # # / , " -)
>
? " # #! " ! (/ ! " #
" " #)

[REDACTED]

@ ! " " # (/)



* O,A- (# (/ " /B ! (\$ # #
/ % (" /B? ! (# # %# # (# (/
")
* \$! O,A- \$! !)
; # " (/ ! O,A- " # " # !
! " " # , ! (B " #" " # # (/ -)





2 C / " " / ! # ? (% #
()





2 ! " # ! \$ (/ ! ! \$ (/ " ! ! ")
! (/ " ! ! " ! (/ # ! " ! # " % !
(/ ! # " " #)
* # " ! (/ ! " # ! ? ! /! %
" " #)



* # ! " ! % " # " ! ! " ! & D ') ! % ! " #



* # " ! ! ! , ! ! - !) * # !
" ! !)



6 ! " ! !
" ! = #
" ! & (# ' #! " "
" ! & # / '
" ! (/ # # "
" ! " !
" ! & # '

< 9 % #)



9 7 9 #! ! " ! !)



Z o ono algorytmie

• Z o ono pami ciowa algorytmu

wynika z liczby i rozmiaru struktur danych wykorzystywanych w algorytmie.

• Z o ono czasowa algorytmu

wynika z liczby operacji elementarnych wykonywanych w trakcie przebiegu algorytmu.

Z O ONO CZASOWA ALGORYTMU - zale no pomi dzy rozmiarem danych wej ciowych a liczb operacji elementarnych wykonywanych w trakcie przebiegu algorytmu (podawana jako funkcja rozmiaru danych, kt ej warto ci podaj liczb operacji)

np.

a) w alg. sortowania b belkowego dla listy o d ugo ci N :

$F(N)$ = liczba porówna par s siednich elementów (?)

b) w alg. rozwiz ywania problemu wie Hanoi dla N kr k:

$F(N)$ = liczba przeniesie pojedynczego kr ka z ko ka na ko ek (?)

c) w alg. wyznaczania najd u szej przek tnej wielok ta wypuk ego o N wierzcho kach:

$F(N)$ = liczba porówna d ugo ci dwu odcinku przek tnych (?)

d) w alg. wyznaczania "najta szej sieci kolejowej" dla N w z i e sieci i M mo liwych do zbudowania odcinku:

$F(N, M)$ = liczba porówna kosztu budowy dwu odcinku (?).

W praktyce z o ono czasowa decyduje o przydatno ci algorytmu

Poniewa

• trzeba rozwiz ywa algorytmicznie coraz wi ksze zadania:

- w komputerowych systemach wspomagania decyzji,
- przy komputerowych symulacjach i prognozach z o onych zjawisk.

• rozwijane s komputerowe systemy czasu rzeczywistego:

- steruj ce automatycznie z o onymi uk adami (transport, produkcja)

Chcemy zmniejsza czas wykonania algorytmu!

Przyk ad 1

Normalizacja wektora (tablicy jednowymiarowej) wzgl dem warto ci maksymalnej; dane wej ciowe zapisane w $V(1), V(2), \dots, V(N)$

Algorytm 1

1. wyznacz w zmiennej MAX najwi ksz z warto ci ;
2. **dla I od 1 do N wykonuj :**
 - 2.1. $V(I) \sim V(I) / MAX$

Algorytm 2

1. wyznacz w zmiennej MAX najwi ksz z warto ci ;
2. $ILORAZ \sim 100 / MAX$;
3. **dla I od 1 do N wykonuj :**
 - 3.1. $V(I) \sim V(I) \cdot ILORAZ$

Je li operacj elementarn jest wyznaczenie warto ci iloczynu lub ilorazu dwu zmiennych,
to $F_1(N) = 2 \cdot N$ i $F_2(N) = N + 1$

Przyk ad 2

Wyszukiwanie liniowe elementu z listy o d ugo ci N ; dane wej ciowe to lista i element do wyszukania:

Algorytm 1

1. we pierwszy element listy ;
 2. **wykonuj:**
 - 2.1. sprawd czy bie cy element jest tym szukanym ;
 - 2.2. sprawd czy osi gn e koniec listy ;
 - 2.3. we nast pny element z listy
- a** znajdziesz szukany element lub przejrzysz ca list

Algorytm 2

1. dopisz szukany element na ko cu listy ;
2. we pierwszy element listy ;
3. **wykonuj:**
 - 3.1. sprawd czy bie cy element jest tym szukany ;
 - 3.2. we nast pny element z listy
- a znajdziesz ;
4. sprawd czy jeste na ko cu listy

Je li operacj elementarn jest sprawd , to $F_1(N) = 2 \cdot N$ i $F_2(N) = N + 1$

Poprawianie z o ono ci algorytmu jest czym wi cej ni tylko zmniejszaniem czasu jego wykonania!

Je eli np. porównujemy dwa algorytmy wykonuj ce to samo zadanie za pomoc jednej p tli ograniczonej, w kt ej liczba iteracji N jest wprost proporcjonalna do rozmiaru danych wej ciowych, to liczby operacji elementarnych (czas wykonania) tych algorytmów w f unkcji rozmiaru danych wynosz odpowiednio:

$$F_1(N) = K_1 + L_1 \cdot N$$

$$F_2(N) = K_2 + L_2 \cdot N$$

do ich porównania mo emy wykorzysta iloraz $s(N) = \frac{F_1(N)}{F_2(N)}$

i wtedy spe nienie warunków:

$s(N) = 1$ oznacza oby jednakow szybko dzia ania

$s(N) < 1$ oznacza oby, e 1. algorytm jest szybszy.

Ale zauwa my, e $s(N) = 1$ zachodzi tylko wtedy, kiedy $K_1 = K_2$ i $L_1 = L_2$, co oznacza, e oba algorytmy s praktycznie identyczne.

A co mamy powiedzie , kiedy $s(N) \neq 1$ dla $N \in N_0$, ale $s(N) < 1$ dla $N > N_0$.

Kt y algorytm jest lepszy?

Przyj to, e porównuj c algorytmy badamy warto ilorazu $s(N)$ dla bardzo du ych warto ci N , czyli formalnie dla $N \rightarrow \infty$

Zatem o wyniku porównania czasu dzia ania dwu algorytmów decyduje warto : $\lim_{N \rightarrow \infty} s(N) = \frac{L_1}{L_2}$

Je li zachodzi jednocze nie $F_1(N) = O(F_2(N))$ i $F_2(N) = O(F_1(N))$, to $F_1(N) = \Theta(F_2(N))$.
 Je li zachodzi $F_1(N) = \Theta(F_2(N))$, to tak e $F_1(N) = O(F_2(N))$.

• je li $C = 0$, to algorytm o czasie wykonania $F_1(N)$ ma ni szy rz d z o ono ci (ma leps z o ono) od algorytmu o czasie wyk. $F_2(N)$; oznaczamy to symbolicznie $F_1(N) \ll F_2(N)$

• algorytm o czasie wykonania $F(N)$ ma z o ono liniow , je li $\lim_{N \rightarrow \infty} \frac{F(N)}{N} = C$, dla $0 < C < \infty$,
 z o ono rz du N oznaczana jest symbolem $\Theta(N)$
 (ma on z o ono co najwy ej liniow , je li $C < \infty$, ozn. $O(N)$).

• algorytm o czasie wykonania $F(N)$ ma z o ono kwadratow , je li $\lim_{N \rightarrow \infty} \frac{F(N)}{N^2} = C$, dla $0 < C < \infty$,
 z o ono rz du N^2 oznaczana jest symbolem $\Theta(N^2)$
 (ma on z o ono co najwy ej kwadratow , je li $C < \infty$, ozn. $O(N^2)$;
 zauwa my jeszcze, e $N \ll N^2$).

• sytuacj , w kt ej zachodzi warunek $\lim_{N \rightarrow \infty} F(N) = C < \infty$ oznaczamy $F(N) = O(1)$.

Dopiero zmniejszenie rz du z o ono ci algorytmu jest istotnym ulepszeniem rozwizania problemu algorytmicznego!

Je eli algorytm wykonuje rz dn liczb operacji elementarnych w zale no ci od konkretnych danych wej ciowych (o tym samym rozmiarze) mo emy bada czas wykonania **w najgorszym przypadku** (czyli skupiaj c si na takich przypadkach dopuszczalnych danych wej ciowych, dla kt ych ta liczba jest najwi ksza) - tzw. *analiza najgorszego przypadku* lub *pesymistyczna*.

W przyk adach normalizacji wektora i wyszukiwania liniowego

• czas wykonania Algorytmu 1. wynosi $F_1(N) = 2N$: $F_1(N) = O(N)$

• czas wykonania Algorytmu 2. wynosi $F_2(N) = N + 1$: $F_2(N) = O(N)$

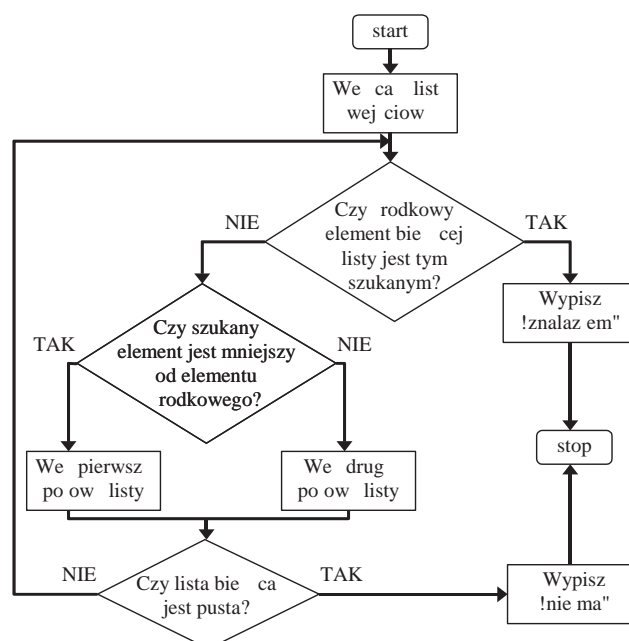
• oba maj pesymistyczny czas wykonania $O(N)$ - mog zdarzy si takie dane wej ciowe dla wyszukiwania liniowego, dla kt ych trzeba b dzie przejrze ca a list o d ugo ci N (jest tak, kiedy szukanego elementu nie ma w og le na li cie)

Czy mo na zaproponowa algorytm o lepszej z o ono ci dla wyszukiwania elementu na li cie?

Szukamy zatem algorytmu o (pesymistycznym) czasie wykonania $F(N) \ll N$.

Wyszukiwanie **binarne** (przez po owienie) elementu z listy uporz dkowanej:

Y_1, Y_2, \dots, Y_N (dla ka dego $i < j$ zachodzi $Y_i \leq Y_j$)



Je li przyjmiemy, e operacj elementarn jest porównanie elementu szukanego z jednym z elementó listy (rodkowym), to analiza z o ono ci polega na znalezieniu odpowiedzi na pytanie:
ile razy jest powtarzana w najgorszym przypadku iteracja w algorytmie?

Odpowied : $1 + \log_2 N$ ($\log_2 N$ b dziemy oznaczali $\lg N$)

Zatem pesymistyczna z o ono algorytmu wyszukiwania binarnego wynosi $O(\lg N)$

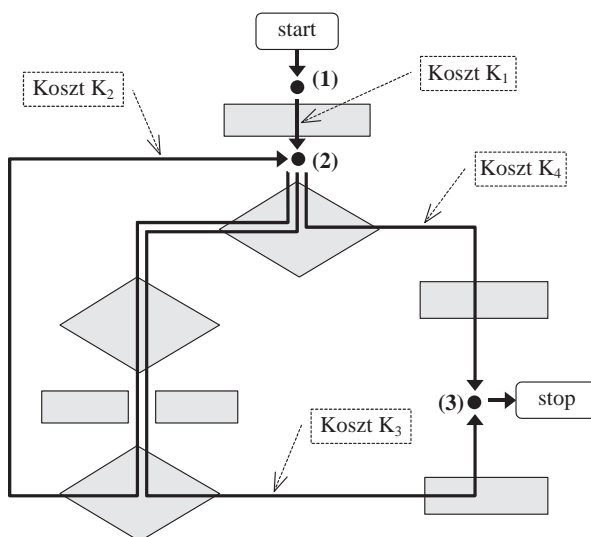
(mówimy, e algorytm ten ma z o ono logarytmiczn);

$\lg N$ **6** N

Skala ulepszenia:

N	$1 + \lg N$
10	4
100	7
1 000	10
10 000	14
1 000 000	20
1 000 000 000	30
1 000 000 000 000	40
1 000 000 000 000 000	50
1 000 000 000 000 000 000	60

O z o ono ci czasowej algorytmu decyduje liczba wykonywanych iteracji



Ca kowity koszt czasowy w najgorszym przypadku:

$$F(N) = K_1 + \max(K_3, K_4) + K_2 \lg N, \text{ a to oznacza } F(N) = O(\lg N)$$

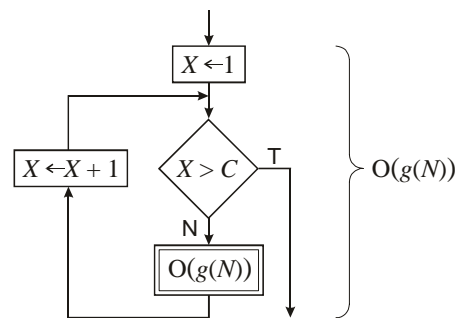
W a ciwo ci notacji $O(\cdot)$ (tzw. rachunek $O(\cdot)$)

- je li $F(N)$ jest funkcj z o ono ci algorytmu, to spe nienie warunku $\lim_{N \rightarrow \infty} \frac{F(N)}{g(N)} = C < \infty$ jest zapisywane

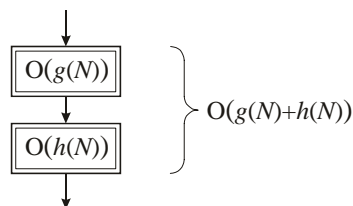
$F(N) = O(g(N))$; odczytujemy "algorytm ma z o ono rz du nie wy szego ni $g(N)$ "
lub krócej "z o ono algorytmu jest $O(g(N))$ "

- róno w zapisie $F(N) = O(g(N))$ powinna by rozumiana w ten sposób e funkcja $F(N)$ jest jedn z funkcji, które spe niaj powy szy warunek lub precyzyjniej, e funkcja $F(N)$ nale y do zbioru wszystkich funkcji spe niaj cych powy szy warunek
- $F(N) = O(F(N))$,
- $O(O(g(N))) = O(g(N))$,

- $C O(g(N)) = O(C g(N)) = O(g(N))$ (dla dowolnego $0 < C < \infty$),



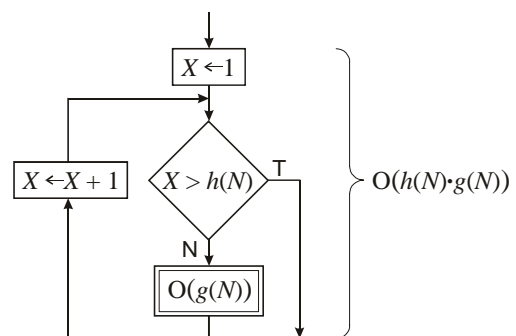
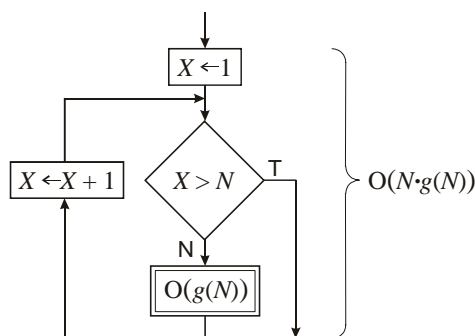
- $O(g(N)) + O(h(N)) = O(g(N) + h(N))$,



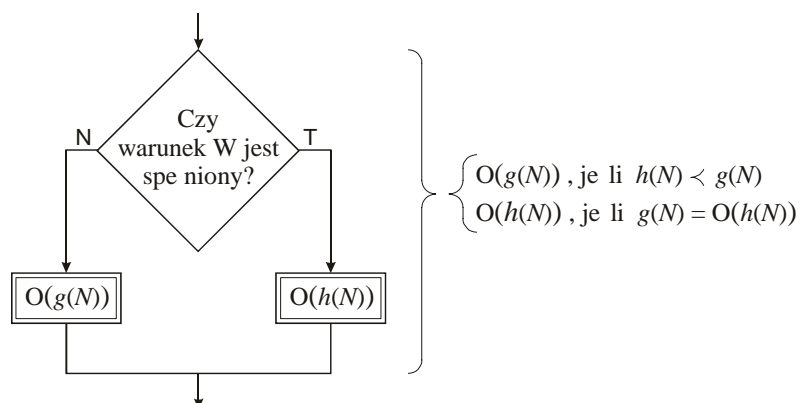
- je li zachodzi $\lim_{N \rightarrow \infty} \frac{g(N)}{h(N)} = 0$, czyli $g(N) = o(h(N))$,

to $O(g(N)) + O(h(N)) = O(g(N) + h(N)) = O(h(N))$

- $O(g(N)) \cdot O(h(N)) = O(g(N) \cdot h(N)) = g(N) \cdot O(h(N)) = h(N) \cdot O(g(N))$,



- przy analizie z o ono ci w najgorszym przypadku (dla warunku zależnego od danych wejściowych) zachodzi:



Z o ono czasowa przyk adowych algorytm

sortowanie b belkowe w pierwotnej wersji (zagnie d one iteracje):

1. wykonuj co nast puje $N - 1$ razy:
 - 1.1. ... ;
 - 1.2. wykonuj co nast puje $N - 1$ razy:
 - 1.2.1. ... ;

Ca kowity koszt czasowy w najgorszym przypadku wynosi z dok adno ci do sta ych:

$$(N - 1) \cdot (N - 1) = N^2 - 2N + 1; \quad 1 \cdot 6 \cdot 2N \cdot 6 \cdot N^2, \text{ czyli z o ono jest } O(N^2)$$

sortowanie b belkowe ulepszone (coraz kr zsze przebiegi):

Ca kowity koszt czasowy w najgorszym przypadku wynosi:

$$(N - 1) + (N - 2) + (N - 3) + \dots + 2 + 1 = 0,5 N^2 \approx 0,5 N, \text{ czyli tak e z o ono jest } O(N^2)$$

sumowanie zarobk

znajdowanie najwi kszej przek tnej w wielok cie wypuk ym metod naiwn - z o ono $O(N^2)$

znajdowanie najwi kszej przek tnej w wielok cie wypuk ym metod jednokrotnego obiegu z par prostych r

rekurencyjny algorytm dla problemu wie Hanoi

procedura **przenie** N ;

1. je li $N = 1$, to wypisz ruch i koniec;
2. w przeciwnym razie (tj. je li $N > 1$) wykonaj co nast puje:
 - 2.1. wywo aj **przenie** $N - 1$;
 - 2.2. wypisz ruch;
 - 2.3. wywo aj **przenie** $N - 1$;

Oznaczmy nieznany koszt czasowy przez $T(N)$ i u kmy r

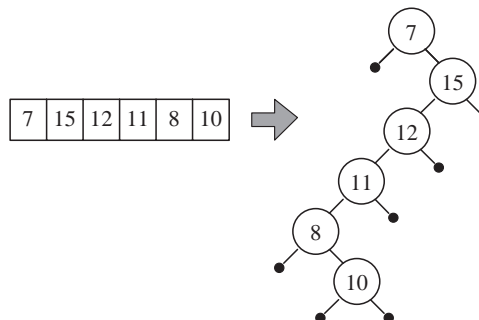
$$T(1) = 1$$

$$T(N) = 2 \cdot T(N - 1) + 1$$

Spe nia je koszt $T(N) = 2^N - 1$, czyli z o ono jest $O(2^N)$

sortowanie drzewiaste bez samoorganizacji drzewa:

ma pesymistyczn (w najgorszym przypadku) z o ono $O(N^2)$, bo wprowadzie lewostronne obej cie drzewa ma z o ono liniow $O(N)$, ale konstrukcja binarnego drzewa poszukiwa ma pesymistyczn z o ono kwadratow



sortowanie drzewiaste z samoorganizacj drzewa:

ma z o ono $O(N \lg N)$, co daje wyra n popraw sprawno ci algorytmu sortowania w por

N	N^2	$N \lg N$
10	100	33
100	10 000	664
1 000	1 000 000	9 965
1 000 000	1 000 000 000 000	19 931 568
1 000 000 000	1 000 000 000 000 000 000	29 897 352 853

sortowanie przez scalanie (rekurencyjne):

procedura **sortuj-list** L ;

1. je li L zawiera tylko jeden element, to jest posortowana;
2. w przeciwnym razie wykonaj co nast puje:
 - 2.1. ... ;
 - 2.2. wywo aj **sortuj-list** po owa L ;
 - 2.3. wywo aj **sortuj-list** po owa L ;
 - 2.4. scal posortowane po owy listy L w jedn posortowan list ;

Oznaczmy nieznany koszt czasowy przez $T(N)$ i u 1/2 my rekurencyjne, kt e $T(N)$ musi spe nia :

$$T(1) = 0$$

$$T(N) = 2 T(0,5 N) + N$$

Spe nia je koszt $T(N) = N \lg N$, czyli z o ono jest $O(N \lg N)$

Sortowanie przez scalanie jest jednym z najlepszych algorytm e sortowania (cho ma nie najlepsz z o ono pami ciow $O(N)$)

rednia z o ono

Analiza najgorszego przypadku **lub** analiza redniego przypadku

W analizie redniego przypadku istotn rol odgrywaj za o enia o rozk adzie prawdopodobie stwa w zbiorze dopuszczalnych danych wej ciowych.

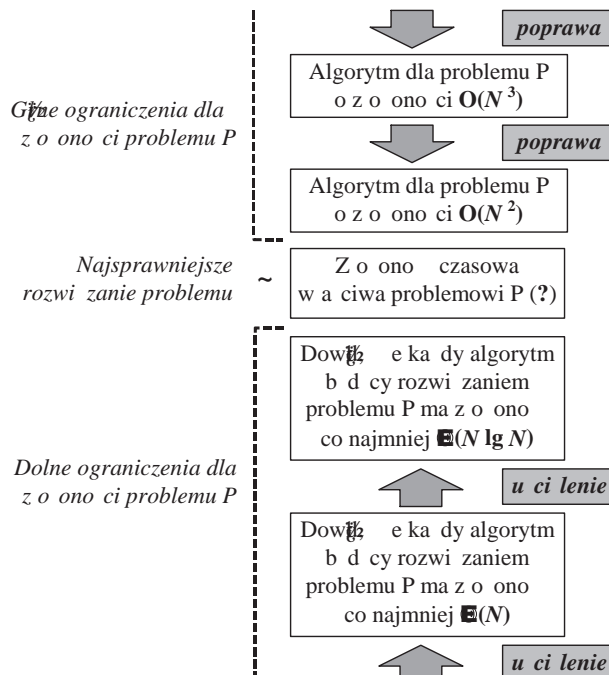
algorytm	redni przyp.	najgorszy przyp.
sumowanie zarbk e	$O(N)$	$O(N)$
sortowanie b belkowe	$O(N^2)$	$O(N^2)$
sortowanie drzewiaste z samoorganizacj drzewa	$O(N \lg N)$	$O(N \lg N)$
sortowanie przez scalanie	$O(N \lg N)$	$O(N \lg N)$
Quicksort	$O(N \lg N)$	$O(N^2)$

redni koszt czasowy algorytmu Quicksort wynosi tylko $1,4 N \lg N$

Dolne i g e ograniczenia z o ono ci problem e algorytmicznych

Czy mo na skonstruowa jeszcze lepszy algorytm?

- z o ono poprawnego algorytmu znajduj cego rozwiz anie danego problemu ustanawia **g e ograniczenie** z o ono ci dla tego problemu.
- **dolne ograniczenie** z o ono ci problemu (otrzymane w wyniku analizy samego problemu) okre la zakres dalszej poprawy rz du z o ono ci algorytm e rozwiz uj cych ten problem



Problemy zamkni te i luki algorytmiczne

problem	dolne ogr.	g e ogr.
---------	------------	----------

przeszukiwanie listy nieuporządkowanej	$\Theta(N)$	$O(N)$
przeszukiwanie listy uporządkowanej	$\Theta(\lg N)$	$O(\lg N)$
sortowanie	$\Theta(N \lg N)$	$O(N \lg N)$
wyznaczanie najtańszej sieci kolejowej	$\Theta(N)$	$O(f(N) N)^{1)}$

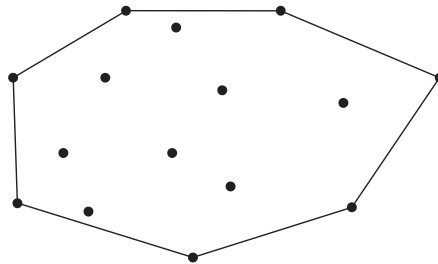
¹⁾ $f(N)$ - bardzo wolno rosnąca funkcja, np. dla $N = 64\ 000$ ma wartość 4

☐ - problem zamknięty

☐ - problem z luką algorytm.

Przykład analizy złożoności algorytmu

Problem wyznaczania powłoki wypukłej dla zbioru N punktów na płaszczyźnie:

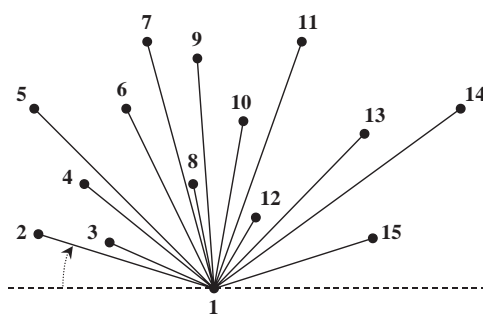


• algorytm "naiwny" ma złożoność $O(N^3)$;
trzeba dla każdego z N punktów dobrać pozostałe $N-1$ do pary, która wyznacza prostą i dla każdej z tych prostych sprawdzić czy $N-2$ punkty leżą wszystkie po tej samej stronie prostej.

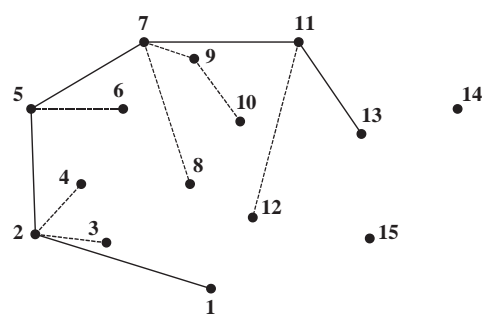
• algorytm o niżej złożoności (?):

1. znajdź punkt "najniższy" po ośn - P_1 ;
2. posortuj pozostałe punkty rosnąco według kątów tworzonych przez odcinki $\overline{P_1 P_j}$ z linią poziomą przechodzącą przez P_1 - powstanie lista P_2, \dots, P_N ;
3. dołącz do powłoki punkty P_1 i P_2 ;
4. **dla J od 3 do N wykonuj**;
 - 4.1. dołącz do powłoki punkt P_J ;
 - 4.2. cofaj się wstecz po odcinkach aktualnej powłoki, usuwaj z niej te punkty P_K , dla których prosta przechodząca przez P_K i P_{K-1} przecina odcinek $\overline{P_1 P_J}$, a do napotkania pierwszego punktu nie dajęcego się usunąć;

krok 2.



kolejne iteracje w kroku 4.



Podsumowanie złożoności algorytmu krok po kroku:

Krok 1.	$O(N)$	(wybór pierwszego)
Krok 2.	$O(N \lg N)$	(sortowanie)
Krok 3.	$O(1)$	(dołączenie 1 odcinka)
Krok 4.	$O(N)$	(dołączanie z usuwaniem)
Razem	$O(N + N \lg N + 1 + N) = O(N \lg N)$	

! " # \$ % #
& ' * " + " % (\$ %
) * . + " \$ %
, -
/ # # ' # " ' # 0 # " 0 %
' # . ' # " ' # 1 # #
2 % # # ' 1 #
(3 4 # 4
! (3 4 # # 5 4 4
! # # % #
& . # 4 \$ " # % &
! " # \$ " # % &
% ' & * " + , -
() . / * . , -
2 % # 3 # ' # 4 4
! . # 3 4 # 4
& 1 3 4 ' # 4
4 # 4
) 1 # 4 # 4
, 1 # 4 + , "
0 # 1
2 # 1 1 # 1 # ' # % "
! &) , # 1 ' # ' /
1 ! # 1 ' 7 8 %
6 # 1 ' 7 8 % 9 . : ;
1 # . 1 # % 1
- % 0
3 ' # # 0
3 # < # "
2 0
3 # 7 % " 8
3 % 7 % # " 0 # = " 8
3 > 7 ' 3 8 #
? # # 0 # / -- 7 / @ ! A = - =
? % ' 3
- 0 ' 8 2' . # @
0
3 B 0 0 # . " % # 1 0
' #

! # # %
1 0 # ' # " " 0 .

& # 3 # ' # " " 0 %
1 0 # ' # " " 0 .
) ; # # %
1 0 ' # " " 0 .
4 * * .
C ' # # ' # . . -
5) " " "
B 0 0 " 0 ' % " 0 " "
0 # 0 0 7 0 0 0 8 # 0
! 0 > 0 7 0 # # 0 # 0
1 (-D8
6 " % " # #
8 # . # # #
> % ' #
8 # . # # #
> % ' # #
8 # # # # >
" 7 # ! 8 # >
" "
? # 7 . E-EC ' = = 8 # #
' 0 . # 0 = C # 7 8
5 . 2 0 # 1 % # "
% % "
/ 7 . D-EC ' # = 8 # #
' 0 . # 0 = 9 ' # " % 7 8
% 7 # 8 # " # "
% # " . # % .
"
D % # % 0 # " ? .
% % # " # # 0 0
> 0 7 # # > # ! # > 8 #
0 = # > 8
" ' 7 8 ' 7 8
(! " "
6 . / ! 7 8
0
8 # . % # # #
> %
8 # . % # # # # >
8 # . %
8 # # " # # (-D
% % % ' 9 %
% % % ' 7F ' # 8
8 # #
8 #
1 "
B % ' 0 # " 3 0 % ? . %
3 # # 7 8
3 # 7 8
3 # > 7 7 # " ' 8 08
3 # > 7 # ' 8
C # '

5 # > 7 ' = 8
! 5 % 7 ' = ' " 8
& % # # . # 1
) # # # . 7 0 #
> 0 1 (-D8
% # ' %
9 # 0 0 ' # # . ' % 0
" # ' % 0
9 " " # 0 # ' " 5
9 #
/; " ' " ' ! # "
GH(G " 0 % # 1 # "

2 9
B " ' %' ' ! ? . % .
1 # ! "
3
? 5 0 % 7 ' = ' " 8 3
9 # 0 '
D 1 0 % 5 7 ' = # 8 3
C " 5 # 5 #
4 : * ;
<=>
% 1 # 5 (0 1
" 1 ' ' # . # .
3 # # ' #
3 # # ' #
5) 7 ;
B
C % #' ' # ' #
C GFBI ;
; "1
! % > ; #

% > ;
2 ' # #' ' 1 . > 3 %
> 3 # 1 > 3
9 /; # #
J 1 . ' # ' #
2 # # ' . ' = # 0 % 5
0 " ' / ;
' # K ' #
/ 3 7 # 1 8
F # ' 0 # " #
0 # " 8
3 ' # ' % 7 8
/ 0 % #
9 5 7 # ' 8

! # ' # 0 % # ' * + # ' # 0 % # ' * + # ' 9 ' # 7 8 # ' * + 3 / 0 % # ' 8 # ' * + ! 9 # ' * + 0 ' & # ' 7 * +8 # ' * + 9 ' * + # ' * + + J # ' 0 # 0 # " ' " 0 # # * ; G# % # 7 # = ' ' # 0 " # . 1 3 # 7 # 0 # = % 5 ' ' 8 3 % # 7 " ' " # # 0 0 " 8 3 % 7 0 0 0 1 0 4 1 = ' " # 0 4 0 " 0 8 * + # * 3 7 " 0 " 0 0 58 ? # ' 5 ' F % . % ' ' 0 # 0 LL # ' 0 0 % ' " # 0 ; # ' ' # # ' " # 0 (! % ; & 0 J # # ' * + # ' ' # 0 # # % (+ # # # ' % 0 # 0 0 % ' # # # ' % 0 # # ' # - # # ' # # # . 0 " 0 1 3 3 ' . . 3 3 % ; ' & 1 J * # . + ' # # . ; 1 % # ' " 1 # 1 ' # # * # . + # ' # (# 3 # # % " % # 0 # " " # " # 0 # " # 0 # # 0 # # # # C % ' # # /C6; MF D- /; N D # # ! # # D D! % /C6; MF D

% /C6; MF D!
D D!
& "1
1!
- % " 0 . 1 % "*" & # " 0
% 5 ' 0% ' ' + F = #
' # " * 0% + #' 5 ' = 8
J 1 " % 0 " " . 1 5 0 #
% 5 0 0 0 % > " 1 0 #
C #'
! 3 5 % % " 0 " % > "
J % ' 5 ' % 5 0
" 0 " % > 7 # 5 8 (' % 0
% > " * # ' # " #
/ # " 7 ') # 8 0 # ' 8 " " ! # 0
7 # # ' #' # 9 " " 0
(0 7 # " 0 ' " 8 % 0 0 % ! # " " 0
0 " ' . . " 0 0 # " " .
" #
C % #' /C6; MF # D4

! # # D D D!
% /C6; MF # D
% /C6; MF # D!
D D! # D 7 8
& "1
3! % + &
J #' " " * # # ' = ' ' # 5 ' +
5 ' 5 # ' ' = ' ' ' 5 ' +
% % * % > + #' 7 # " ' . 8
9 0 # # # " #1 0% 0 0 # " " #
" " = ' ' = # 5 0 # = 5 # ' =# ' * +
' . ' = ' 5 0 .# 0 " " 0
4 / " *;
(% " % " 7 % 8
% . ' .

3 . = 7 K@ O LL8
3 = 7 K@4 O 4 LL8

?
3 # # #
3 # # % %
3 # # %
%
.

3 # . 5 = 7 K@4 O 4 LL8 1 L
?
3 ' # #' "
3 % # . 0 1 '
% # ' 0
% # ' #' % ' ,

3 # . #' # % * + #
* * F . 1 # . 1
5

?
3 #' 1
3 # 0 0 0 '
3 . 1 1 0
3 % ' > ' % ' 1 1
% #'
9 % ' #'

3 #' ># ' = % # "
0

?
3 #' # 0 # 0 0
3 '
3 # #' 5 ' %
5 ' ; * \$; .
' % 5 ' . # # . 0
0 % 0 # # 5 5 % 5
0 # 5 " 5
' # 5 ' #
6" # ' #
5 #'
9 = 0 0 0 1 #' 1 . # 5 # 0
9

(" . #'
7J " # ' " # 0
F # . # " ' 0 # ' ' "
' . ' 0 % ' #
' # 1 8 " # ' #
2 . " # % " 7J % " # #
(# " 8
" # ' 1 " # " 8
* ' . / 1
2% . 1 #' " . 1
3
3
2% . 1 # . 1 # % #' 0 #'
7 # # 08 0 0 0 #'

2% . 1 # . 1 # # 0 0
' # ' # 0 0 #
% . ' . #
@ \$ ' * ' . /
B # 0 E 7(8 E! 7(8 % . 1 ' ' #

$$\lim_N \frac{F_1(N)}{F_2(N)} = C$$

F # K @ #' E 7(8 % .
F # K #' E! 7(8 % .
F # @O O % . 1
(: @ ; * ' .
C % . # ' C7# ' (8
E7(8 % . 1 # ' . # # E7(8P(@
O : @ ; * ' .
C % . # E7(8KC7(8
E7(8 % . 1 # . # # E7(8P(K ' @O O
1 : @ ; * ' .
C % . E7(8KC7(!8
E7(8 % . 1 . # # E7(8P(!K ' @O O
2 : @ ; * ' . *
C % . % E7(8KC7! (8
E7(8 % . 1 % . # # E7(8P! (K ' @O O
3 9 A+! , B?+ +! , , @ A+! ,
2 E7(8KC7' 7(88 % # E7(8P' 7(8K ' @O O 1
% . 1 ' 7(8 # # C7' 7(88
6" 1 E7(8KC7' 7(88 1 " . = E7(8
= % " % # . # . 0 =
% 0 .
4 ? " * . /
B % # ' # " F . #
= 8 9 # " % # ' 1 7 #
9 ' # F # " = # %
9 # = # 0 . '
2% . 1 ' #' E7(8KC7# ' (8 ' (# . # # "
B% ' 1 #
3! @ # " 7, " 58
3 @ @ @ @ @ # " 7! @ " 58
(5 " * ' . /
9 # '
3 ' C7(!8
3 7# 8 C7(8
3 # C7(!8
3 ' C7(Q# ' (8
3 7 # 8 C7(Q# ' (8
3 R C7(Q# ' (8 3 C7(!8
(9 ' ; * ' .
9 # ' ' % . 1 #'
" ' 0 0 7 ' # # 0 #'
9 # ' 1 % # 9 # 0
' % # ' # 1 % #' 5 9 # 0
0 0 0 % . %

F . # 1 . ' ' 1 #
, # % . 1 ' . 1 ' #
(# ; " 0 # " ' % .
0 ' # 2 ' " ' % .
% . ' # ' ' % .
F # # ' " 1 0 " " #
0
(: * ; " * ' .
% # " 0 % . #
3 " 3 # ' " ' % . C7(8
3 # # ' " ' % . C7# ' (8
% # # 0 ' " ' % . C7(!8 #
3 # # ' ' " ' % . C7(8
C7(8 ((* ' ./ ' C
9 . : # ' " ' % . " 7 " #
' ' 8 C7(!8
(0 9 \$ ' @ * ' .
C = . % . 1 . = ' " ' .
3 # ' #
D (1 9 \$ ' * ' ./ . # = ' % . ' " (?
' ? # E7(8KC7(?8
(2 ; ! : - * " : *
?# # " (% 0 # 0 0 0
3 . # # % 1 ' .
3 # 0 # 7 % 8
3 # ' # 7 # ' 8
3 # ' ' % ' .
3 # 7 # # 0 1 ' # >1 #
. # 0 1 ' # 1 # 0 . '
8
% : # " (% 0
3 . : # 7 % % . = ' ' 8
3 % #
3 #
3 % #
(3 \$ ' ! : - * ' .
/ * ' .
S . 0 1 # " (% 0
% # # ' . % 1
9 # . # # # # % 1
(4 * ! : - *
* ' . " # (% #
S . 0 1 " # . 9 = . .
% 1 # " .
05 !
S% ' 0 # " % . # 0
' # # % ' # ;
* + # # " 0
% # 3 # 7(% 8 # . 0 % ' =
1 % % ' # 0
J ' "% / K P #

9 . # # . / , % . 1 .
0 # C7(8
" ' . % ? *
F # # #
% # ' # #
' # * + # * +
0 1 . # 0 #
1 > # % . #
B# . ' #' 7 ' # = 1
" 8 " 0 # " 0 #' % % 5 1 #
% >
0 9 9
1 0 > * + # * +
#' 6 # D " ' # 0 # O T F # " .
= % 1 > * .
0 9 > \$ * .
J ; ' # # # = '
0 = 1 #' "
J ; ' %
3 5 ' #=
3 5 ' "
3 5 # "
7 . " 1 #8
3 0 ' %
7' % " 8
3 ' 1
7 ' 18
9 # . ; ' 1
3 ' # %
3 #
3 ' % =
O(> ' 1 " ' =
/ " ' 0 % ' # 0 " ? # >1 B
" . ' # 0 " ? > 7 P 8 ' U
1 # # P
7 P # 8 # = * + * + > # #
7 P 8 # = * + * + > #
00 ? 9" " D> * ' .
" "
J ; ' = 1 . = # # " =
6 1 # C7=7(88 " . C7 7=7(888 # # =

01 9
5 ' ; ' 7 % ; ' 8
7 % # 8 # 7 3 " 8 ' %
6" # ' . % 1
% . # 7 1 " 0
/ ' % " 1 " # % 8
' 1 0 B ' " .
' 5 " 0 # " O V # 0 0

02 9 ' * ' / " # 5 = # 1 9 1
' ' 5 . 1 # 5 7 " 1 0
' 0 . % 8 = 7 ' % " 1 0
08 .# 1 #
' 1 * "
03 1 #
3 # #
3 #
(' 1 # .# # ' #
' % . 1 % 5 # " " " #
/ # = ' % 5
3 . " 1 % . % # 7 # . 8 #
" 0
3 " # ' % ' " #
#'
B % # ' 7 # 8 #
= # ' 0 7 (B C6 (C; ((B 8 " " * ' +
04 . " ; " @ " @ , " ' #
2 " 0 0 " 2 . 1
' 1 # # % ' # * 0 # ' + F #
' 0 ' 1 9 % * 0 5 0 = # = " +
% B# (= # = " # (P! L % . = # = " " 0
#1
15 . ; " " * ' 0
2 " 0 0 "% . " # 0
" " " ' . ' 0 # 0 % %
' % * 0 0 # ' + ' 0 # 0 ' %
' 0 # 0 0 1 ' 0 0 ' #
" 9 # = 0 # "' %
0 ' . 0 #
1 1 ; * " " * ' ' %
2 ' % " "% . " " " %
" ' % * 0 # ' + ' ' %
0 " 0 # 0 1 # =
1 . ; " " * ' ,
' %
% #
' ' * 0 # ' + % ' % 0
5 " 0 # # %
9 * 0 5 0 = # = " + % ' % = # =
0 # " 0 %