DOLNE GRANICE

Problem (sortowanie)

Dane: $a_1, \dots, a_n \in \mathbb{R}$

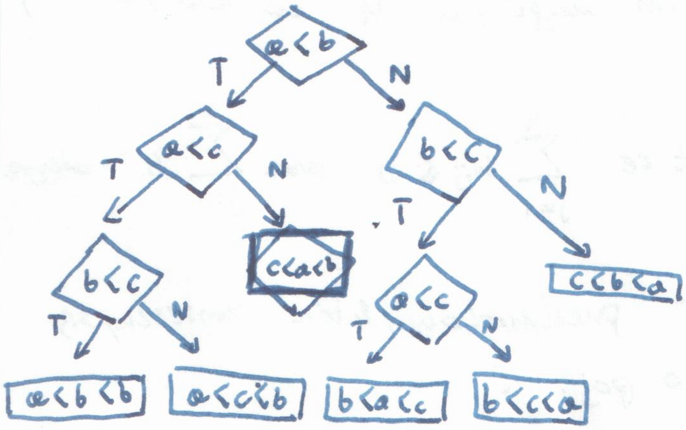
Wynik: posortowane $a_1 \dots a_n$

Model

Drzewa decyzyjne

Przykład select sort:

Dane a, b, c



← za pomocą takich drzew możemy reprezentować działanie programu dla ustalonego rozmiaru danych

Zatem dla każdego n rozmiarowych danych mamy inne drzewo D_n. Niech $\{D_n\}_{n=1}^{\infty}$ to rodzina drzew decyzyjnych odpowiadających pewnemu algorytmowi A. (D_i – drzewo dec. algorytmu A działającego na danych rozmiaru n)

FAKT

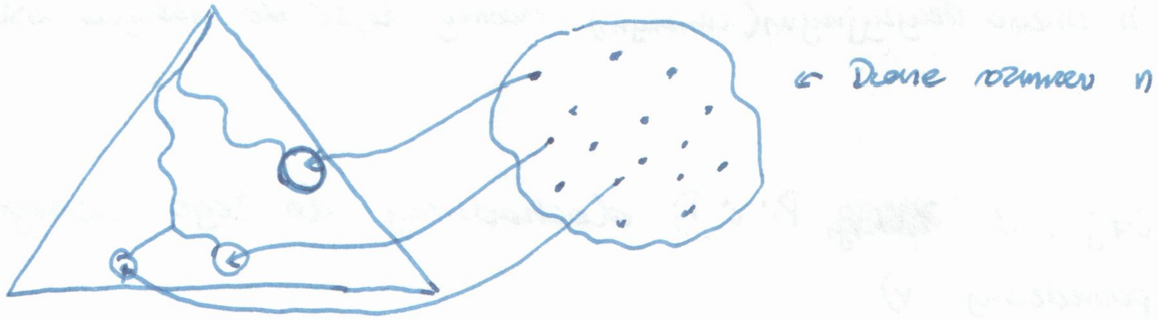
Jeżeli A – algorytm sortujący oraz $\{D_i\}_{i=1}^{\infty}$ – rodzina odpowiadających mu drzew decyzyjnych, to D_n ma conajmniej $n!$ liści.

WNIOSEK

Jeśli A: $\{D_i\}_{i=1}^{\infty}$ – jak w Fakcie, to D_n musi mieć wysokość

$$\Omega(\log n!) = \Omega(n \log n)$$

Jak to jest w przypadku złożoności średniego przypadku?



Trzeba by znać rozkład prawdopodobieństwa danych

Problem (ELEMENT UNIQUENESS)

Dane: $x_1, \dots, x_n \in \mathbb{R}$

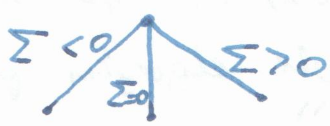
Wynik: TAK if $\forall_{i \neq j}\ x_i \neq x_j$

NIE w.p.p.

Model: drzewo decyzyjne

Pomysł:
Popatrzmy na dane jak na punkty w $\mathbb{R}^n$

Troszeńkę wzmocnimy model: liniowe drzewa decyzyjne. Teraz w wierzchołkach są pytania postaci czy $\sum_{i=1}^{n} c_i x_i$ jest większe mniejsze lub równe zero, gdzie $c_1, \dots, c_n$ - stałe, a $x_1 \dots x_n$ - dane



Niech $S(v)$ - zbiór punktów, z którymi wędrujemy w drzewie do wierzchołka v

Oznaczenie

$S(v)$ - są obszarami wypukłymi (naprawdę interesuje nas, że są one spójne)

Niech $P_0 = \langle 1, 2, \dots, n \rangle$

$n!$ punktów $\Big\langle$ P_i - ita permutacja ciągu $1, \dots, n$

FAKT

$\forall_{\substack{P_i, P_j \\ i \neq j}}$ w drzewie decyzyjnym (liniowym) musimy dojść do różnych liści

D-d a.a.

Niech $i \neq j$. Z P_i i P_j dochodzimy do tego samego liścia (powiedzmy v).

Wiemy, że $S(v)$ jest zbiorem spójnym, więc P_i oraz P_j można połączyć krzywą (nawet odcinkiem) całkowicie zawartym w $S(v)$

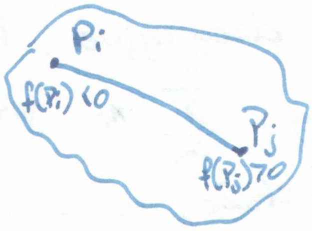
$P_i : \; x_1 \,.\, k \,.\, \odot \; x_n$ ← r-ta współrzędna

$P_j : \; x_1' \,.\, \odot \,.\, k \; x_n'$ ← s-ta współrzędna

perm $1 \ldots n$

Niech k - min takie że, k jest na innej pozycji w P_i i P_j. Zatem w $\odot$ są wartości większe od k. Niech $f(y_1 \ldots y_n) = y_r - y_s$. Widzimy, że $f(P_i) < 0$, a dla $f(P_j) > 0$

Czyli w naszym drzewie:



Wartość funkcji f zmienia się w sposób ciągły, tzn. że na tej prostej / tym odcinku jest punkt, dla którego $f(P_x) = 0$, czyli ma dwie współrzędne równe, czyli algorytm powinien dać odpowiedź NIE, a daje TAK (bo $P_x \in S(v)$)

Problem (MIN-MAX)

Dane - $S = \{a_1, \ldots, a_n\}$

Wynik - $m = \min(S)$

$M = \max(S)$

Q: mogę brać udział wyłącznie w porównaniach

Istnieje algorytm dla problemu $\lceil \frac{3}{2} n - 2 \rceil$ porównań

I $a_1, a_2, a_3, a_4, \dots, a_n = 2k$

$S_{MIN} = \{ \ , \ , \dots \}$ k porównań $\rightarrow$ $3k - 2$

$S_{MAX} = \{ \ , \ , \dots \}$

II Znajdź min w S_{min} $\leftarrow k-1$

Znajdź max w S_{max} $\leftarrow k-1$

2-ch gracy : algorytm : adversary

Początkowo:

Cel gry:

- Alg : wskazanie i oraz j takich że $a_i = \min S$, $a_j = \max S$

- Adv : zmuszenie algorytmu do zadania $\geq \lceil \frac{3}{2} n - 2 \rceil$ odp

Ruchy:

- Alg : zadaje pytanie $a_k = a_r$
- Adv : odpowiedź na to pytanie

Koniec gry:

- Alg : podaje indeksy i, j
- Adv : odsłania dane

W trakcie gry Adv pamięta zbiory:

A = {a_i : nie brały udziału w porównaniach }

B = {b_i : brały udział w porównaniach i wszystkie wygrały }

C = {c_i : brały udział w porównaniach i wszystkie przegrały }

D = pozostałe

Początkowo

$|A| = n \quad , |B| = |C| = |D| = 0$

Na koniec

$|A| = 0 \quad , \quad |C| = |B| = 1 \, , \quad |D| = n-2$

Strategia Adwersarza:

(i) ustala początkowe wartości dla $a_1 \dots a_n$

(ii) w razie potrzeby będzie te wartości zmieniać, ale tak by dotychczasowe odpowiedzi były prawdziwe

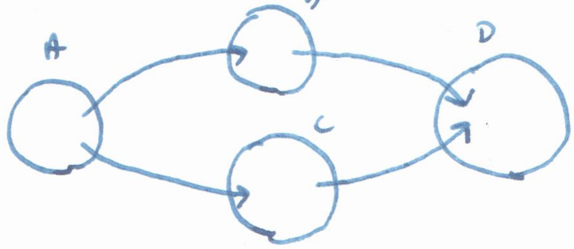
(iii) elementy $a_1 \dots a_n$ będą umieszczane w zbiorach A, B, C, D

Ad iii

Adwersarz będzie utrzymywał niezmiennik

$\forall_{a \in A} \forall_{b \in B} \forall_{c \in C} \forall_{d \in D} \quad b > a > c \;\&\; b > d > c$

typ porównania	$\|A\|$	$\|B\|$	$\|C\|$	$\|D\|$
A A	-2	+1	+1	0
A B	-1	0	+1	0
A C	-1	+1	0	0
A D	-1	+1		
			+1	
B B	~~[illegible]~~ 0	~~[illegible]~~ -1	~~[illegible]~~ 0	+1
B C	0	0	0	0
B D	0	0	0	0
C C	0	0	-1	+1
C D	0	0	0	0
D D	0	0	0	0



Aby elementy opuściły zbiór A potrzeba $\lceil \frac{n}{2} \rceil$ porównań

Aby $n-2$ elementów trafiło do D potrzeba $n-2$ porównań

DOLNE GRANICE c.d.

- redukcja

Problem:

A - znamy jego dolną granicę

B - potrafimy problem A zredukować do problemu B

$$x \xrightarrow{f} y = f(x)$$

dane dla A

dane dla problemu B

Przykład lipnej redukcji (po to, aby umieć!!!)

(Sort) Sortowanie - dolna granica $\Omega(n \log n)$

(CH) Otoczka wypukła

Problem:

Dane:

$p_1 = (x_1, y_1) \dots p_n = (x_n, y_n) \in \mathbb{R}^2$

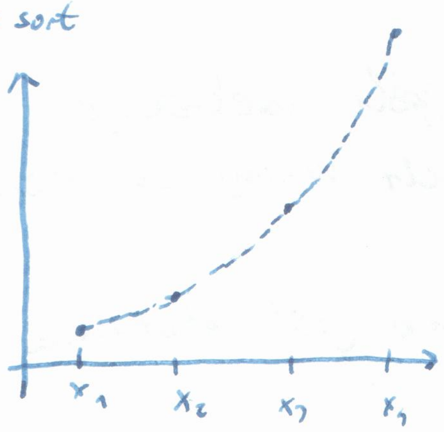
Wynik: Wielokąt wypukły o najmniejszym polu zawierający (w węzłach lub dowolnie) wszystkie $p_1 \dots p_n$

Sort < CH

$$x_1, x_2, \dots, x_n \longrightarrow (x_1, x_1^2) \dots (x_n, x_n^2)$$

dane dla sort

dane dla CH



Teraz gdybyśmy umieli zrobić otoczkę np liniowo, to umiemy posortować liniowo, bo posortowane elem będą na porządku. Pytanie, gdzie jest pęknięcie?

Oszustwo jest takie, że w modelu drzew decyzyjnych możemy wykonywać wyłącznie porównania. Chodzi o pewną ostrożność, aby uważać, czy aby na pewno wciąż jesteśmy w modelu i możemy korzystać z własności tego modelu. My nie pokazaliśmy, że sortowanie wymaga nlogn operacji tylko nlogn porównań

SORTOWANIE

Problem (IZOMORFIZM DRZEW)

Dane: T_1, T_2 - drzewa nieskierowane

Pytanie: czy $T_1 \cong T_2$ (są izomorficzne)

Cel: Algorytm $O(n)$

Sortowanie przez zliczanie

Dane: $a_1 \dots a_n$ $\forall_i\, a_i \in \{1 \dots k\}$

Zał: k nie jest duże

$O(n)$ • $\forall_{i=1\dots n}$ $c_i \leftarrow 0$ (liczniki)

$O(n)$ • for($i=1..n$) c_{a_i}++

$O(k)$ • for($i=1\dots k$) $c_i \leftarrow c_i + c_{i-1}$

$O(n)$ • wypisz elementy idąc od prawej do lewej

$O(n+k)$

2 2 4 3 4 1 2 4 1 3 3 4 3 3

2 5 10 14

2	3	5	4

1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	3	3	4	1

Def: Metoda jest stabilna jeśli zachowuje względną kolejność równych kluczy (z wejścia)

FAKT: Podana metoda sortowania jest stabilna