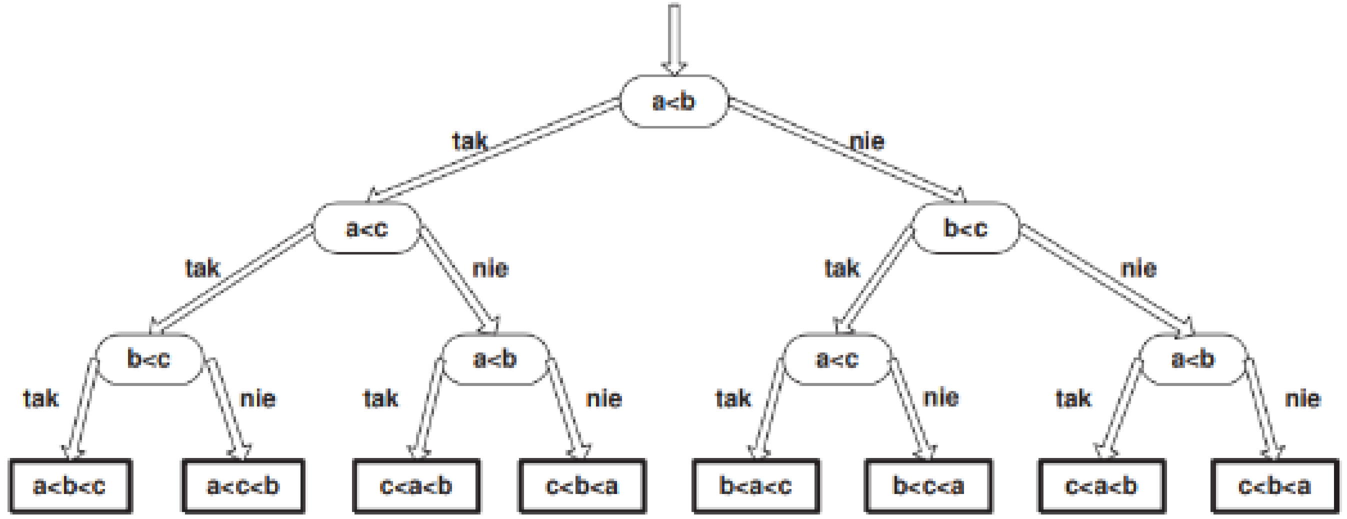
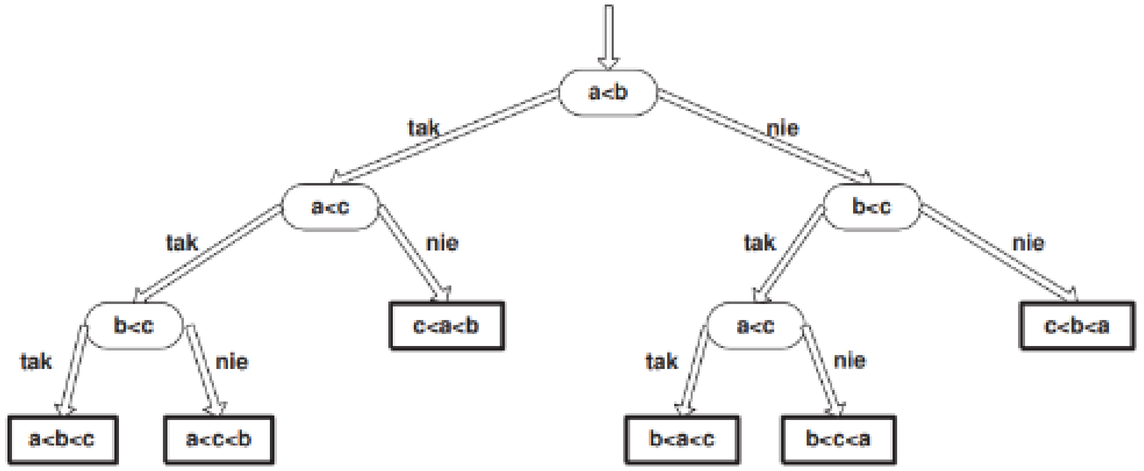
Dolne granice

Ponieważ od drzew decyzyjnych wymagamy by były skończone, jedno drzewo nie może reprezentować działania algorytmu dla dowolnych danych. Z reguły przyjmujemy, że algorytm reprezentowany jest przez nieskończoną rodzinę drzew decyzyjnych $\{D_i\}_{i=1}^{\infty}$, gdzie drzewo D_n odpowiada działaniu algorytmu na danych o rozmiarze n.



Rysunek 1: *Drzewo D_3 dla algorytmu sortowania przez selekcję.*



Rysunek 2: *Optymalne drzewo decyzyjne dla algorytmów sortujących ciągi 3-elementowe.*

Fakt 1 *Niech $\mathcal{A}$ będzie algorytmem sortującym, a $\{D_i\}_{i=1}^{\infty}$ – odpowiadającą mu rodziną drzew decyzyjnych. Wówczas drzewo D_n posiada co najmniej $n!$ liści, dla każdego n.*

Uzasadnienie: Każda permutacja ciągu wejściowego może być wynikiem, a każdy liść drzewa D_n odpowiada jednemu wynikowi. □

Twierdzenie 1 *Niech $\mathcal{A}$ będzie algorytmem sortującym, a $\{D_i\}_{i=1}^{\infty}$ – odpowiadającą mu rodziną drzew decyzyjnych. Wówczas drzewo D_n ma wysokość co najmniej $\Omega(n \log n)$.*

Uzasadnienie: Drzewo binarne o $n!$ liściach (a takim jest D_n) musi mieć wysokość co najmniej $\log(n!)$. Ze wzoru Stirlinga, $n!$ możemy z dołu oszacować przez $(n/e)^n$, co daje nam:

$$\log n! \geq n(\log n - \log e) \geq n \log n - 1.44n$$

Wniosek 1 *Każdy algorytm sortujący za pomocą porównań ciąg n - elementowy wykonuje co najmniej $cn \log n$ porównań dla pewnej stałej $c > 0$.*

2.1 Ograniczenie na średnią złożoność

Działanie algorytmu sortowania, który dane wykorzystuje wyłącznie w porównaniach, zależy jedynie od względnego porządku pomiędzy elementami. W szczególności nie zależy ono od bezwzględnych wartości elementów. Dlatego badając złożoność takich algorytmów możemy ograniczać się do analizy zachowania algorytmu na permutacjach zbioru $\{1, 2, \ldots, n\}$, a średnia złożoność algorytmu na danych rozmiaru n może być policzona jako suma:

$$\sum_{\sigma \in S_n} P[\sigma] c(\sigma),$$

gdzie S_n jest zbiorem permutacji zbioru $\{1,2,\ldots, n\}$, $P[\sigma]$ jest prawdopodobieństwem wystąpienia permutacji σ jako danych wejściowych, a $c(\sigma)$ jest równa liczbie porównań wykonywanych na tych danych. W języku drzew decyzyjnych można ją wyrazić jako średnią wysokość drzewa, tj.

$$\sum_{v - \text{ liść } T} p_v d_v,$$

gdzie p_v oznacza prawdopodobieństwo dojścia do liścia v, a d_v - jego głębokość.

Twierdzenie 2 *Jeżeli każda permutacja ciągu n-elementowego jest jednakowo prawdopodobna jako dana wejściowa, to wówczas każde drzewo decyzyjne sortujące ciągi n-elementowe ma średnią głębokość co najmniej $\log n!$.*

Uzasadnienie: Na głębokości nie większej niż $\log(n/e)^n - 1$ znajduje się mniej niż $n!/2$ liści. Tak więc co najmniej $n!/2$ liści osiągalnych z prawdopodobieństwem $1/n!$ leży na głębokości większej, co implikuje, że średnia wysokość drzewa decyzynego jest większa niż $(1/n!)(n!/2)\log((n/e)^n)$. □

Gra z adwersarzem

Twierdzenie 3 *Każdy algorytm rozwiązujący powyższy problem (i używający elementów zbioru S jedynie w porównaniach) wykonuje co najmniej $\lceil \frac{3}{2}n - 2 \rceil$ porównań.* (ALG MIN MAX1)

Dowód: Rozważmy następującą grę między algorytmem a złośliwym adwersarzem:

- Sytuacja początkowa: adwersarz twierdzi, że zna trudny dla algorytmu zbiór S, tj. taki, dla którego wskazanie przez algorytm minimum i maksimum będzie wymagało wykonania co najmniej $\lceil \frac{3}{2}n - 2 \rceil$ porównań. Algorytm nie zna S; wie tylko, że liczy on n elementów.
- Cel gry
  - algorytmu: wskazanie indeksów elementów minimalnego i maksymalnego w zbiorze S przy użyciu mniej niż $\lceil \frac{3}{2}n - 2 \rceil$ porównań;
  - adwersarza: zmuszenie algorytmu do zadania co najmniej $\lceil \frac{3}{2}n - 2 \rceil$ porównań.
- Ruchy
  - algorytmu: pytanie o porównanie dwóch elementów ze zbioru S;
  - adwersarza: odpowiedź na to pytanie.
- Koniec gry następuje, gdy algorytm wskaże minimum i maksimum w S. Wówczas adwersarz ujawnia zbiór S.

Tezę twierdzenia udowodnimy, jeśli pokażemy, że adwersarz zawsze, niezależnie od algorytmu, posiada strategię wygrywającą.

Strategia dla adwersarza:

- W trakcie gry adwersarz dzieli S na 4 rozłączne zbiory:

  $A = \{i \mid a_i$ jeszcze nie był porównywany $\}$,
  $B = \{i \mid a_i$ wygrał już jakieś porównanie i nie przegrał żadnego $\}$,
  $C = \{i \mid a_i$ przegrał juz jakieś porównanie i nie wygrał żadnego$\}$,
  $D = \{i \mid a_i$ wygrał już jakieś porównanie i jakieś już przegrał $\}$.

  Początkowo oczywiście $|A| = n$ oraz $|B| = |C| = |D| = 0$.
- Adwersarz rozpoczyna grę z dowolnymi kandydatami na wartości elementów a_i. W trakcie gry będzie, w razie konieczności, modyfikował te wartości, tak by spełniony był warunek

  $$(*) \qquad \forall_{a \in A} \forall_{b \in B} \forall_{c \in C} \forall_{d \in D} \;\; b > d > c \text{ oraz } b > a > c.$$

  Zauważ, że wystarczy w tym celu zwiększać wartości elementów o indeksach z B i zmniejszać wartości elementów o indeksach z C. Takie zmiany są bezpieczne dla adwersarza, ponieważ pozostawiają prawdziwymi jego odpowiedzi na dotychczasowe pytania.

Fakt 4 *Powyższa strategia adwesarza jest zawsze wygrywająca.*

Dowód faktu: W trakcie gry wszystkie elementy przechodzą ze zbioru A do B lub C, a dopiero stąd do zbioru D. Ponadto dla danych spełnających (*):

- jedno porównanie może usunąć co najwyżej dwa elementy ze zbioru A,
- dodanie jednego elementu do zbioru D wymaga jednego porównania,
- porównania, w których bierze udział element z A, nie zwiększają mocy zbioru D.

Dopóki A jest niepusty lub któryś ze zbiorów B lub C zawiera więcej niż jeden element, algorytm nie może udzielić poprawnej odpowiedzi. Na opróżnienie zbioru A algorytm potrzebuje co najmniej $\lceil n/2 \rceil$ porównań. Następnych $n-2$ porównań potrzebnych jest na przesłanie wszystkich, poza dwoma, elementów do zbioru D. □ (faktu i twierdzenia)

Zadania Egzaminacyjne

Treść

Dolną granicę $\lceil \frac{3}{2}n - 2 \rceil$ na liczbę porównań niezbędnych do wyznaczenia max i min w
adwersarzem.

Opisz skuteczną strategię w takiej grze.

Jeśli jest to strategia opisana na wykładzie, możesz na tym poprzestać.

Jeśli jest to inna strategia, wykaż, że jest skuteczna.

Treść

Na wykładzie pokazaliśmy, że Ω(nlogn) jest dolną granicą na złozoność problemu Element Uniqueness w modelu liniowych drzew decyzyjnych.

Podaj definicję tego problemu.

Co można powiedzieć o dolnej granicy na złożoność tego problemu w modelu (zwykłych) drzew decyzyjnych?

▾ Rozwiązanie

Problem: (Różność elementów)
Dane: Liczby rzeczywiste $x_1, x_2, \ldots, x_n$.
Zadanie: Sprawdzić, czy $\forall_{1 \leq i \neq j \leq n} x_i \neq x_j$.

Dolna granica na złożoność tego problemu w modelu zwykłych drzew decyzyjnych to log2(2) = 1, ponieważ mamy jedynie dwa liście z odpowiedziami **TAK** albo **NIE**. Ale ta odpowiedź jest bezwartościowa, ponieważ nie jesteśmy w stanie jednym pytaniem sprawdzić, czy zbiór nie zawiera czy zawiera duplikaty.

Zadania z Ćwiczeń

1. (1pkt) Pokaż, że problem znajdowania otoczki wypukłej nie może być rozwiązany w modelu drzew decyzyjnych.

2. (2pkt) Rozważmy następujący problem:

   Problem *3SUM*:
   Dane: Liczby rzeczywiste $x_1, \ldots, x_n$.
   Wynik: 'TAK' - jeśli $\exists_{1 \le i < j < k \le n}\ x_i + x_j + x_k = 0$,
   'NIE' - w przeciwnym przypadku.

   Udowodnij, że $\Omega(n \log n)$ jest dolną granicą na rozwiązanie tego problemu w modelu liniowych drzew decyzyjnych.

3. (1,5pkt) Ułóż algorytm rozwiązujący problem *3SUM* w czasie $O(n^2)$.

4. (Z 2pkt) Rozwiąż poprzednie zadanie dla takiej "wersji" problemu *3SUM*, w której operacja dodawania jest zastąpiona operacją *XOR*, tj. chcemy sprawdzić, czy w danym ciągu istnieją elementy a, b, c, takie że $a \oplus b \oplus c = 0$.

5. (2pkt) Rozważmy problem, polegający na sprawdzeniu, czy w zbiorze n punktów leżących na płaszczyźnie znajdują się co najmniej 3 punkty współliniowe. Udowodnij, że problem ten nie jest prostszy niż problem *3SUM*, a dokładniej, że istnienie algorytmu rozwiązującego ten problem w czasie $o(n^{2-\epsilon})$ implikuje istnienie algorytmu o złożoności $o(n^{2-\epsilon})$ dla problemu *3SUM*.

6. (1.5pkt) Udowodnij, że $2n - 1$ porównań trzeba wykonać, aby scalić dwa ciągi n elementowe w modelu drzew decyzyjnych. Zastosuj grę z adwersarzem, w której adwersarz na początku ogranicza przestrzeń danych tak, by zawierała $2n$ zestawów danych takich i by każde porównanie wykonane przez algorytm eliminowało co najwyżej jeden zestaw.

7. (Z 2pkt) Mamy dany ustalony zbiór permutacji na n elementach $F \subseteq S_n$. Chcemy posortować liczby $x_1, x_2, \ldots, x_n$ (używając tylko porównań) wiedząc, że ich permutacja sortująca należy do F. Pokaż, że istnieje dla tego problemu drzewo decyzyjne o głębokości $O(n + \log |F|)$.

8. (2pkt) W problemie *3D-Matching* dane są n elementowe zbiory A, B, C oraz zbiór uporządkowanych trójek $X \subseteq A \times B \times C$ a zadaniem jest sprawdzenie, czy istnieje n elementowy podzbiór $S \subseteq X$, taki że każdy element z $A \cup B \cup C$ należy do dokładnie jednej trójki z S.

   Pokaż redukcję wielomianową problemu *3SAT* do problemu *3D-Matching*.

9. (2pkt) Rozważmy problem wyznaczenia za pomocą porównań elementów największego i drugiego z kolei w zbiorze n-elementowym. Udowodnij, że $n + \lceil \log n \rceil - 2$ porównań potrzeba i wystarcza do wyznaczenia tych elementów.

---

1. (0pkt) Pokaż, że $\Omega(n \log n)$ pozostaje dolną granicą dla problemu sortowania, jeśli w modelu drzew decyzyjnych na zapytania o relację między elementami a i b możliwe są trzy odpowiedzi: "a<b", "a=b" i "a>b".

2. (1pkt) Pokaż, że problem "Element uniqueness" rozbija R^n na $\Omega(n!)$ spójnych składowych.

3. (2pkt) Rozważmy następujący problem weryfikacji rozmiaru wielozbioru (MSV). Dany jest wielozbiór złożony z n liczb rzeczywistych oraz liczba naturalna k. Należy sprawdzić, czy w tym wielozbiorze jest dokładnie k różnych elementów. Postaraj się wskazać jak najwięcej różnych spójnych składowych, na które problem ten rozbija R^n.

4. (2pkt) *Algebraiczne Drzewo Obliczeń* jest uogólnieniem algebraicznego drzewa decyzyjnego. Posiada ono dwa rodzaje wierzchołków:

   - wierzchołki obliczeniowe: z każdym takim wierzchołkiem u związana jest wartość f_u, która jest określona jako wynik jednej z poniższych opeacji:
     $$f_u \leftarrow f_w + f_v, \quad f_u \leftarrow f_w - f_v, \quad f_u \leftarrow f_w * f_v, \quad f_u \leftarrow f_w / f_v, \quad f_u \leftarrow \sqrt{f_v},$$
     gdzie f_w i f_v są wartościami skojarzonymi z pewnymi przodkami wierzchołka u lub są elementami ciągu wejściowego lub stałymi z R.
   - wierzchołki rozgałęziające: wierzchołek v wykonuje test $f_u < 0$ bądź $f_u \ge 0$ bądź $f_u = 0$, gdzie u jest przodkiem v.

   Problem Set Equality (SE) zdefiniowany jest następująco: dane są zbiory $X = \{x_1, \ldots, x_n\}$ oraz $Y = \{y_1, \ldots, y_n\}$; pytamy, czy $X = Y$. Pokaż , że jeśli dane dla problemu SE są liczbami całkowitymi, to problem ten może być rozwiązany w modelu Algebraicznych Drzew Obliczeń w czasie liniowym.

5. (1pkt) Rozważmy decyzyjną wersję problemu otoczki wypukłej: mamy dane n punktów $p_1, \ldots, p_n$ na płaszczyźnie oraz liczbę naturalną k. Pytamy, czy otoczka wypukła tego zbioru składa się z k punktów. Wiedząc, że problem MSV zdefiniowany w poprzednim zadaniu wymaga $\Omega(n \log k)$ operacji w modelu algebraicznych drzew decyzyjnych, pokaż, że w tym modelu decyzyjna wersja problemu otoczki także wymaga tylu działań.

6. (2pkt) Problem "Przekrój zbiorów" zdefiniowany jest następująco:

   Problem:
   Dane: Liczby rzeczywiste $x_1, \ldots, x_n, y_1, \ldots, y_n$.
   Wynik: 'TAK' - jeśli $\{x_1, \ldots, x_n\} \cap \{y_1, \ldots, y_n\} = \emptyset$
   'NIE' - w przeciwnym przypadku.

   Udowodnij, że $\Omega(n \log n)$ jest dolną granicą na rozwiązanie tego problemu w modelu liniowych drzew decyzyjnych.