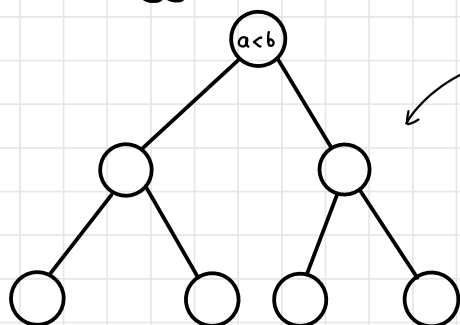


Dolma Granica 

# Drzewa Decyzyjne

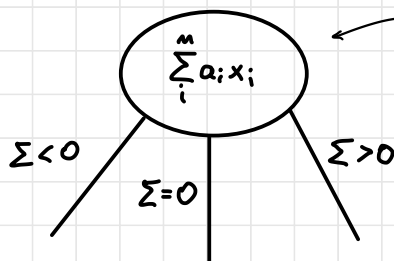
Drzewo decyzyjne daje nam tylko informację, czy  $a < b$  na każdym poziomie algorytmu. Mając tylko tę informację łatwo zauważyć, że dolną granicą takich algorytmów generuje odpowiednie drzewo decyzyjne:



Drzewo decyzyjne jest unikalne dla różnych rozmiarów danych jak i algorytmów.

## Linijowe Drzewa Decyzyjne

Tak jak zwykłe drzewa decyzyjne, lecz ternarne zamiast binarne.



Teraz operujemy na kombinacjach liniowych

# Gra z Adwersarzem



Gra z Adwersarzem przypomina grę w statki, gdzie masz algorytm pyta się adwersarza o jakąś ustaloną prymitywną operację - dla przykładu w modelu decyzyjnym czy  $a > b$ , a adwersarz odpowiada.

Jednakże adwersarz nie wymusza układu statków przed grą - w naszym przypadku nie wybiera zestawu danych przed uruchomieniem algorytmu. Ustala je z czasem gdy algorytm pyta o informacje.

Celem adwersarza jest sprawienie, by algorytm najdłużej pytał się o wyniki utrzymując wątpliwość danych.

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

Wyobraźmy sobie, że chcemy pokazać, że dla problemu min-maxowego wykonamy co najmniej  $\frac{3}{2}m-2$  porównań.

Znajdź najmniejszy i największy element

Okazuje się, że możemy wstawić dane do 4 zbiorów / kategorii:

A - Jeszcze nie porównywane

B - Większe od innych

C - Mniejsze od innych

D - Większe od innych ale mniejsze od innych

Teraz algorytm porównuje dwie liczby. Większą wrzuca do kategorii B, a mniejszą wrzuca do kategorii C. Wykona  $\frac{n}{2}$  porównań.

Adwersarz chce by te zbiory były jak największe, żeby utrudnić algorytmowi obliczenia.

Algorytm mógłby teraz porównywać elementy z B z elementami ze zbioru C, ale to by nic nie dało, bo adwersarz twierdziłby, że te z B są większe niż te z C.

Zatem algorytm porównuje elementy w kontekście jednego zbioru. Większe elementy ze zbioru C wrzuca do D, oraz mniejsze elementy ze zbioru B wrzuca do D.

Wykonuje zatem razem: 
$$\frac{n}{2} + (n-1) + (n-1) = \frac{3n}{2} - 2$$

$\uparrow$  segregacja do B lub C       $\uparrow$  znalezienie max w B       $\uparrow$  znalezienie min w C

# Problemy NP - trudne

## Terminologia

P - zbiór algorytmów deterministycznych w złożoności wielomianowej (polynomial algorithms)

NP - zbiór algorytmów niedeterministycznych w złożoności wielomianowej (non-deterministic polynomial algorithms).

## Opis

Jest to taki problem, którego nie potrafimy rozwiązać algorytmem A w złożoności wielomianowej, gdzie  $A \in P$ .

Jeśli problemy są ze sobą powiązane i potrafimy pokazać, że zachodzi między nimi taka sama relacja w złożoności czasowej (np. dla dwóch algorytmów w złożoności  $O(2^n)$ ), to razem są w tym samym zbiorze algorytmów **NP-trudnych**.

Jeśli dodatkowo napiszemy dla tych problemów algorytmy niedeterministyczne  $A' \in NP$ ,  $A'' \in NP$ , to wtedy są w tym samym zbiorze problemów **NP-kompletnych**.

Jeśli ktoś w tym momencie znajdzie algorytm  $A \in P$ , taki, że A rozwiązuje problem NP-kompletny, to wtedy na mocy tego, że te problemy są ze sobą powiązane możemy rozwiązać resztę problemów z tego samego zbioru.

W ten sposób ta osoba pokazała, że  $P = NP$ .