

Matematyka dyskretna

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

inf.ug.edu.pl/~amb/MaD

Andrzej M. Borzyszkowski (Instytut Informatyki) Matematyka dyskretna sem. zimowy 2025/26 1 / 21

Drzewa Drzewa binarne

Drzewa binarne

- drzewo ma jeden korzeń
każdy węzeł ma lewego i prawego syna
każdy z tych potomków może nie istnieć
- każdy węzeł drzewa można reprezentować słowem z $\{0, 1\}^*$
 - ε jest korzeniem
 - potomkami w są $w0$ oraz $w1$
 - długość słowa = odległość od korzenia
 - wysokość drzewa = największa odległość od korzenia
(czyli samotny korzeń ma wysokość zero)
- każde słowo $w \in \{0, 1\}^*$ określa węzeł w nieskończonym drzewie binarnym
 - słów długości n jest 2^n
 - słów długości $\leq n$ jest $\sum_{i=0}^n 2^i = 2^{n+1} - 1$
 - są to liście i węzły w drzewie binarnym pełnym wysokości n

Andrzej M. Borzyszkowski (Instytut Informatyki) Matematyka dyskretna sem. zimowy 2025/26 3 / 21

Drzewa binarne i uporządkowane

Andrzej M. Borzyszkowski (Instytut Informatyki) Matematyka dyskretna sem. zimowy 2025/26 2 / 21

Drzewa Drzewa binarne

Drzewa binarnych przeszukiwań

- dany zbiór uporządkowany (np. słowa jakiegoś utworu literackiego)
- umieszczamy je w węzłach drzewa binarnego tak by
 - słowo o kodzie $w0$ było wcześniejsze niż słowo o kodzie w
 - a $w1$ – późniejsze
 - jeśli dopuszczamy powtarzanie się słów w utworze, a każde wystąpienie chcemy umieścić, to jedna z nierówności powinna być nieostra
- w drzewie wysokości n można umieścić $2^{n+1} - 1$ elementów
jeśli drzewo ma wszystkie węzły

Andrzej M. Borzyszkowski (Instytut Informatyki) Matematyka dyskretna sem. zimowy 2025/26 4 / 21

Zgadywanie

- „zgadnij liczbę, którą właśnie pomyślałem”
wiadomo, że dodatnia, mniejsza od 1000
możliwe odpowiedzi: „mniejsza”, „większa”, „zgadłeś”
- „czy jest to 500?”
- jeśli mniejsza, to „może 250?”
a jeśli większa, to „może 750?”
- w każdym kroku przedział, gdzie znajduje się liczba zmniejsza się dwukrotnie
- najdalej w 10 kroku liczba zostanie znaleziona

Przykład

- „Wertowaliśmy odurzeni światłem w tej wielkiej księdze wakacji, której wszystkie karty pały od blasku i miały na dnie słodki do omdlenia miąższ złotych gruszek.”

The Project Gutenberg EBook of *Sklepy cynamonowe*, by Bruno Schulz

Title: Sklepy cynamonowe

Author: Bruno Schulz

Release Date: May, 2005 [EBook #8119]

Algorytmy dla drzew binarnych przeszukiwań

- wstawianie (etykiety e do drzewa T)
 - jeśli T jest puste, to wstaw do korzenia: $T(\varepsilon) = e$
 - jeśli $e < T(\varepsilon)$, to wstaw do lewego syna,
t.zn. wykonaj algorytm z lewym synem zamiast T
 - jeśli $e > T(\varepsilon)$, to wstaw do prawego syna
- trzeba wyjaśnić sprawę ew. powtórzeń etykiety, czy są możliwe? czy ignorujemy? czy notujemy krotkość wystąpienia?
- wyszukiwanie etykiety e w drzewie T
 - jeśli T jest puste, to e nie ma w T
 - jeśli $e = T(\varepsilon)$ to e znalezione w korzeniu
 - jeśli $e < T(\varepsilon)$ to szukaj e u lewego syna
 - w p.p. szukaj e u prawego syna

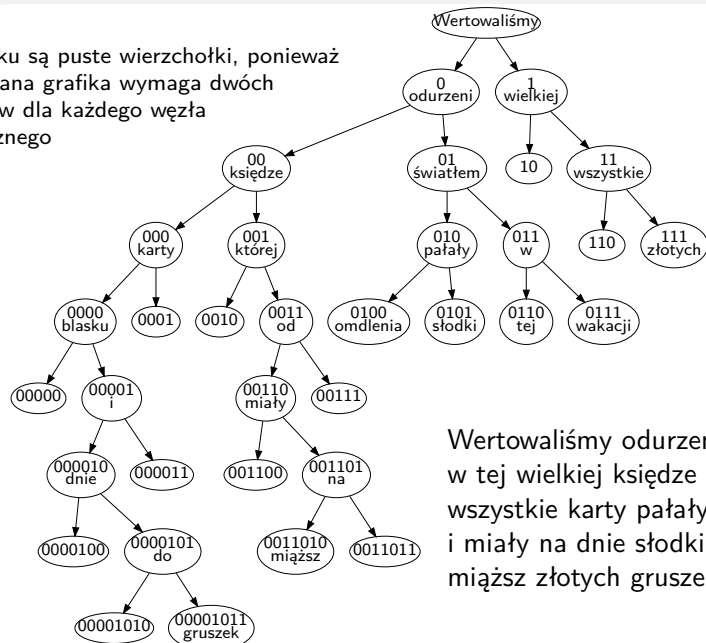
Przykład

Wertowaliśmy (ε)	odurzeni (0)	światłem (01)
w (011)	tej (0110)	wielkiej (1)
księdze (00)	wakacji (0111)	której (001)
wszystkie (11)	karty (000)	pały (010)
od (0111)	blasku (0000)	i (00001)
miały (00110)	na (001101)	dnie (000010)
słodki (0101)	do (0000101)	omdlenia (0100)
miąższ (0011010)	złoty (111)	gruszek (00001011)

- wysokość drzewa wynosi 8 (najdłuższe słowo: 00001011)
- drzewo pełne wysokości 8 ma 511 węzłów, a to tylko 24
- wystarczyłoby drzewo wysokości 4 ($24 < 32 = 2^5$), gdyby słowa były starannie ułożone

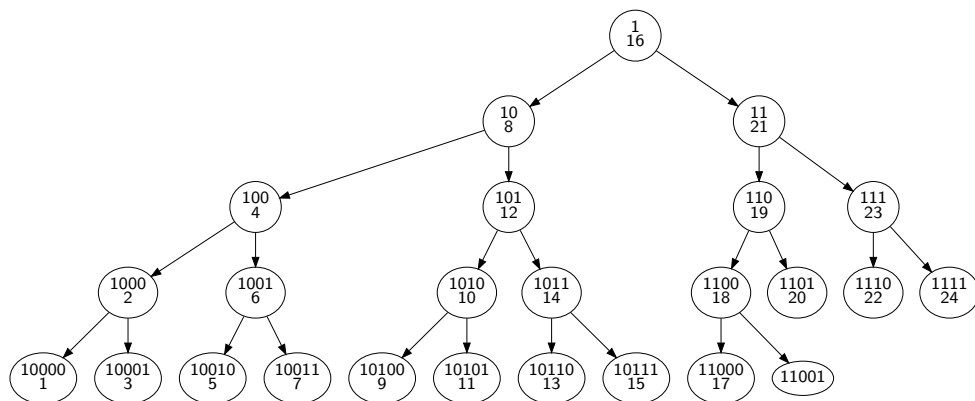
Przykład – drzewo binarnych przeszukiwań

na rysunku są puste wierzchołki, ponieważ zastosowana grafika wymaga dwóch potomków dla każdego węzła wewnętrznego



Wertowaliśmy odurzeni światłem w tej wielkiej księdze wakacji, której wszystkie karty pały od blasku i miały na dzień słodki do omdlenia miąższ złotych gruszek

Zrównoważone drzewo binarnych przeszukiwań



- jedynka z przodu kodowania węzła daje numerację od 1 do 24 (ostatni węzeł nie jest tutaj używany)
- teraz można wstawić dowolne 24 uporządkowane elementy, tak by tworzyły drzewo binarnych wyszukiwań z optymalną złożonością wyszukiwania

Dalsze algorytmy dla drzew binarnych przeszukiwań

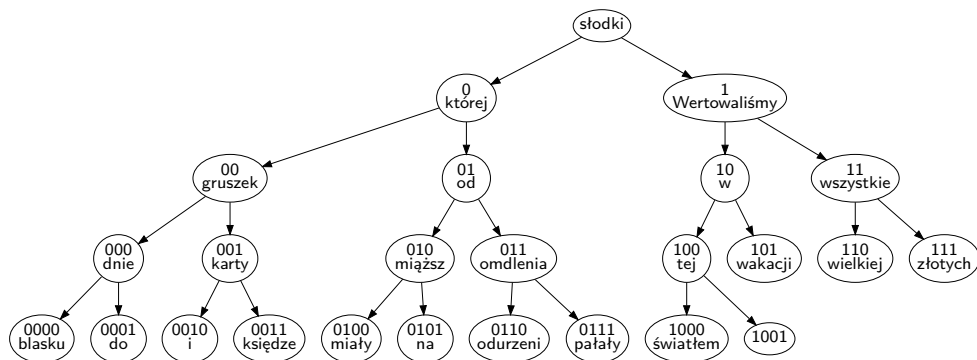
- Wyszukiwanie najmniejszej etykiety w drzewie T :
 - jeśli T jest puste, to nie ma elementów
 - jeśli T nie ma lewego syna, najmniejszym elementem w T jest $T(\varepsilon)$
 - w p.p. szukaj u lewego syna
- Wypisywanie uporządkowanych etykiet drzewa T :
 - jeśli T jest puste, to nie ma co wypisywać
 - w p.p. wypisz uporządkowane etykiety lewego syna T
 - następnie etykietę $T(\varepsilon)$
 - i na końcu wypisz uporządkowane etykiety prawego syna T

Przykład raz jeszcze

- 1 blasku
- 2 dzień
- 3 do
- 4 gruszek
- 5 i
- 6 karty
- 7 księdze
- 8 której
- 9 miały
- 10 miąższ
- 11 na
- 12 od
- 13 odurzeni
- 14 omdlenia
- 15 pały
- 16 słodki
- 17 światłem
- 18 tej
- 19 w
- 20 wakacji
- 21 Wertowaliśmy
- 22 wielkiej
- 23 wszystkie
- 24 złotych

- kolejność alfabetyczna wyrazów w przykładzie pokazuje jak zbudować drzewo zbalansowane
- na początek wyraz 16, potem 8, 21 itd.

Przykład – drzewo binarnych przeszukiwań zrównoważone



- Wysokość tego drzewa wynosi 4, wszystkie liście są na dwóch sąsiednich poziomach
- złożoność wyszukiwania jest $O(\log_2 \ell)$, gdzie ℓ jest liczbą elementów dokładniej, wysokość jest $\leq \log_2 \ell$

Reprezentacja drzew binarnych w jęz. C

```
struct tnode {
    char *word;
    int count;
    struct tnode *left;
    struct tnode *right;
};
```

drzewo binarne, etykietą jest para:
słowo i licznik wystąpień

```
void treeprint(struct tnode *p) {
    if (p != NULL) {
        treeprint(p->left);
        printf("%4d %s\n", p->count, p->word);
        treeprint(p->right);
    }
}
```

wydruk zawartości
drzewa binarnego,
najpierw lewe podrzewo,
potem korzeń i prawe
podrzewo

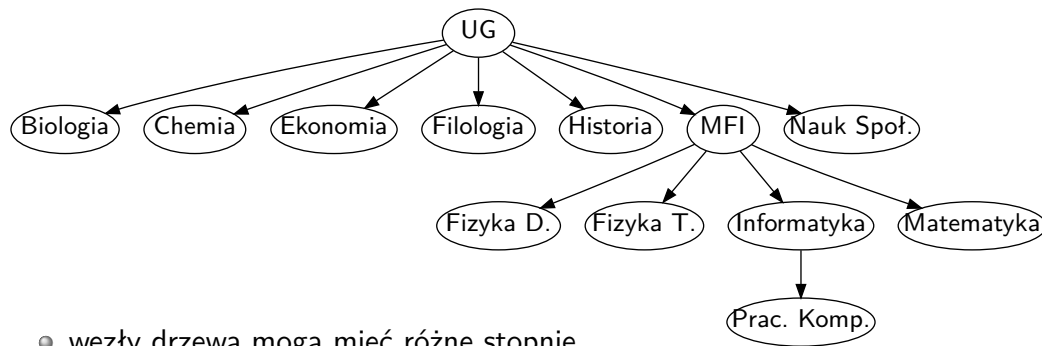
Reprezentacja drzew binarnych – wstawianie

```
struct tnode *addtree(struct tnode *p, char *w) {
    int cond;
    if (p == NULL) {
        p = malloc(sizeof(struct tnode));
        p->word = strdup(w);
        p->count = 1;
        p->left = p->right = NULL;
    } else if ((cond = strcmp(w, p->word)) == 0) {
        p->count++;
    } else if (cond < 0) {
        p->left = addtree(p->left, w);
    } else {
        p->right = addtree(p->right, w);
    }
    return p;
}
```

Reprezentacja drzew binarnych w tablicy

- drzewo binarne pełne ma węzły o numerach od 1 do $2^n - 1$
- mniejsza liczba węzłów może być umieszczona kolejno na najniższym poziomie
- odpowiada to tablicy o kolejnych numerach $1 \dots N$
- korzeń ma numer 1, lewy potomek węzła i ma numer $2 \cdot i$, prawy $2 \cdot i + 1$
- poddrzewo pod węzłem i (potomkowie i) mają numery $2^\ell \cdot i, \dots, 2^\ell \cdot i + 2^\ell - 1$ gdzie ℓ jest odległością od węzła

Drzewa niebinarne



- węzły drzewa mogą mieć różne stopnie
- ale potomkowie każdego węzła mają ustaloną kolejność
- reprezentacja w jęz. C wymaga raczej listy potomków – najstarszy syn i wskaźnik na kolejnego

Drzewa o stałym stopniu rozgałęzienia, c.d.

- dany alfabet A mocy $\ell = |A|$, każde słowo $w \in A^*$ określa węzeł
 - ε jest korzeniem
 - bezpośrednimi potomkami w są wx dla $x \in A$
 - czyli potomkowie w są rozróżniani wg nazwy z A
 - wszyscy potomkowie węzła w są postaci wu dla $u \in A^+$
 - słów długości n jest ℓ^n
 - słów długości $\leq n$ jest $\sum_{i=0}^n \ell^i = \frac{\ell^{n+1}-1}{\ell-1}$

Drzewa o stałym stopniu rozgałęzienia

- drzewa ternarne
 - każdy węzeł ma trzech potomków o ustalonych znaczeniach
 - np. „mało”, „średnio”, „dużo” albo inne sytuacje
 - liczba węzłów dla drzewa wysokości h wynosi $\sum_{i=0}^h 3^i = \frac{1}{2}3^{h+1}$
 - reprezentacja w jęz. C za pomocą struktury z polami odpowiadającymi trzem opcjom
 - reprezentacja w tablicy gdzie potomkiem węzła i są węzły $3 \cdot i$, $3 \cdot i + 1$ oraz $3 \cdot i + 2$

Ciąg geometryczny raz jeszcze

- $\sum_{i=0}^n 2^i = 2^{n+1} - 1$
- dowolna liczba $0 \leq \ell < 2^{n+1}$ jest sumą pewnych wyrazów tego ciągu i to na jeden tylko sposób
- dw.: są to cyfry przedstawienia ℓ w układzie dwójkowym
- dane są odważniki o wagach $3^i, i = 0, 1, \dots, n$ można je kłaść na dowolnej z dwóch szalek wagi
- $\sum_{i=0}^n 3^i = \frac{1}{2} \cdot (3^{n+1} - 1)$ dowolny ciężar nie większy niż $\frac{1}{2} \cdot (3^{n+1} - 1)$ może być zważony za pomocą tych odważników
- dw.: przyjmujemy układ trójkowy z cyframi $-1, 0, 1$ np. $(20)_{10} = (1, -1, 1, -1)_3$ czyli $20 = 27 - 9 + 3 - 1$
- zamiana szalek miejscami mogłaby być interpretowana jako ważenie ciężarów ujemnych

Drzewa algorytmów

- dane są cztery monety, jedna być może fałszywa
- monety można ważyć na wadze z szalkami
- algorytm szukania fałszywej monety jest drzewem stopnia 3
korzeniem jest start
każde ważenie może dać jeden z trzech możliwych wyników
- drzewo o 9 liściach musi mieć wysokość co najmniej 2
 - jeśli wysokość 2, to przy każdym wyniku pierwszego ważenia muszą pozostać 3 możliwości
 - i muszą wyjaśnić się w drugim ważeniu
- nie ma algorytmu działającego w dwóch ważeniach
 - jeśli porównujemy M_1 i M_2 i np. $M_1 = M_2$, to pozostaje 5 możliwości
 - jeśli porównujemy $M_1 + M_2$ i $M_3 + M_4$ i np.
 $M_1 + M_2 < M_3 + M_4$, to pozostają 4 możliwości
- istnieje algorytm, jeśli mamy pięć, dobrą monetę