Spis treści

Rozdział 1

Zagadnienia

[TODO: trzeba podzielić na części egzaminu]

- Algorytmy
  - mnożenie po rosyjsku
  - szybkie potęgowanie
  - szybkie obliczanie liczb Fibonacciego
  - algorytmy sortowania: select i insert
  - maszyna RAM jako model obliczeń
  - notacja asymptotyczna
- Algorytmy zachłanne
  - problem wydawania reszty
  - algorytmy znajdowania minimalnego drzewa rozpinającego
    - algorytm Prima
    - algorytm Kruskala
    - algorytm Boruvki
  - szeregowanie zadań
  - szeregowanie zadań z deadlineami
- Metoda Dziel i Zwyciężaj
  - algorytmy sortowania (mergesort, quick sort)
  - algorytmy mnożenia długich liczb (Karatsuby)
- Metoda Dziel i Zwyciężaj (cd)
  - algorytmy mnożenia długich liczb oparte na podziale na wiele części
  - jednoczesne znajdowanie maksimum i minimum
  - sieć permutacyjna Waksmana-Benesa
  - znajdowanie najbliższej pary punktów na płaszczyźnie
  - zachłanny algorytm aproksymacyjny dla problemu Set Cover
- Programowanie dynamiczne
  - obliczanie współczynnika dwumianowego

  - stokrotki
  - znajdowanie nakrótszych dróg w grafie między wszystkimi parami wierzchołków
  - algorytm Floyda Warshalla
  - najdłuższy wspólny podciąg
- Programowanie dynamiczne (cd)
  - optymalna kolejność mnożenia macierzy
  - przynależność słowa do języka bezkontekstowego
  - drzewa rozpinające drabin
- Dolne granice (cd)
  - gra z adwersarzem - jednoczesne znajdowanie minimum i maksimum
  - kopiec
    * definicja
    * implementacja tablicowa
    * realizacja operacji kopcowych
    * zastosowania: heapsort, kolejka priorytetowa
  - metody dowodzenia dolnych granic
    * argumentacja teorio-informatyczna
      · drzewa decyzjne - sortowanie; dolna granica w najgorszym i średnim przypadku
      · liniowe drzewa decyzyjne - problem: niepowtarzalność elementów
    * redukcje
      · nieaproksymowalność problemu komiwojażera
- Kopiec (cd)
  - kopiec minmax
  - Qucksort
  - metody wyboru pivota (deterministycza, losowa, mediana z małej próbki)
  - analiza oczekiwanego czasu działania przy losowym pivocie
  - usprawnienia Quicksorta
    * trójpodział
    * eliminacja rekursji
    * quicksort w miejscu
- Sortowanie
  - przez zliczanie
  - kubełkowe
  - leksykograficzne ciągów jednakowej długości
- Quicksort
  - prostsza metoda analizy oczekiwanego czasu działania Quicksorta przy losowym wyborze pivota
- Selekcja
  - znajdowanie drugiego co do wielkości elementu

  - metoda Hoare'a
  - metoda magicznych piątek

- Słowniki
  - drzewa zrównoważone
    - drzewa AVL (definicja, realizacja operacji słownikowych)
    - drzewa czerwono-czarne (definicja, realizacja operacji słownikowych)
  - selekcja
    - zrandomizowany algorytm oparty na próbkowaniu
  - sortowanie ciągów niekoniecznie jednakowej długości z zastosowaniem do sprwadzania izomorfizmu drzew

- Hashowanie
  - przykłady funkcji hashujących
  - kolizje
  - pamiętanie elementów kolidujących (nawlekanie, adresowanie otwarte)
  - struktury samoorganizujące się
    - idea
    - heurystyki dla list samoorganizujących się
    - drzewa rozchylane (splay)
    - analiza zamortyzowania ciągów operacji splay
  - kopce dwumianowe (wersja leniwa)
    - analiza zamortyzowana kosztu operacji
  - kopce Fibonacciego
    - zastosowanie: algorytm Dijkstry
    - wpływ kaskadowego odcięcia drzew na koszt operacji
  - B-drzewa
    - definicja
    - liczba operacji i/o jako miara złożoności
    - realizacja operacji słownikowych
  - 2-3-drzewa i ich związek z drzewami czerwono czarnymi
  - złączalne kolejki priorytetowe
    - kopce dwumianowe (wersja gorliwa)

- Hashowanie (cd)
  - usuwanie kolizji w otwartym adresowaniu
    - metoda liniowa
    - metoda kwadratowa
    - podwójne hashowanie
  - oczekiwana liczba prób przy poszukiwaniu elementu
  - uniwersalne rodziny funkcji hashujących

- Słowniki statyczne
  - optymalne drzewa BST

  - statyczny słownik Fredman, Komlosa, Szemeredi'ego
- Drzewce
  - definicja
  - istnienie
  - realizacja operacji słownikowych
- Statyczny słownik Fredman, Komlosa, Szemeredi'ego (cd)
  - idea konstrukcji
  - istnienie odpowiednich funkcji hashujących
  - znajdowanie takich funkcji

Rozdział 2

I termin 2010 - odsłona 1 i 2

2.1. zadanie 1

Przedstaw ideę algorytmu Boruvki (Sollina).

G - nasz graf G' - nasza odpowiedź, początkowo tylko wierzchołki z G i żadnych krawędzi Wykonujemy kroki:

1. Dla każdego wierzchołka w G znajdź najkrótszą incydentną krawędź i dodaj ją do G'
2. Utwórz nowy graf G', w którym wierzchołkami są spójne składowe w starym G'

Iterujemy do póki nie zostanie jednen wierzchołek. Wszystkie dodane krawędzie do G' utworzą odpowiedź.

2.2. zadanie 2

Które z poniższych algorytmów mogą działać niepoprawnie dla grafów z ujemnymi wagami krawędzi? Odpowiedź uzasadnij.

a) algorytm Kruskala

b) algorytm Prima

c) algorytm Dijsktry

a) Kruskala - działa dobrze

b) Prima - działa dobrze

c) Dijkstra - Jeżeli gdzieś w grafie występuje cykl o negatywnej wadze, to wszystkie drogi nie mają najkrótszej ścieżki. Algorytm może tego nie wykryć, bo nigdy nie wraca do wierzchołków już rozważonych, a cykl możemy znaleść dopiero pod koniec wykonania, dlatego zwróci jakieś wartości dla wierzchołków, a nie powinien.

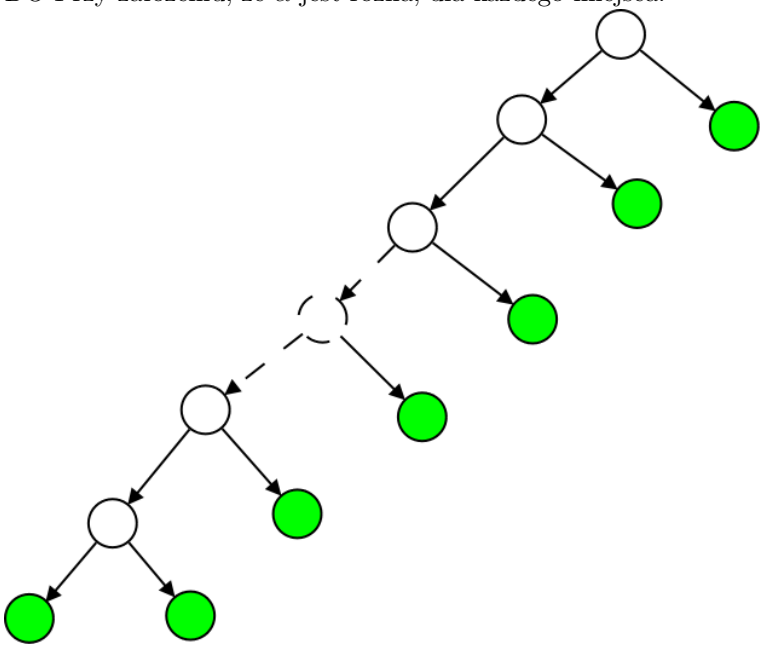
2.3. zadanie 3

Rozważmy następujące kryterium zrównoważenia drzew:

$$l(w) < \alpha l(v)$$

dla każdego wierzchołka v i dla każdego jego syna w, gdzie $l(v)$ oznacza liczbę liści w poddrzewie o korzeniu w v a α jest pewną liczbą mniejszą od 1. Czy ten warunek gwarantuje, że w drzewie nie powstaną długie ścieżki?

TODO Przy założeniu, że α jest różna, dla każdego miejsca.



Zielone to liście. Jeżeli skonstruujemy drzewo w taki sposób, jak na rysunku, to widzimy, zę w każdym punkcie niezmiennik jest spiełniony. W każdym wierzchołku, który ma synów, dokonujemy podziału pozostałych elementów w stosunku $(k-1):1$.

Ścieżka jest liniowej długości, bo ma długość około $0.5n = O(n)$, czyli jest długa.

2.4. zadanie 4

O ile co najwyżej może zwiększyć się liczba drzew w kopcu Fibonacciego w skutek wykonania pojedynczej operacji *decreasekey*?

Zmniejszamy wartość liścia znajdującego się na najniższym poziomie w największym drzewie w kopcu. Jeśli wszyscy przodkowie tego liścia mają po jednym synu(czyli są zaznaczone jak mówi CLRS), wtedy kaskadowo odcinamy każdego przodka liścia, tworząc nowe drzewo w kopcu. Czyli liczba drzew zwiększy się o wysokość największego drzewa w kopcu. Ta liczba wynosi $k = O(\log n)$, bo wiemy, że drzewo o stopniu k ma wykładniczą ilość elementów, może być to jedyne drzewo, więc $n = 2^k$.

2.5. zadanie 5

Dla której z podanych struktur danych koszt (najgorszego przypadku) wykonania operacji $find(i)$ sprawdzającej czy klucz i jest pamiętany w strukturze jest $O(\log n)$, gdzie n jest rozmiarem struktury?

a) drzewo binarnych przeszukiwań

b) drzewo AVL

c) kopiec

d) kopiec dwumianowy

e) kopiec Fibonacciego

f) drzewo czerwono-czarne

a) drzewo poszukiwań binarnych - może być listą - $O(n)$

b) drzewo AVL - niezmienniki gwarantuja zrównoważenie - $O(\log n)$

c) kopiec - w ogóle nie ma posortowania - $O(n)$

d) kopiec dwumianowy - j.w. - $O(n)$

e) drzewa czerwono-czarne - jak AVL - $O(\log n)$

2.6. zadanie 6

Rozwiąż równanie rekurencyjne

$$T(1) = 1$$
$$T(2) = 3$$
$$T(n) = T(n-2) + 2n - 1 \qquad \text{dla } n > 2$$

$$T(1) = 1$$
$$T(2) = 3$$
$$T(n) = T(n-2) + 2n - 1 = T(n-2) + n + (n-1) \qquad \text{dla } n > 2$$

$$T(n) = n + (n-1) + T(n-2) =$$
$$= n + (n-1) + (n-2) + (n-3) + T(n-4) =$$
$$= n + (n-1) + (n-2) + (n-3) + \cdots + 4 + 3 + 2 + 1 =$$
$$= \frac{n(n+1)}{2}$$

2.7. zadanie 7

Który z poniższych algorytmów sortowania może w najgorszym przypadku wykonać $\Omega(n^2)$ porównań:

a) quicksort

b) mergesort

c) insertsort

Przypomnienie: $\Omega(n^2)$ oznacza nie mniej niż cn^2 dla pewnej stałej $c > 0$.

a) quicksort - jeżeli znajdowanie pivota jest źle zrobione, tzn. odcina stałą liczbę elementów przy każdym partition, np. stosunek $1 : (n-1)$, to wtedy złożoność to $n * O(n) = O(n^2)$

b) mergesort - worst case $O(n \log n)$

c) insertsort - posortowany lub odwrotnie posortowany ciag bedzie mieć $O(n^2)$

2.8. zadanie 8

Złożoność algorytmu magicznych piątek wyraża się nierównością

$$T(n) \leqslant T(\lceil n/5 \rceil) + T(\lceil 7n/10 \rceil) + O(n)$$

Wyjaśnij skąd biorą się składniki po prawej stronie nierówności. Uzasadnij czemu $T(n)$ jest $\theta(n)$.

- $T(\lceil n/5 \rceil)$ - podczas algorytmu dzielimy elementy na piątki, w każdej piątce znajdujemy medianę i rekurencyjnie wybieramy medianę z tych $\lceil n/5 \rceil$ median, aż zostanie jedna wartość.
- $T(\lceil 7n/10 \rceil)$ - jak znaleźliśmy już medianę median z kroku wyżej, to wiemy z przechodniości mniejszości, że ten element jest większy od, co najmniej, $3n/10$ elementów (w każdej piątce mediana była większa od 2 elementów, my bierzemy medianę median, czyli w połowie piątek nasz element był większy od 3 elementów).
- $O(n)$ - koszt przejścia przez elementy i znalezienia median piątek.

Dlaczego jest $\theta(n)$

$$T(n) \leqslant T(1n/5) + T(7n/10) + O(n) = c(4n/20 + 14n/20) + O(n) = cn - (2n/20 - O(n)) \leqslant cn$$

Dobieramy c na tyle duże, żeby pasowało do wartości kosztu obu wywołań.

2.9. zadanie 9

Napisz procedurę partition (nie musi być to wersja z wykładu, ale musi być efektywna).

Pivot jest w A[p], funkcja zwaraca granicę podziału

```
Part(A[1..n], p, r)
  x <- A[p]
  i <- p-1
```

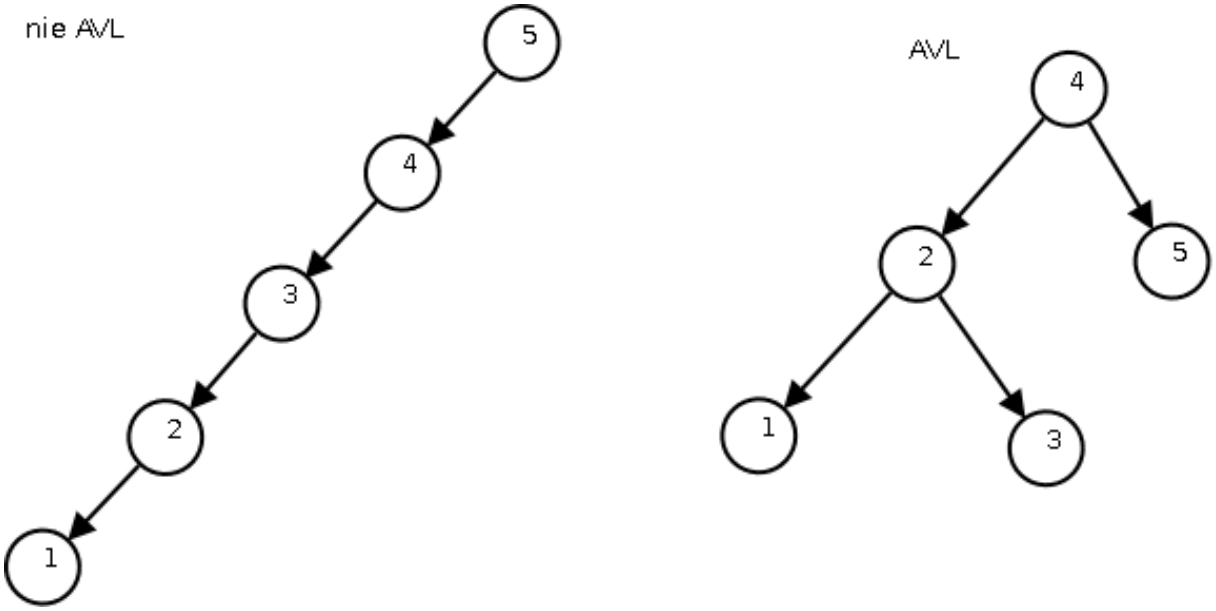
```
j <- r+1

 while(i<j) do
   repeat(j--) until A[j] <= x
   repeat(i++) until A[i] >= x

   if(i<j) swap(A[i], A[i])
   else return j
```

2.10. zadanie 10

Narysuj drzewo binarnych wyszukiwań pamiętające klucze 1,2,3,4,5, które

a) jest drzewem AVL

b) nie jest drzewem AVL



2.11. zadanie 11

Przedstaw strategię zachłanną algorytmu aproksymacyjnego dla problemu Set Cover o współczynniku aproksymacji H_n.

Problem: Wybrać najmniejszy taki podzbiór z rodziny zbiorów, aby wszystkie pojedyncze elementy z całej rodziny znalazły sie w tym podzbiorze. Algorytm aproksymacyjny: W każdym kroku odrzucamy zbiór, który nie pokrywa jak nawiekszej części elementów.

U – uniwersum do pokrycia
R – wejsciowa rodzina zbiorow
W – wynikowa rodzina zbiorow na poczatu pusta

```
while U jest niepusty
    wybierz takie Z z R ze Z przekroj U jest maksymalne
    dodaj Z do rodziny wynikowej W
    U \= Z
    usun Z z R
```

2.12. zadanie 12

Ile operacji *join* wykona się podczas łączenia kopców dwumianowych (wersja eager) zawierających odpowiednio 53 i 35 elementów. Przypomnienie: operacja *join* łączy dwa drzewa dwumianowe tego samego rzędu.

Tyle ile jest binarnych carry podczas dodawania 35 i 53.
0100011
0110101
———-
1011000
Wychodzi 4.

2.13. zadanie 13

Podaj definicję uniwersalnej rodziny funkcji hashujących.

Niech H będzie rodziną funkcji hashujących z U w $\{0, ..., m-1\}$.
Rodzinę H nazywamy uniwersalną, jeśli $\forall_{x,y \in U; x \neq y} |h \in H : h(x) = h(y)| = \frac{|H|}{m}$

2.14. zadanie 14

Ile różnych dzewców można utworzyć dla n-elementowego zbioru kluczy $\{a_1, \ldots, a_n\}$, którego pewnym dwóm elementom omyłkowo przypisano takie same priorytety (a pozostałym różne)?

Wiemy, że dla jednego setu priorytetów i kluczy mamy jednego możliwego drzewca. Zakładamy raz, że jeden z równych priorytetów jest wyższy - mamy 1. drzewca, potem zakładamy, że jest niższy - mamy 2. drzewca.

2.15. zadanie 15

Na czym polega opracja kaskadowego odcinania w kopcach Fibonacciego?

Żeby zachować niezmienniki potrzebne do amortyzowanej złożoności, nie pozwalamy w operacji odcięcia, usunięcia więcej niż jednego poddrzewa z jednego wierzchołka. Dlatego zaznaczamy ojca, który utracił jednego syna i przy próbie usunięcia mu kolejnego syna usuwamy całe poddrzewo i dodajemy je do ogólnej listy. Po tym markujemy jego dziadka (jeżeli jest zmarkowany, to znowu usuwamy poddrzewo i rekurencyjnie w gore).

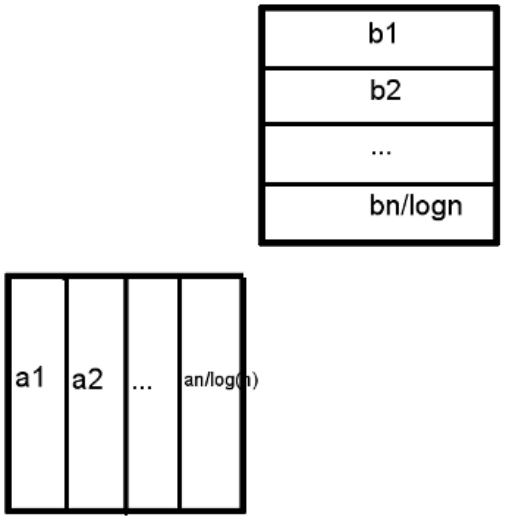
2.16. zadanie 16

Ile drzew może zawierać n-elementowy kopiec dwumianowy (w wersji lazy) po wykonaniu operacji deletemin?

Mamy n-elementowy kopiec w wersji lazy. Robimy deletemin. W najgorszym przypadku minimum będzie w korzeniu największego drzewa w kopcu. Wysokość tego drzewa to $k = \lfloor log_2 n \rfloor + 1$. Wielkość tego drzewa to oczywiście $2^{\lfloor log_2 n \rfloor}$.
Wiadomo też z definicji drzewa dwumianowego, że drzewo wysokości k posiada $k-1$ synów, z których każdy jest korzeniem drzewa wysokości kolejno $k-1, k-2, ..., 1$. Zatem usuwając minimum z drzewa o wysokości k dodajemy do kopca $k-1 = \lfloor log_2 n \rfloor$ nowych drzew.

2.17. zadanie 17

W algorytmie czterech Rosjan obliczane są iloczyny macierzy o rozmiarze $nx \log n$ i macierzy o rozmiarze $\log nxn$ Ile takich iloczynów jest obliczanych?



Liczymy iloczyny macierzy dla każdego $a_i \cdot b_i$. Takich iloczynów jest oczywiscie $\frac{n}{\log(n)}$, a każdy w wyniku daje jedną macierz kwadratową. Jeśli zsumujemy te macierze to otrzymamy iloczyn, o który nam chodzi.

2.18. zadanie 18

Opisz, w jaki sposób DFS może być zastosowane do znalezienia cyklu Eulera w grafie.

Graf przechodzimy przy pomocy rekurencyjnej procedury DFS. Przebyte krawędzie usuwamy, a wierzchołki po zakończeniu przetwarzania umieszczamy na początku kolejki. Jeśli graf posiada cykl Eulera, to po zakończeniu algorytmu w kolejce znajdą się kolejne wierzchołki tego cyklu.

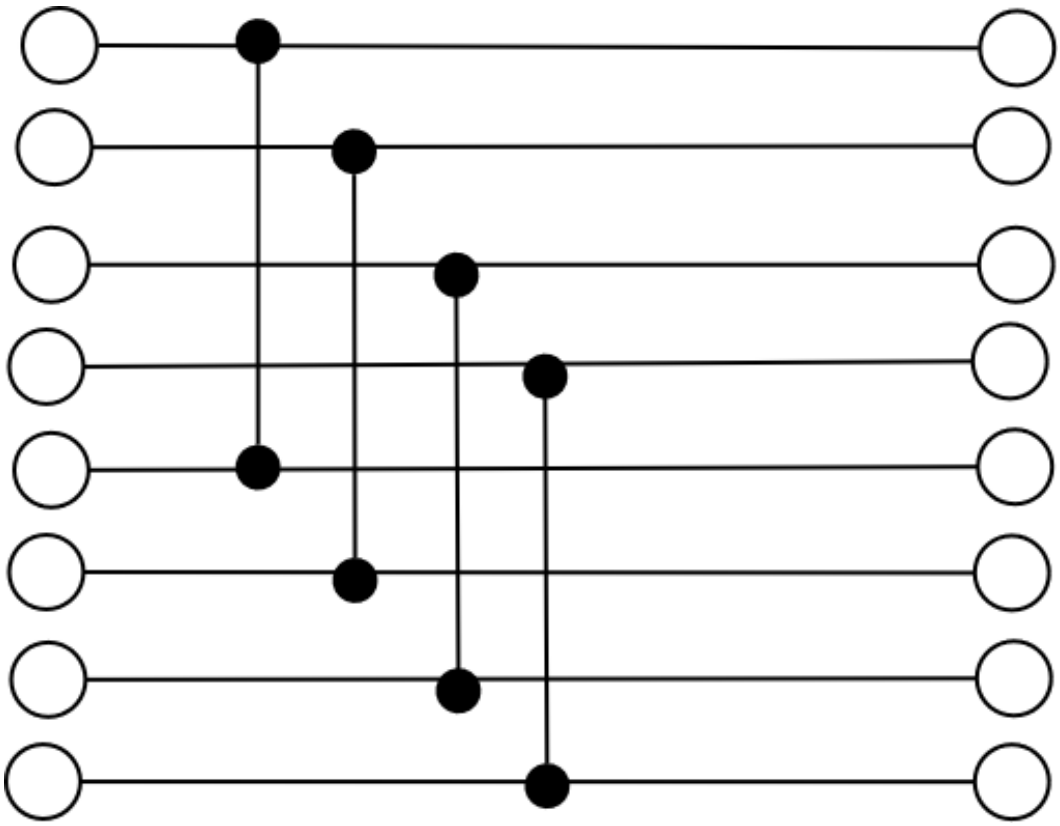
2.19. zadanie 19

Ile funkcji hashujących musimy znaleźć knstruując słownik statyczny dla zbioru n kluczy (chodzi o konstrukcję opartą na hashowaniu dwupoziomowym).

n kluczy rozrzucamy do m różnych tablic jedną funkcją hashującą. Potem w każdej z m tablicy hashujemy elementy zamiast tworzyć listę elementów. Czyli potrzeba $m + 1$ funkcji hashujących.

2.20. zadanie 20

Narysuj sieć półczyszczacą o ośmiu wejściach.



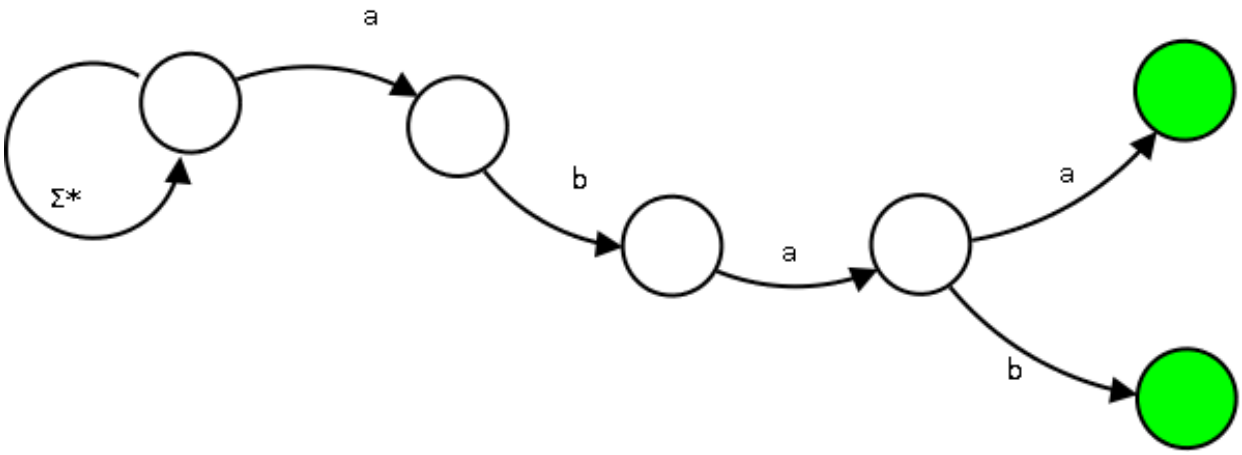
2.21. zadanie 21

Wylicz funkcję π dla wzorca *abracadabra*.

a	b	r	a	k	a	d	a	b	r	a
0	0	0	1	0	1	0	1	2	3	4

2.22. zadanie 22

Narysuj automat rozpoznający *abaa* i *abab*.



2.23. zadanie 23

Jaki byłby koszt wykonania ciągu δ złożonego z $O(n)$ operacji $UNION$ i $FIND$, gdyby w operacji $UNION$ zbiory były łączone w dowolny (niekoniecznie zrównoważony sposób), a operacja $FIND$ nadal byłaby wykonywana z kompresją ścieżek?

Przy zastosowaniu tylko kompresji ścieżki, koszt uniona pozostaje $O(1)$, a amortyzowany koszt finda to $O(\log n)$. W najgorszym przypadku, koszt, to pewnie $O(n \log n)$.

2.24. zadanie 24

Opisz ideę algorytmu Shift-Or

TODO

2.25. zadanie 25

W jaki sposób problem mnożenia macierzy może być wykorzystany do rozwiązania problemu najkrótszych ścieżek w grafie?

Mamy grafy A i B w postaci macierzy adjacencji.
Możemy korzystać z algebry, w której:
'+' to działanie min
'·' to dodawanie
Wtedy jedno mnożenie macierzy, to jeden krok relaksacji najkrótszych ścieżek, musimy wykonać n takich relaksacji. Złożoność $O(n^4)$. Można poprawić złożoność. Nie da się zastosować algorytmu Strassena, bo nie mamy działania odwrotnego do min. Można za to skorzystać z szybkiego potęgowania, co zmniejsza nam koszt do $O(n^3 \log n)$.

2.26. zadanie 26

Opisz, w jaki spósb obliczenie wielomianu n-tego stopnia w n-tych pierwiastkach jedności jest redukowalne do obliczenia dwóch wielomianów stopnia $n/2$ w $(n/2)$-tych pierwiastkach jedności.

Mamy wielomian $A(x)$ stopnia n, reprezentowany przez współczynniki $a_0, a_1, \cdots, a_n$.
Tworzymy 2 nowe wielomiany:
$A^{[0]}(x) = a_0 + a_2x + a_4x^2 + ...a_{n-2}x^{n/2-1} + a_nx^{n/2}$
$A^{[1]}(x) = a_1 + a_3x + a_5x^2 + ...a_{n-1}x^{n/2-1}$

$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$

2.27. zadanie 27

Opisz ideeę algorytmu klasy NC dla problemu dodawania dwóch liczb n-bitowych.

Powiedzmy, ze mamy dodac 2 liczby n-bitowe i zalozmy ze mamy policzony wektor przeniesien:
$c_{n+1}c_n \dots c_3c_2c_1c_0$
$b_n \dots b_3b_2b_1b_0$
$+a_n...a_3a_2a_1a_0$
wtedy wynik dodawania wyraza sie wzorem $d_i = a_i xor b_i xor c_i$

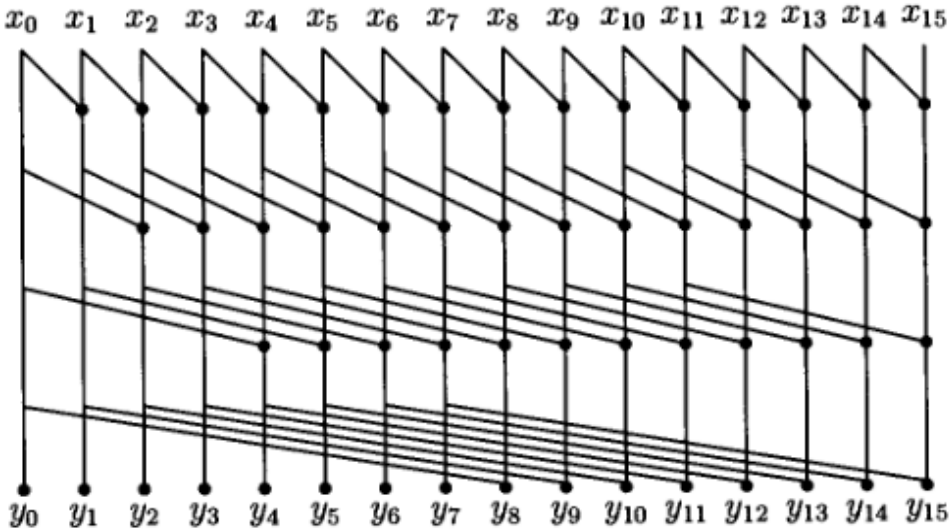
Teraz zobaczmy ze gdy $a_i = b_i = 0$ to $c_{i+1} = 0$ oraz gdy
$a_i = b_i = 1$ to $c_{i+1} = 1$ niezaleznie od tego jaki jest remainder c_i.
Ale takze gdy $a_i = 1 b_i = 0$ lub na odwrot to wtedy $c_{i+1} = c_i$.

Wiec zdefiniujemy sobie dzialanie * na zbiorze X = 0, 1, p
dzialajace dokladnie jak powyzej czyli dla kazdego x ze zbioru X mamy:
0 * x = 0 - ustaw na 0
1 * x = 1 - ustaw na 1
p * x = x - przepisz to co masz z prawej strony
to dokladnie odpowiada trzem przypadkom jakie mielismy powyzej
dzialanie jest lacznie ale nie przemienne.

Teraz z wektorem przeniesien mozemy zwiazac wyrazenie:
czyli gdy mamy $a_i = 1 b_i = 0$ lub na odwrot to $x_i = p$ lub
$a_i = b_i = 1$ to $x_{i+1} = 1$ lub $a_i = b_i = 0$ to $x_{i+1} = 0$

Zatem z wektorem przeniesien zwiazujemy wyrazenie $x_{n+1} * x_n * ... * x_0$ gdzie $x_i \in X$ sa wyliczone jak powyzej.
Jesli moglibysmy policzyc wszystkie $y_i = c_i = x_i * ... * x_0$ w czasie $logn$ z uzyciem wielu procesorow to mielibysmy wektor przeniesien wiec caly problem rozwiazalibysmy w czasie $O(1 + logn)$.

Jak to policzyc jest zaprezentowane na rysunku:


x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}
y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12} y_{13} y_{14} y_{15}

Rozdział 3

II termin 2010 - odsłona 1 i 2

3.1. zadanie 1

Rozwiąż poniższe równanie:

$$T(n) = \begin{cases} c & \text{jeśli n} = 1 \\ 2 \cdot T(n-1) + 1 & \text{jeśli } n > 1 \end{cases}$$

Czy funkcja $T(n)$ jest $O(n^{log_2 n})$?

Rozwiązanie równania rekurencyjnego:

$$\begin{aligned} T(n) &= 2 \cdot T(n-1) + 1 \\ &= 2 \cdot (2 \cdot T(n-2) + 1) + 1 \\ &= 2^2 \cdot T(n-2) + 2 + 1 \\ &= 2^3 \cdot T(n-3) + 2^2 + 2 + 1 \\ &... \\ &= 2^n \cdot T(1) + 2^{n-1} + ... + 2^2 + 2 + 1 \\ &= 2^n \cdot c + 2^n - 1 \end{aligned}$$

Pokażemy że $T(n) \notin O(n^{log_2 n})$:

Z powyżej rozwiązanej rekurencji widać że: $T(n) \in \Omega(2^n)$,

$$\exists_{n_0 \in N} \forall_{n > n_0} n^{log_2 n} = \left(2^{log_2 n}\right)^{log_2 n} = 2^{(log_2 n)^2} < 2^n \tag{3.1}$$

Wynika z tego, że $n^{log_2 n}$ nie jest ograniczeniem górnym.

3.2. zadanie 2

Wyznacz z dokładność do Θ (przy jednorotnym kryterium) poniższego fragmentu algorytmu:

suma $\leftarrow 0$
for $i \leftarrow 1$ *to* n **do**

```
    k ← 1
    while k ⩽ i do
      read(x)
      suma ← suma + x · k
      k ← k + k
    end while
  end for
```

$$\sum_{i=0}^{n} log_2 i = log_2 1 + ... + log_2 n = log_2(1 \cdot ... \cdot n) = log_2(n!)$$

Algorytm jest $\Theta(log_2(n!))$

3.3. zadanie 3

Załóżmy, że w definicji drzewa czerwno-czarnego zmienimy warunek mówiący iż dzieci czerwonego ojca są czarne na warunek:
dzieci czerwonego ojca, którego ojciec też jest czerwony są czarne
Określ jak zmieni się (z dokładnością) do stałego czynnika maksymalna wysokość tak zdefiniowanych drzew?

Przy orginalnym założeniu:

$$h \leqslant 2 \cdot log(n-1)$$

Po zamianie założeniam, minimalna liczba czarnych wierzchołków w każdej scieżce od korzenia do liścia, zmieniła się z $\frac{h}{2}$ na $\frac{h}{3}$. Warunek na h przyjmuje teraz postać:

$$h \leqslant 3 \cdot log(n-1)$$

3.4. zadanie 4

Z ilu drzew może składać sie kopiec dwumianowy (wersja eager) zawierający 49 elementów?

Ponieważ jest to kopiec typu eager nie może miec dwóch drzew dwumianowych tego samego stopnia. Aby przekonać się z ilu drzew dwumianowych się on składa, zamieńmy liczbę jego elementów na liczbe o binarną.

$$49_{10} = 32_{10} + 16_{10} + 1_{10} = 2^5_{10} + 2^4_{10} + 2^0_{10} = 110001_2 \tag{3.2}$$

Widźimy, że kopiec ten składa się z trzech drzew: B_5 , B_4 i B_1.

3.5. zadanie 5

Dla której z poniżej podanych struktur danych koszt (najgorszego przypadku) wykonania operacji $find(i)$ sprawdzającej czy klucz i jest pamiętany w struktrze jest $O(logn)$, gdzie n jest rozmiarem struktury?

(a) drzewo binarnych przeszukiwań,

(b) drzewo AVL

(c) kopiec

(d) kopiec dwumianowy

(e) kopiec Fibonacciego

(f) drzewo czerwono-czarne

Dla struktur b, f koszt wykonania operacji $find(i)$ jest $O(logn)$. Dla pozostałych jest on $O(n)$.

3.6. zadanie 6

Podaj przykłady nietrywialnych zastosowań poniższych algorytmow i struktur danych: kopiec, kopiec Fibonacciego, kopiec dwumianowy, sortowanie leksykograficzne ciągów róznej dlugości

	Przykład zastosowania
kopiec	algorytm sortowanie przez kopcowanie
kopiec Fibonacciego	algorytm Dijskry
kopiec dwumianowy	kolejka priorytetowa
sortowanie leksykograficzne ciągów różnej długości	algorytm rozpoznający czy dwa drzewa są izomorficzne

3.7. zadanie 14

W problemie LCS stosowaliśmy redukcję problemu porównując ostatnie litery X i Y. Czy jakies znaczenie ma fakt, że są to ostatnie litery a nie pierwsze?

Nie jest to istotne.

Weźmy sobie ciągi X, Y i niech algorytm $LCS(X, Y)$, obliczający długość najdłuższego podciągu, wykorzystując redukcję problemu porównując ostatnie litery ciagów. Zaóważmy że algorytm $LCS'(X, Y) = LCS(X^R, Y^R)$ również poprawnie oblicza długośc najdłuższego podciągu ciagów X, Y, ale można powiedzieć że wykorzystuje on redukcję problemu porównując pierwsze litery X, Y.

Rozdział 4

Egzamin 2011 część 1

4.1. zadanie 1

Opisz algrotym mnożenia długich liczb (n bitowych)?

4.2. zadanie 2

Dlaczego porównania współrzędnych nie wystarczają do stiwerdzenia współliniowości punktów?

4.3. zadanie 3

Jak posortwoać n liczb z przedziału $[1, n^2]$ w czasie liniowym?

4.4. zadaine 4

Zapisz procedurę łączenia dwóch kopców dwumianowych typu eager.

4.5. zadanie 5

4.6. zadanie 6

Czemu algorytm magicznych trójek byłby gorszy od algorytmu magicznych piątek?

4.7. zadanie 7

Jak zwiększyć prawdopodobieństwo wylosowania "poprawnej" funkcji w trakcie konsturkcji słownika statycznego?

4.8. zadanie 8

Ile w pesymistycznym przpadku potrzeba rotacji do usunięcia węzła z AVL?

4.9. zadanie 10

Ile warstw musi mieć siec przełączników, żeby dalo się uzyskać wszystkei przesunięcia cykliczne ciągu n-elemntowego?

Bibliografia

[1] test reference