AiSD wszystkie - wersja z dnia 14 czerwca 2018

Spis treści

1 Zadania z części 1.

1.1 Drzewa AVL, B-drzewa

1. (Zad. 12, cz. 1, 09.2017) Niech v będzie wierzchołkiem w drzewie binarnym a Diff(v) niech oznacza wartość bezwzględną różnicy liczby wierzchołków w jego lewym i prawym poddrzewie. Czy w drzewie AVL o n wierzchołkach największa wartość $Diff(v)$ może być $\Omega(n)$? Odpowiedź uzasadnij.

2. (Zad. 9, cz. 1, 09.2017) Rozważamy B drzewa, których wierzchołki mogą pamiętać od dwóch do czterech kluczy. Narysuj, jak będzie wyglądać takie B drzewo po wstawieniu do początkowego pustego drzewa kolejno klucz $1, 10, 3, 8, 5, 6, 7, 4, 9, 2$.

3. (Zad. 12, cz. 1, 06.2017) Jak mocno można ograniczyć (w pesymistycznym przypadku) liczbę rotacji podczas usuwania wierzchołka z drzewa AVL o n wierzchołkach? Uzasadnij, że nie da się bardziej niż podałeś(aś).

4. (Zad. 9, 06.2017) Rozważamy B drzewa, których wierzchołki mogą pamiętać od dwóch do czterech kluczy. Narysuj, jak będzie wyglądać takie B drzewo po wstawieniu do początkowego pustego drzewa kolejno klucz $1, 2, \ldots, 10$.

5. (Zad. 19, cz. 1, 06.2016) Jaką największą wysokość może mieć drzewo AVL zawierające 67 kluczy? Odpowiedź uzasadnij.

   Rozwiązanie:

   Zbudujmy drzewo AVL o maksymalnej wysokości przy użyciu jak najmniejszej ilości wierzchołków. Nazwijmy je T_i Dla $T_0 = 0$ będzie to drzewo puste, dla $T_1 = 1$ będzie to drzewo jednoelementowe, dla $T_2 = 2$ będzie to drzewo składające się z korzenia i jednego syna. Dla T_n, gdzie $n \geq 3$ będzie to drzewo które składa się z korzenia i dwóch poddrzew, którymi są drzewa o wysokości T_{n-1} oraz T_{n-2}. Drzewa te nazywamy drzewami Fibbonacciego. Liczba wierzchołków takiego drzewa to $N_h = N_{h-1}+1+N_{h-2}$. W takim razieiczba wierzchołków kolejnych drzew Fibbonacciego to odpowiednio $0, 1, 2, 4, 7, 12, 20, 33, 54, 88, \ldots$. Drzewo o 67 kluczach będzie miało więc maksymalnie wysokość 8, ponieważ, żeby zbudować drzewo AVL o wysokości 9 potrzeba minimum 88 kluczy.

---

6. (Zad. 14, cz. 1, 06.2015) Jeśli w drzewach AVL zmienilibyśmy warunek, by poddrzewa mogły różnić się o 2 (nie o 1) wysokością, to Czy drzewo n-wierzchołkowe dalej ma wysokość $\Theta(n)$?

1.2 Drzewa Splay

1. (Zad. 3, cz. 1, 09.2017) Czy po wykonaniu operacji $find(x)$ w drzewie splay o n wierzchołkach może się zwiększyć wysokość drzewa?

2. (Zad. 3, cz. 1, 06.2017) Narysuj

   - drzewo *Splay* po wykonaniu na początkowo pustym drzewie ciągu operacji:

     $$insert(n), insert(n-1), insert(n-2), \ldots, insert(1),$$

   - drzewo *Splay* po wykonaniu operacji *Splay(n), Splay(n-1)* na drzewie otrzymanym w poprzednim punkcie

3. (Zad. 6, cz. 1, 06.2016) Czy trójelementowe drzewo złożone z korzenia i dwóch jego synów może być drzewem splay? Odpowiedź uzasadnij.

1.3 Haszowanie

1. (Zad. 6, cz. 1, 09.2017) Na czym polega metoda podwójnego haszowania stosowana do rozwiązywania kolizji podczas haszowania metodą adresowania otwartego?

2. (Zad. 16, cz. 1, 09.2017) Jaka jest oczekiwana liczba kolizji, gdy do umieszczenia 100 różnych kluczy w tablicy 1000 elementowej użyjemy funkcji losow wybranej z uniwersalnej rodziny funkcji haszujących? Odpowiedź uzasadnij.

3. (Zad. 6, cz. 1, 06.2017) Rozważamy haszowanie metodą adresowania otwartego, w której konflikty rozwiązujemy metodą liniową. Pokaż, że po umieszczeniu $n/2$ kluczy w tablicy n elementowej, mogą istnieć dwie lokalizacje w tej tablicy, do których kolejny (tj. $(n/2+1)$szy) klucz ma szansę trafić z prawdopodobieństwem $1/n$.

4. (Zad. 15, cz. 1, 06.2017, Zad. 15, cz. 1, 09.2017) Ile pamięci zajmuje słownik statyczny (oparty o haszowanie dwupoziomowe) zawierający n kluczy? Co musimy w nim pamiętać oprócz samych kluczy?

5. (Zad. 16, cz. 1, 06.2017) Podaj definicję i przykład uniwersalnej rodziny funkcji haszujących.

6. (Zad. 20, cz. 1, 06.2016) Jaka jest oczekiwana liczba kolizji podczas wstawiania n kluczy do tablicy o $m = n^2$ elementach, jeśli do wyznaczania miejsc wstawiania użyjemy funkcji o postaci $h(k) = ((ak+b) \bmod p) \bmod m$, gdzie:

7. (Zad. 12, cz. 1, 06.2015) Oszacuj prawdopodobieństwo, że nie będzie żadnej kolizji podczas haszowania funkcją z uniwersalnej rodziny $\sqrt{n}$ kluczy w tablicy rozmiaru n.

1.4 FFT

1. (Zad. 17, cz. 1, 09.2017) Przekształcenie wektora $\overline{a}$ dokonywane przez algorytm FFT można opisać jako mnożenie pewnej macierzy A przez ten wektor: $\overline{y} = A \times \overline{a}$. Jaka jest wartość j-tego elementu i-tego wiersza tej macierzy? Jeśli $\overline{a}$ jest wektorem współczynników pewnego wielomianu, to czym są składowe wektora $\overline{y}$?

2. (Zad. 17, cz. 1, 06.2017) Algorytm FFT używaliśmy do zamiany reprezentacji wielomianu w reprezentację jako zbiór wartości wielomianu. Uzasadnij, dlaczego FFT możemy także zastosować do zamiany odwrotnej.

3. (Zad. 16, cz. 1, 06.2016) Jak wiadomo FFT jest algorytmem opartym na strategii Dziel i Zwyciężaj. Przedstaw redukcję wykonaną w tym algorytmie.

4. (Zad. 5, cz. 1, 06.2015) Przedstaw macierze dla transformacji Fouriera (???)

1.5 Algorytmy wyszukiwania wzorca: KMP, KMR, Shift-And

1. (Zad. 11, cz. 1, 09.2017) Czy *Shift-And* jest odpowiednim algorytmem wyszukiwania:

   - długich wzorców w tekstach?
   - wzorców w długich tekstach?

   Odpowiedź uzasadnij.

2. (Zad. 8, cz. 1, 09.2017 Zapisz w pseudokodzie funkcję obliczającą funkcję π z algorytmu KMP. Podaj, jaki jest jej czas działania. Podaj uzasadnienie swego stwierdzenia.

3. (Zad. 11, cz. 1, 06.2017) W algorytmie *Shift And* wykorzystywane są operacje logiczne na słowach maszynowych. Wytłumacz, w jaki sposób?

4. (Zad. 8, cz. 1, 06.2017) Czy istnieje wzorzec o długości n (dla dowolnego $n > 5$) nad alfabetem $\{a, b\}$, dla którego maksymalna wartość funkcji prefiksowej π jest równa

   a 0,

   b 1?

5. (Zad. 8, cz. 1, 06.2016) Dlaczego algorytm *Shift-And* stosowany jest jedynie do wyszukiwania krótkich wzorców?

6. (Zad. 10, cz. 1, 06.2016) Jaka jest największa wartość funkcji π dla wzorca $P = (ab)^k$? Odpowiedź uzasadnij.

7. (Zad. 1, cz. 1, 06.2015) Podaj przykład tekstu i wzorca dla których tablica C[0] = C[1] = C[9] = prawda, a dla pozostałych fałsz. Algorytm Shift-And.

8. (Zad. 2, cz. 1, 06.2015) Jak w KMR numeruje się słowa o długości 16?

9. (Zad. 4, cz. 1, 06.2015) Uzasadnij, że obliczenie funkcji pi(Wzorzec[1..m]) w algorytmie KMP ma złożoność O(m).

1.6 Algorytmy klasy NP i NC

1. (Zad. 14, cz. 1, 09.2017) Czy poniższy problem:

   - należy do klasy NP?
   - jest problemem NP-zupełnym?
   - jest problem NP-trudnym?

   PROBLEM:
   DANE ciąg liczbowy $A = a_1, \ldots, a_n$ oraz liczba naturalna n
   WYNIK 1 - jeśli w A istnieje podciąg rosnący o długości co najmniej k; 0 - w p.p.

2. (Zad. 5, cz. 1, 06.2016) Opisz ideę algorytmu klasy NC dla problemu dodawania liczb naturalnych.

3. (Zad. 15, cz. 1, 06.2016) Podaj definicję problemu plecakowego z powtórzeniami i przedstaw pseudowielomianowy algorytm rozwiązujący ten problem. Uzasadnij, że jest on pseudowielomianowy.

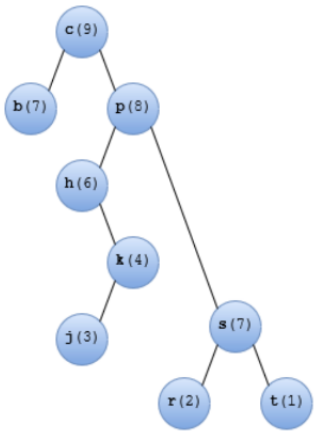
4. (Zad. 13, cz. 1, 06.2015) Podaj pseudowielomianowy algorytm, który wypisuje dzielniki pierwsze liczby n.

1.7 Drzewce

1. (Zad. 4, cz. 1, 09.2017) Ile różnych drzewców można zbudować dla ciągu $(k_1, p_1), (k_2, p_2), \ldots, (k_n, p_n)$, gdzie k_i są kluczami, a p_i - parami różnymi priorytetami? Odpowiedź uzasadnij.

2. (Zad. 4, cz. 1, 06.2017) Podaj przykład drzewca (tj. podaj wartość kluczy wraz z przydzielonymi im priorytetami) o n wierzchołkach, w którym każdy wierzchołek wewnętrzny ma tylko prawego syna. Następnie podaj, który wierzchołek będzie wymagał najwięcej rotacji podczas ustawiania go. Ile to będzie rotacji? Odpowiedź uzasadnij.

3. (Zad. 4, cz. 1, 06.2016) Narysuj ciąg rotacji, które zostaną wykonane w trakcie wykonywania **delete(p)** na poniższym drzewcu. Litery w wierzchołkach drzewca oznaczają klucze, a liczby w nawiasach - priorytety. Rotacje wypisz w kolejności wykonywania.



Rysunek 1: Drzewiec dla zadania 4.

Rysunek 1: rys do zad 4

4. (Zad. 8, cz. 1, 06.2015) Przedstaw drzewiec o n wierzchołkach, w którym usunięcie korzenia wymaga $\Omega(\sqrt{n})$ operacji, ew. podaj uzasadnienie dlaczego nie ma takiego drzewca.

1.8 Drzewa decyzyjne, gra z adwersarzem

1. (Zad. 5, cz. 1, 09.2017) Na wykładzie pokazaliśmy, że $\Omega(n \log(n))$ jest dolną granicą na złożoność 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?

2. (Zad. 5, cz. 1, 06.2017) Dolną granicę $\lceil \frac{3}{2}n-2 \rceil$ na liczbę porównań niezbędnych do wyznaczenia *max* i *min* w zbiorze n elementów można wykazać stosując grę z 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.

1.9 Algorytmy wyboru k-tego elementu (mediany)

1. (Zad. 7, cz. 1, 09.2017) Jaką złożoność ma algorytm *Quicksort*, w którym pivot wybierany jest algorytmem magicznych piątek?

2. (Zad. 7, cz. 1, 06.2017) Opisz w jaki sposób wybierany jest *pivot* w każdym z następujących z następujących algorytmów znajdowania k-tego elementu:

   - Algorytm Hoare'a,
   - Algorytm Magicznych Piątek,
   - Lazy Select.

3. (Zad. 7, cz. 1, 06.2016) Opisz ideę algorytmu znajdowania mediany opartego na idei próbkowania losowego.

4. (Zad. 16, cz. 1, 06.2015) Przerób kod QuickSorta na QuickSelect (selekcja k-tego elementu zamiast sortowania). Jaką ma złożoność?

5. (Zad. 19, cz. 1, 06.2015) Podaj wzór rekurencyjny algorytmu magicznych piątek dla podziału na 7 elementów.

1.10 Izomorfizm drzew

1. (Zad. 10, cz. 1, 06.2017) Porównaj trudność problemu sprowadzania izomorfizmu drzew ukorzenionych i problemu sprawdzaniaa izomorfizmu drzew nieukorzenionych.

2. (Zad. 13, cz. 1, 06.2016) Przedstaw ideę szybkiego algorytmu sprawdzania izomorfizmu drzew. W jakim czasie działa ten algorytm?

1.11 Geometria obliczeniowa

1. (Zad. 14, cz. 1, 06.2017) W jaki sposób, stosując iloczyn wektorowy można sprawdzić, czy dwa punkty (powiedzmy p_1 i p_2) leżą po tej samej stronie prostej przechodzącej przez dwa punkty (powiedzmy A i B)?

1.12 Union Find

1. (Zad. 18, cz. 1, 09.2017) Podaj sensowne ograniczenie górne na największą wysokość drzewa, które może powstać w wyniku wykonania w sposób zbalansowany n operacji *union* na rozłącznych zbiorach (początkowo zbiory te są jednoelementowe). Udowodnij swoje stwierdzenie.

2. (Zad. 18, cz. 1, 06.2017) W analizie problemu *Union Find* wykorzystywaliśmy pojęcie rzędu wierzchołka oraz grupy rzędu. Przypomnij definicję tych pojęć. Ile maksymalnie bitów potrzebujemy przeznaczyć na pamiętanie rzędu w każdym wierzchołku?

3. (Zad. 14, cz. 1, 06.2016) W jakim czasie można wykonać ciąg n operacji **union** i **find**, w którym wszystkie operacje **union** poprzedzają operację **find**? Odpowiedź uzasadnij.

4. (Zad. 6, cz. 1, 06.2015) Podaj definicje: rząd wierzchołka i grupa rzędu wierzchołka

1.13 Drzewa Van Emde Boasa

1. (Zad. 19, cz. 1, 06.2017, Zad. 19, cz. 1, 09.2017) Wyjaśnij (możliwie precyzyjnie) po co pamiętane są wartości *min* i *max* w każdej strukturze rekurencyjnej w drzewach (kolejkach) van Emde Boasa.

2. (Zad. 9, cz. 1, 06.2016) Opisz (albo zapisz w pseudokodzie), w jaki sposób wykonywana jest operacja wstawiania klucza w drzewie van Emde Boasa.

1.14 Kopce - zwykłe, dwumianowe, Fibonacciego

1. (Zad. 2, cz. 1, 09.2017) O ile może zmienić się liczba drzew w kopcu Fibonacciego, zawierającym n kluczy, podczas wykonywania operacji $decreasekey(k, \Delta)$?

2. (Zad. 13, cz. 1, 09.2017) Opisz w jaki sposób wykonywana jest operacja *deletemin* na kopcu dwumianowym (w wersji lazy). Jaki jest jej koszt?

3. (Zad. 2, cz. 1, 06.2017) Ile maksymalnie operacji *join* wykona się podczas łączenia kopców dwumianowych (wersja eager), z których każdy zawiera nie więcej niż 500 elementów? Przypomnienie: operacja *join* łączy dwa drzewa dwumianowe tego samego rzędu.

Rozwiązanie:

Kopiec dwumianowy składa się ze zbioru pełnych drzew Fibonacciego, w którym nie może być dwóch drzew tego samego rzędu. Drzewu rzędu 0 ma jeden wierzchołek, drzewo rzędu 1 ma dwa wierzchołki, drzewo rzędu k ma 2^k wierzchołków. Największym drzewem Fibonacciego, które mogłoby się znaleźć w kopcu o 500 elementach jest B_8, które ma 256 wierzchołków. Jednak nie możemy wtedy posiadać w kopcu wszystkich mniejszych, drzew Fibonacciego, gdyż suma ich wierzchołków wynosiłaby 511. Zapisując liczbę drzew w kopcu jak liczbę binarną (1 - występuje drzewo o danym rozmiarze, 0 - nie występuje) możemy zauważyć, że liczba operacji *join* jest równa przenoszeniu jedynki podczas dodawania dwóch takich liczb. Weźmy więc kopiec o 385 wierzchołkach, jego zapis binarny będzie wyglądał następująco 110000001. Dodajmy do tego drzewo binarne o 255 wierzchołkach, którego zapis binarny wygląda następujaco 011111111. Wtedy podczas połączenia tych dwóch kopców będziemy przenosić jedynkę (czyli wykonywać operację join) 9 razy. Więcej operacji nie możemy wykonać, ponieważ drzewo B_9 posiada 512 wierzchołków i nie mieści się w podanym zakresie.

---

4. (Zad. 13, cz. 1, 06.2017) Niech T_i oznacza najmniejsze pod względem liczby wierzchołków drzewo o rzędzie i, które może zawierać kopiec Fibonacciego. Narysuj drzewa T_i, dla $i = 0, 1, \ldots, 6$.

5. (Zad. 1, cz. 1, 06.2016) W jakim czasie można wykonać operację succ(x) w:

   - kopcu,
   - kopcu dwumianowym,
   - kopcu Fibonacciego,

   która znajduje następnik klucza znajdującego się w wierzchołku o adresie x? Przez następnik klucza k rozumiemy najmniejszy występujący w kopcu klucz k' taki, że k' taki, że $k' > k$. Jeśli$k$ jest największym kluczem w kopcu, to $k' = \infty$. Możesz założyć, że wszystkie klucze w kopc są unikalne. Odpowiedź uzasadnij.

6. (Zad. 17, cz. 1, 06.2016) Napisz w pseudokodzie szybką procedurę budowy kopca. W jakim czasie działa ta procedura?

Rozwiązanie:

buduj˙kopiec(K)
for $i = \lfloor \frac{n}{2} \rfloor$ to 1
przesun-w-dol(K,i)

Dla uproszczenia ustalmy, że n jest postaci 2^k. Kopiec budujemy od dołu, ma on wysokość $log(n)$. Zaczynamy od rzędu $log(n) - 1$, który ma $\frac{n}{4}$ wierzchołków. Budujemy więc $\frac{n}{4}$ kopców o wysokości 2, przesuniemy więc wierchołek maksymalnie o 1 w dół. Operacja przesuń niżej wykonuje 2 porównania, co daje nam w sumie $\frac{n}{4} \times 2 \times 1 = \frac{n}{2}$ porównań. Następny poziom $log(n) - 2$ ma $\frac{n}{8}$ wierzchołków. Podstruktura kopca jest zachowana, więc jedynym miejscem, które trzeba naprawić jest korzeń. Liczbę znajdującą się w korzeniu przesuniemy maksymalnie o dwa poziomy, co daje nam $\frac{n}{8} \times 2 \times 2 = \frac{n}{2}$ porównań. Przy pierwszym poziomie wykonamy maksymalnie $2 \times log(n)$ porównań dla jednego wierzchołka. Całość da nam złożoność $\Theta(n)$

---

7. (Zad. 18, cz. 1, 06.2016) Wyjaśnij, na czym polega operacja kaskadowego odcinania w kopcu Fibonacciego.

Rozwiązanie:

Kaskadowe odcięcie polega na odcięciu wierzchołka p z kopca h. Odcinamy poddrzewo z korzeniem p z kopca i dołączamy do listy drzew kopca. Następnie sprawdzamy czy ojciec p, nazwijmy go p' utracił już jakiegoś syna. Jeżeli tak, to odcinamy również p' i rekurencyjnie sprawdzamy jego ojca, aż znajdziemy wierzchołek, który nigdy nie stracił syna, lub korzeń. Jeżeli p' nigdy nie stracił syna, to zapamiętamy informację o utracie syna.

---

8. (Zad. 10, cz. 1, 06.2015) Ile jest maksymalnie drzew w kopcu:
   - Fibonacciego,
   - dwumianowym.

Rozwiązanie:

- Redukcja liczby drzew wykonuje się dopiero podczas operacji *delete-temin*. Operacja insert dodaje drzewo Fibonacciego rozmiaru 0 (jednowierzchołkowe). Stąd maksymalna liczba drzew Fibonacciego dla n-elementowego kopca będzie wynosić n

- W wersji lazy będzie tak samo jak w kopcu Fibonacciego. W wersji Eager może zawierać maksymalnie $O(log(n))$ drzew, którymi będą kolejne pełne drzewa Fibonacciego.

---

9. (Zad. 11, cz. 1, 06.2015) Złożoność procedury budującej kopiec (wersja z *przesun-do-gory()*).

Rozwiązanie:

Operacja *przesun-do-gory()* kosztuje nas $O(log(n))$. Kopiec rozszerzamy od 1-elementowego do n-elementowego, stąd $O(nlog(n))$

---

10. (Zad. 17, cz. 1, 06.2015) Porównanie kosztów operacji findmin, delmin, insert, meld dla kopców dwumianowych w wersji Lazy i Eager.

Rozwiązanie:

Eager:

- *findmin* - trzymamy przy kopcu wskaźnik MIN wskazujący minimalny element. Koszt $O(1)$.
- *meld* - przywracamy strukturę kopca natychmiast. Drzewa kopca dostępne są przez tablicę wskaźników. Każdy kopiec zawiera co najwyżej jedno drzewo każdego rzędu. Kopiec zawierający n elementów składa się z co najwyżej $log(n)$ różnych drzew dwumianowych. Koszt $O(log(n))$.
- *deletemin* - wskaźnik MIN wskazuje na drzewo dwumianowe B, które zawiera najmniejszy element. Usunięcie B to czas stały, otrzymujemy zbiór drzew B_0, B_1, $B_{d(B)-1}$, który tworzy nowy kopiec dwumianowy. Łączymy kopce za pomocą operacji meld. Koszt $O(log(n))$.
- *insert* - TODO $O(1)$ zamortyzowany

Lazy :

- *findmin* - tak jak w Eager $O(1)$

- *meld* - Drzewa kopca przechowywane są jako lista cykliczna, dwukierunkowa. Łączymy dwie listy i aktualizujemy wskaźnik MIN. Czas $O(1)$, ale nie zachowujemy struktury kopca i możemy mieć kilka drzewa tego samego rozmiaru
- *insert* - czas $O(1)$ tworzymy nowe drzewo i dodajemy, ew. aktualizujemy wskaźnik MIN
- *deletemin* Pojedyncza operacja *deletemin* może kosztować nawet $O(n)$ (gdy kopiec składa się z drzew jednoelementowych). Czas zamortyzowany można ograniczyć przez $O(log(n))$. Analiza w notatkach.

---

11. (Zad. 18, cz. 1, 06.2015)

    - Podaj definicje rzędu drzewa w kopcu Fibbonaciego,
    - Podaj górne ograniczenie na ten rząd,
    - Podaj ideę dowodu tego ograniczenia.

    Rozwiązanie:

    - r - korzeń, d - rząd, T - drzewo $d(T) = d(r(T))$, gdzie $r(x)$ = liczba dzieci x-a
    - $O(log(n))$
    - TODO: w notatkach.

1.15 Algorytmy znajdowania MST

- (Zad. 20, cz. 1, 09.2017) Pokaż, że istnieją n wierzchołkowe grafy (n jest dowolnie duże), dla których algorytm Boruvki znajduje minimalne drzewo spinające wykonując dokładnie dwie fazy.
- (Zad. 11, cz. 1, 06.2016) W jakim czasie działa algorytm Kruskala, jeśli:
  - krawędzie podane są w kolejności rosnących wag„
  - kolejka priorytetowa zaimplementowana jest przy pomocy kopca Fibonacciego.

Odpowiedź uzasadnij. *Uwaga: Oba te warunki są spełnione jednocześnie*

- (Zad. 20, cz. 1, 06.2015) Przykład grafu pełnego o n wierzchołkach takiego, że algorytm Boruvki znajdzie MST w jednej fazie.

1.16 Różne algorytmy

1. (Zad. 10, cz. 1, 09.2017) Podaj definicję problemu plecakowego (wersja bez powtórzeń), a następnie podaj dla niego algorytm zachłannny działający w czasie pseudowielomianowym. Uzasadnij pseudowielomianowość podanego algorytmu.

2. (Zad. 1, cz. 1, 06.2017) Opisz algorytm oparty na programowaniu dynamicznym wyznaczający optymalną kolejność mnożenia macierzy. Jaka jest jego złożoność? Jeśli jest to algorytm podany na wykładzie, możesz na tym poprzestać, w przeciwnym razie uzasadnij jego poprawność i złożoność.

3. (Zad. 1, cz. 1, 09.2017) Opisz algorytm oparty na programowaniu dynamicznym wyznaczający najdłuższy wspólny podciąg dwóch ciągów. Jaka jest jego złożoność? Jeśli jest to algorytm podany na wykładzie, możesz na tym poprzestać. W przeciwnym razie uzasadnij jego poprawność i złożoność.

4. (Zad. 20, cz. 1, 06.2017) Przypomnij sobie algorytm oparty na zasadzie *Dziel i zwyciężaj*, dla problemu znajdowania najbliższej pary punktów na płaszczyźnie. Opisz trzecią fazę algorytmu, a więc tę, która następuje po wywołaniach rekurencyjnych. Jaka jest jej złożoność?

5. (Zad. 12, cz. 1, 06.2016) Zapisz w pseudokodzie algorytm wielomianowy, znajdujący minimalny koszt obliczenia iloczynu ciągu macierzy.

6. (Zad. 15, cz. 1, 06.2015) Podaj pseudokod rozwiązania problemu LCS.

1.17 Inne

1. (Zad. 2, cz. 1, 06.2016) Rozwiąż rozwiązanie rekurencyjne (z redukcją do pierwiastka):
$$T(n) = \begin{cases} 1 & : \quad n = 1 \\ T(\sqrt{n} + O(1) & : \quad wpp \end{cases}$$
Możesz ograniczyć się do rozwiązania dla n mających odpowiednią postać (taką, by w trakcie redukcji argumenty dla T były liczbami naturalnymi).

2. (Zad. 3, cz. 1, 06.2016) Narysuj sieć Benesa-Waksmana dla n = 8.

3. (Zad. 3, cz. 1, 06.2015) Przedstaw graficznie sieć komparatorów o głębokości ≤ 4 sortującej wszystkie ciągi 0-1 o długości 7.

4. (Zad. 9, cz. 1, 06.2015) Podaj rekurencyjny wzór na T(n) tak, by jego rozwiązanie było $O(\log \log n)$.

5. (Zad. 7, cz. 1, 06.2015) Czy drzewa A i B będą miały równą wysokość, jeśli przeprowadzi się na nich n operacji insert o wartościach:

   - dla A: 1, 2, 3, ..., n
   - dla B: n, n-1, ..., 2, 1

2 Zadania z części 2.

1. (Zad. 1, cz. 2, 06.2017) Przypomnijmy problem przechodzenia przez tablicę:
   Problem (Przejście przez tablicę)
   Dane: Tablica $\{i, j\}$ liczb nieujemnych $(i = 1, \ldots, n; j = 1, \ldots, m)$
   Wynik: Ciąg indeksów $i_1, \ldots, i_n$ taki, że $\forall_{j=1,\ldots,m-1} 1 \leq i_j \leq n$ oraz $|i_j - i_{j+1}| \leq 1$ maksymalizujący sumę $\sum_{j=1}^{m} a_{i_j,j}$

   Ułóż algorytm dla następującego uogólnienia tego problemu:
   Problem

   Dane: Tablica $\{a_{i,j}\}$ liczb nieujemnych $(i = 1, \ldots, n; j = 1, \ldots, m)$
   Wynik: Dla każdych i, j takich, że $i = 1, \ldots, n; j = 1, \ldots, m$ należy podać największą wartość, po dodaniu której do $a_{i,j}$ wartość rozwiązania *problemu przejścia przez tablicę* nie ulegnie zmianie.

2. (Zad. 2, cz. 2, 06.2017) Dla zbiorów n chłopców i n dziewcząt chcemy znaleźć skojarzenia między nimi (tj. zbiór n par (chłopiec, dziewczynka)), które minimalizują średnią różnicę wzrostu w parze. Dokładniej: dane są dwa n - elementowe ciągi liczb naturalnych $C = c_1, c_2, \ldots, c_n$ oraz $D = d_1, d_2, \ldots, d_n$, gdzie c_i jest wzrostem i-tego chłopca, a d_i wzrostem i-tej dziewczynki (abstrahujemy od rzeczywistości c_i oraz d_i mogą być dowolnie duże). Chcemy znaleźć taką n-elementową permutację Π, że wartość

   $$\frac{\sum_{i=1}^{n} |c_i - d_{\Pi(i)}|}{n}$$

   jest minimalna.

   - Rozwiąż następujący algorytm zachłanny: w i-tym kroku $(i = 1, 2, \ldots, n)$ spośród nieskojarzonych jeszcze chłopców i dziewcząt kojarzymy parę o najmniejszej różnicy wzrostów. Czy ten algorytm jest poprawny? Udowodnij swoje stwierdzenie.
   - Podaj lepszy algorytm dla tego problemu , tzn. poprawny w przypadku, gdy powyższy jest niepoprawny, bądź, szybszy (i oczywiście poprawny) w przypadku, gdy powyższy jest poprawny. Udowodnij poprawność swojego algorytmu i podaj złożoność.

Rozwiązanie:

Najpierw zauważmy, że n w mianowniku jest niezależne od parowania, należy więc znaleźć parowanie które da nam jak najniższą sumę. Rozważmy podany algorytm zachłanny. Weźmy najmniejszy sensowny ciąg, czyli $n = 2$ i spróbujmy znaleźć przykład dla którego algorytm jest niepoprawny. Powiedzmy, że $d_1 = 5$, $d_2 = 50$, $c_1 = 53$, a $c_2 = 90$. Podany algorytm musiałby wyliczyć wszystkie różnice między każdym chłopcem i dziewczynką, a następnie wybrać najmniejszą i usunąć daną parę. Wyliczenie wszystkich różnic to koszt $n \times n$, a dodatkowo dla podanego przykładu wynik algorytmu to $|c_1 - d_2| + |d_1 - c_2| = 3 + 90 = 93$ podczas gdy wybranie innych par, czyli $|d_1 - c_1| + |d_2 - c_2| = 48 + 40 = 88$ da nam niższą sumę. Bez problemu znaleźliśmy więc kontrprzykład dla którego algorytm z zadania nie działa.
Na podstawie podanego algorytmu można zauważyć, że powinno dać się znaleźć mniej kosztowny sposób wybierania oraz trzeba wykazać poprawność wybieranej pary. Poprawne rozwiązanie dla wymyślonego wcześniej przykładu działa dla wyboru najwyższego chłopca i najwyższej dziewczynki, przeanalizujmy czy wybór takiej pary jest optymalny. Załóżmy, że chłopcy dziewczynki zostali ustawieni w dwóch szeregach, takich, że: $c_1 > c_2 > \ldots > c_n$ oraz $d_1 > d_2 > \ldots > d_n$, gdzie c_1 i d_1 to odpowiednio najwyższy chłopiec i dziewczynka. Bez straty ogólności załóżmy, że $c_1 > d_1$ (interesuje nas wartość bezwzględna, a chłopcy zazwyczaj są wyżsi niż dziewczynki). Wtedy mogą być następujące przypadki.

1. $c_1 \geq d_1 \geq c_i \geq d_i$
2. $c_1 \geq d_1 \geq d_i \geq c_i$
3. $c_1 \geq c_i \geq d_1 \geq d_i$

gdzie $2 \leq i \leq n$.
Wtedy dla kolejnych przypadków należy udowodnić, że $|d_1 - c_1| + |d_i - c_i| \geq |d_1 - c_i| + |d_1 - c_i|$:

1. Niech $|c_1 - d_1| = p$, $|d_1 - c_i| = q$, $|c_i - d_i| = r$. Więc $|d_1 - c_1| + |d_i - c_i| = p + r$, natomiast $|d_1 - c_i| + |d_1 - c_i| = q + p + q + r = p + 2q + r$. Czyli dla tego przypadku nierówność jest zachowana.
2. Niech $|c_1 - d_1| = p$, $|d_1 - d_i| = q$, $|d_i - c_i| = r$. Więc $|d_1 - c_1| + |d_i - c_i| = p + r$, natomiast $|d_1 - c_i| + |d_1 - c_i| = q + r + p + q = p + 2q + r$. Również zachowana.
3. Niech $|c_1 - c_i| = p$, $|c_i - d_1| = q$, $|d_1 - d_i| = r$. Więc $|d_1 - c_1| + |d_i - c_i| = p + q + q + r$, natomiast $|d_1 - c_i| + |d_1 - c_i| = p + q + r + q = p + 2q + r$. W tym przypadku wybór będzie taki sam, ale na pewno nie będzie gorszy.

Z powyższego rozumowania wynika, że wybór najwyższej pary da zawsze lepszy lub równy wynik do wyboru innej dowolnej pary. Po wyborze pary mamy zbiory o długości $n-1$ i możemy zastosować algorytm ponownie. Pozostaje problem znalezienia najwyższej pary, możemy to osiągnąć w czasie $O(n \log_n)$ (TODO: czy da się posortować w czasie liniowym?). Dla posortowanego zbioru czas wykonania algorytmu to $O(1)+T(n-1)$ co po rozwiązaniu zależności daje czas $O(n)$. Ostateczny czas działania algorytmu to $O(n \log_n)$ (jeżeli da się kubełkowo to $O(n)$).

---

3. (Zad. 3, cz. 2, 06.2017) Skonstruuj strukturę danych, która umożliwia efektywne wykonywanie następujących operaci na n, początkowo wyzerowanych, licznikach:

   - *inc(k)* - zwiększ o 1 wartość k-tego licznika
   - *settomax()* - ustaw wszystkie liczniki na maksymalną (w momencie wykonywania tej operacji) wartość licznika
   - *get(k)* - wypisz wartość k-tego licznika

3 Zadania z części 3.

1. (Zad. 1, cz. 3, 06.2017) Ciąg nazywamy nienudnym, jeżeli każdy jego spójny podciąg zawiera co najmniej jedne unikalny element (tzn. występujący w tym podciągu dokładnie raz). Ułóż algorytm, który dla danego ciągu liczb naturalnych sprawdzi, czy jest on nienudny.

2. (Zad. 2, cz. 3, 06.2017) Wariancją ciągu liczbowego $A = \langle a_1, ..., a_n \rangle$ nazywamy liczbę

$$V_A = \begin{cases} 0 & gdy \quad n = 1 \\ \sum_{i=1}^{n-1} |ai + 1 - a_i| & gdy \quad n > 1 \end{cases}$$

   ułóż algorytm, który dla zadanego ciągu liczbowego A znajdzie podział zbioru indeksów $\{1, 2 \ldots, n\}$ na dwa rozłączne podzbiory $I = \{i_1, i_2 \ldots, i_k\}$ oraz $J = \{j_1, j_2 \ldots, j_{n-k}\}$ takie, że:

   - $i_1 < i_2 < \ldots < i_l$, $j_1, j_2, \ldots, j_{n-k}$ oraz $I \cup J = \{1, 2 \ldots, n\}$
   - suma $V_{A_I} oraz V_{A_J}$ jest minimalna,

   gdzie $A_I = \langle x_{i_1}, \ldots, x_{i_k} \rangle$, a $A_J = \langle x_{j_1}, \ldots, x_{k_{n-k}} \rangle$,

3. (Zad. 3, cz. 3, 06.2017) Rozważamy ciąg operacji *Insert(i)*, *DeleteMin* oraz *Min(i)* wykonywanych na S-podzbiorze zbioru $\{1, \ldots, n\}$. Obliczenia rozpoczynami z $S = \phi$. Instrukcja *Insert(i)* wstawia liczbę i do S. Instruckja *DeleteMin* wyznacza najmniejszy element w S i usuwa go z S. Natomiast wykonanie *Min(i)* polega na usunięciu z S wszystkich liczb mniejszych od

i.
Niech ρ będzie ciągiem instrukcji $Insert(i)$, $DeleteMin$ oraz $Min(i)$ takimi, że dla każdego i, $1 \leq i \leq n$, instrukcja $Inser(i)$ występuje co najwyżej jeden raz. Mając dany ciąg ρ naszym zadaniem jest znaleźć ciąg liczb usuwanych kolejno przez instrukcję $DeleteMin$. Podaj algorytm rozwiązujący to zadanie.
Uwaga: *Zakładamy, że cały ciąg ρ jest znany na początku, czyli interesuje nas wykonanie go off-line*

4. (Zad. 4, cz. 3, 06.2017) Rozpiętością ciągu liczbowego $A = \langle a_1, ..., a_n \rangle$ nazywamy liczbę:

$$Span(A) = max\{a_i | i = 1, \ldots, n\} - min\{a_i | i = 1, \ldots, n\}$$

Ułoż algorytm obliczający

$$\sum_{1 \leq p \leq k \leq n} Span(A_p^k)$$

gdzie $A_p^k = \langle a_p, \ldots, a_k \rangle$