Algorytmy i struktury danych

Proste algorytmy

1. Niech wielokąt będzie kołową podwójnie linkowaną listą współrzędnych wierzchołków. Napisz funkcje do obliczania:

   (a) pola tego wielokąta;

   (b) obwodu;

   (c) sprawdzania czy jest to wielokąt wypukły.

2. W pliku "a.txt" znajdują się współczynniki wielomianu $a(x)$ w następującej postaci: $(a_0, a_2, a_3, .., a_n)$ a w pliku "b.txt" współczynniki wielomianu $b(x)$ $(b_0, b_2, b_3, .., b_n)$. Napisz program, który odczyta dane z tych plików, obliczy iloczyn wielomianów $c(x) = a(x)b(x)$ a następnie współczynniki wielomianu $c(x)$ wpisze do pliku "c.txt".

3. Stwórz klasę `ułamek` z polami `licznik` i `mianownik`. Powinien to być zawsze ułamek nieskracalny. Zaimplemenutuj 4 działania (jako operatory `+ - * /`) jednoargumentowy operator `-` oraz czytanie ze strumienia i pisanie do strumienia ułamków typu 1/2, 4/5, 77/3 itp. Napisz też kontruktor dwuargumentowy, gdzie drugi argument ma wartość domyślną 1.

4. Napisz program nie zawierający instrukcji `if` ani `switch`, który policzy ile razy występuje każdy znak ASCII w pliku podanym jako argument programu.

5. Napisz funkcję obliczającą x^n w czasie $O(\log n)$ w wersjach z użyciem i bez użycia rekurencji.

6. Przepisz klasy: kolejka, stos, kolejka_priorytetowa jako szablony.

7. Jak obliczyć wszystkie współczynniki:

   $$\binom{n}{0}, \binom{n}{1}, \ldots \binom{n}{n}$$

   w łącznym czasie $O(n)$.

8. Ile potrzeba mnożeń, aby wyliczyć x^n? Napisz odpowiednią funkcję.

9. Ile potrzeba mnożeń, aby wyliczyć wartość wielomianu stopnia n, o współczynnikach zawartych w tablicy `a`?
   Napisz funkcję `double oblicz(double a[], int n, double x)` realizującą twój algorytm.

10. Napisz procedurę `sito_erastotenesa` znajdującą wszystkie liczby pierwsze mniejsze od n. Wynikiem jest tablica `bool prime[n]`, taka że `prime[k]==true` wtedy i tylko wtedy gdy `k` jest liczbą pierwszą.

11. Jaka jet najmniejsza liczba mnożeń potrzebna do obliczenia wartości wielomianu $W(x)$ stopnia n dla pewnego x. Napisz odpowiednią funkcję, zakładając, że wielomian jest reprezentowanuy jako tablica współczynników. tzn.

$$W(x) = \texttt{t[0]} + \texttt{t[1]}x + \texttt{t[2]}x^2 + \ldots + \texttt{t[n]}x^n.$$

12. Pokaż, że wystarczą 3 mnożenia, by znaleźć współczynniki wielomianu $W(x) = (ax + b)(cx + d)$.

13. (a) Ile jest permutacji zbioru n-elementowego?
    (b) Ile co najmniej pytań trzeba zadać by „zgadnąć" określoną permutację?
    (c) Udowodnić korzystając ze wzoru Stirlinga, że liczba ta jest rzędu $O(n \log_2 n)$.

14. Napisz w języku `C++` klasę `set`, reprezentującą zbiór liczb naturalnych nie większych od 255. Rozmiar obiektu klasy `set` nie powinien przekraczać 32 bajtów.

15. Każdy podzbiór zbioru liczb naturalnych nie wiekszych od 8 można reprezentować na jednym bajcie (i-ty bit jest ustawiony gdy i należy do podbioru). Zaimplementuj w C++ klasę `zbior_malych_liczb`, która z podobny sposób na $(n + 1)/8$ bajtach będzie reprezentować zbiory liczb naturalnych nie większych od n. Zaprogramuj:
    (a) dodawanie elementu do zbioru, usuwanie elementu ze zbioru i sprawdzanie czy liczba należy do zbioru.
    (b) sumę, iloczyn, różnicę zbiorów.
    (c) konstruktor, taki że wywołanie `zbior_malych_liczb a("1,2,4,8");` utworzy zbiór $a = \{1, 2, 4, 8\}$.
    (d) operator przypisania,

(e) sprawdzanie relacji równości i zawierania zbiorów (można przeciążyć operatory `==`, `<=` i `>=`.

16. (Sito Erastotenesa) Algorytm, który bez użycia dzielenia znajduje wszystkie liczby pierwsze nie większe od n wypełniając tablicę n elementową jedynkami, a następnie zerując wszystkie elementy tablicy o indeksach podzielnych przez 2, 3, ... $\sqrt{n}$ natrafia dla dużych n na problem braku pamięci. Można górną granicę na n zwiększyć 8-krotnie korzystając z wyników poprzedniego zadania (tablicy bitów).

17. Zaimplementuj kolejkę FIFO przy pomocy: a) listy linkowanej; b) tablicy.

18. Zaimplementuj kolejkę dwukierunkową przy pomocy listy podwójnie linkowanej. Kolejka dwukierunkowa pozwala na operacje: `push_front`, `push_back`, `pop_front`, `pop_back`. Użyj wartownika.

19. Udowodnij, że każdy wielomian stopnia k jest rzędu $O(n^k)$.

20. Udowodnij, że $\log n$ jest rzędu $O(n^\epsilon)$ dla każdego $\epsilon > 0$.

21. Dla wyszukiwania liniowego w tablicy nieposortowanej o n elementach, znajdź oczekiwaną oraz wariancję ilości porównań wykonanych przez program, przy założeniu, że wyszukiwany element jest w tablicy.

22. Zakładając, że każda permutacja danych wejściowych jest jednakowo prawdopodobna, znajdź wartość oczekiwaną i wariancję ilości inwersji występujących w tych danych.

23. Dana jest funkcja `double f(double)`ciągła, taka że $f(0) < 0 < f(1)$. Napisz program, który metodą bisekcji znajdzie pierwiastek funkcji `f`. Warunkiem zakończenia pętli uczyń wykrycie zapętlenia :).

Obliczanie złożoności

24. W pliku jest 10^6 liczb. Ile potrzeba pamięci i dodawań by sprawdzić, która z sum 1000 kolejnych liczb jest największa? (Tych sum jest $10^6 - 1000 + 1$).

25. Jak znaleźć współczynniki wielomianu $c_2x^2+c_1x+c_0 = (a_1x+a_0)(b_1x+b_0)$ używając tylko trzech mnożeń.

26. Jak wyliczyć współczynniki c_0, c_1, c_2 wielomianu $c_2x^2 + c_1x + c_0 = (a_1x + a_0)(b_1x + b_0)$ znając a_0, a_1, b_0, b_1 używając tylko trzech mnożeń.

27. Korzystając z twierdzenia o rekursji uniwersalnej rozwiąż następujące zależności:

    (a) $T(N) = 5T(n/3) + n$,
    (b) $T(N) = 4T(n/2) + n^2$,
    (c) $T(N) = 9T(n/3) + n^2$,
    (d) $T(N) = 6T(n/3) + n^2$,
    (e) $T(N) = 3T(n/3) + n$,
    (f) $T(N) = 5T(n/2) + n^2$,
    (g) $T(N) = T(n/2) + 1$.

28. Udowodnij, że $3n^3 + n^2 - 12n + 5 = \Theta(n^3)$.

29. Udowodnij, że $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$ oraz $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$

30. Wykaż, że do zgadnięcia liczby z przedziału od 1 do n, wystarczy $\lceil \log_2 n \rceil$ pytań binarnych. (Pytanie binarne to takie na które odpowiedzią jest „tak", lub „nie").

31. Ile jest permutacji liczb $1, \ldots n$. Ile trzeba zadać pytań binarnych, by odgadnąć, którą permutację ktoś ma na myśli.

32. Jaka będzie złożoność algorytmu jeżeli:

    (a) $T(1) = 1$, $T(n) = 4T(\lceil \frac{n}{2} \rceil)$,
    (b) $T(1) = 1$, $T(n) = 8T(\lceil \frac{n}{2} \rceil)$.

33. Ile potrzeba mnożeń by obliczyć wartość wielomianu stopnia n?

34. Ile potrzeba mnożeń by znaleźć współczynniki wielomianu (ax+b)(cx+d)?

35. Jak zrealizować mnożenie wielomianów stopnia $2n$ przy pomocy mnożenia wielomianów stopnia n, dodawania wielomianów i mnożenia przez jednomian ? Napisz zależność rekurencyjną jaką będzie spełniała złożoność twojego algorytmu, i oszacuj ją.

36. Ile porównań trzeba by znaleźć: a) drugi b) trzeci co do wielkości element tablicy.

37. Wielomian stopnia $2n$ można pomnożyć używając 4 mnożeń wielomianów stopnia n. Jaka będzie asymptotyczna złożoność takiego algorytmu?

38. Znajdź postać funkcji X jeśli:

    (a) $X(1) = 0$, $X(n) = 2X(\lceil \frac{n}{2} \rceil) + (n-1)$;

    (b) $X(1) = 0$, $X(n) = X(n-1) + (n-1)$.

39. Napisz program wypisujący wszystkie możliwe permutacje elementów tablicy `int t[n];`.

40. Ilość partycji liczby n to ilość jej różnych rozkładów na składniki naturalne (kolejność nie gra roli). Znaleźć algorytm lub wzór rekurencyjny na $P(n)$.

41. Wielomian stopnia $2n$ można pomnożyć używając 4 mnożeń wielomianów stopnia n. Jaka będzie asymptotyczna złożoność takiego algorytmu? Jak zmieni się ona gdy zamiast 4 użyjemy 3 mnożeń, wykorzystując pomysł z poprzedniego zadania.

Sortowanie

42. Jak jest minimalna ilość porównań gwarantująca posortowanie n liczb dla $n = 2$, $n = 3$, $n = 4$ oraz $n = 5$. (Nadobowiązkowe $n = 6$, $n = 7$).

43. Jak wiąże się faktyczna ilość porównań wykonywanych przez algorytm sortowania przez proste wstawianie z ilością inwersji w danych wejściowych?

44. Znajdź związek między ilością porównań faktycznie wykonywanych przy sortowaniu przez wstawianie tablicy $a_0, \ldots a_{n-1}$ a ilością inwersji w tej tablicy. (Inwersją nazywamy parę indeksów (i, j) taką, że $i < j \wedge a_i > a_j$).

45. Napisz funkcję `merge(double tab[],int i, int j, int n);` potrzebną w algorytmie `merge_sort` (sortowanie przez scalanie). Funkcja `merge` zakładając, że ciągi `tab[0]` do `tab[i-1]` oraz `tab[i]` do `tab[n-1]` są posortowane scala je w jeden posortowany ciąg od `tab[0]` do `tab[n-1]`. Zadbaj by złożoność twego algorytmu nie była większa niż $O(n-i)$.

46. Napisz funkcję
    `double czas_dzialania(void sort(double* ,int), int n)`, która posłuży do badania szybkości różnych funkcji sortujących. Jej działanie

polega na wygenerowaniu przypadkowych danych o rozmiarze `n`, uruchomiemiu „stopera”, uruchomieniu funkcji sortującej `sort`, i zatrzymaniu „stopera”. Funkcja powinna zwracać zmierzony czas działania funkcji `sort` wyrażony w (mili-) sekundach.

47. Jaka byłaby złożoność algorytmu `merge_sort` gdyby dzielić tablice nie na równe części lecz na części o wielkości 25% i 75%?

48. Napisz algorytm `merge_sort`, który działałby na liście 1-linkowanej.

49. Jaka byłaby złożoność algorytmu `merge_sort` gdyby dzielić tablice nie na równe części, lecz na części o wielkości 25% i 75%, lub ogólniej, na części o rozmiarach nq oraz $n(1-q)$.

50. Napisz algorytm `merge_sort`, który działałby na liście 1-linkowanej.

51. Napisz funkcję `void rotate(T *p, T *s, T *k)` zamieniającą miejscami obszary `[p,s)` i `[s,k)` o długościach m i n.

    (a) Pokaż, że można zrobić to przy użyciu trzech inwersji i wówczas algorytm sprowadza się do wykonania $m+n$ zamian czyli $3(m+n)$ podstawień.

    (b) Dla przypadku, gdy $m = n$ napisz procedurę wykonującą tylko m zamian czyli $3m$ podstawień.

    (c) Dla przypadku, gdy $m = n+1$ napisz procedurę wykonującą tylko $2m$ podstawień.

52. Napisz funkcję `void in_place_merge(T *p, T *s, T *k)`, która nie używa dodatkowej pamięci. Wsk. znajdź takie $p1 \in [p, s)$ i $k1 \in [s, k)$ aby po wykonaniu `rotate(p1,s,k1)` element $*s$ stanowił medianę i skorzystaj z rekurencji.

53. Udowodnij, że złożoność `in_place_merge` wynosi $O(n \log n)$, a złożoność `in_place_merge_sort`, który jej używa, wynosi $O(n \log^2 n)$.

54. Pokaż, że `selection_sort` nie jest stabilny. Ile dodatkowej pamięci potrzeba, by uczynić go stabilnym ?

55. Dana jest macierz $n \times m$, której wiersze są ciągami rosnącymi. Udowodnij, że własność ta nie zmieni się, jeśli posortujemy wszystkie kolumny tej macierzy.

56. Ciąg $(t_1, t_2, \ldots, t_N)$ nazywamy n-posortowanym jeśli $t_p \leq t_{p+n}$ gdy $1 \leq p \leq N - n$. n-sortowanie to sortowanie wszystkich podciągów postaci $(t_i, t_{i+n}, t_{i+2n}, \ldots)$ dla $i \leq n$.

    Udowodnij, że ciąg n-posortowany nie traci tej własności w trakcie m-sortowania.

57. Posortuj metodą sortowania pozycyjnego liczby: 1, 34, 123, 321, 432, 132, 543, 651, 91, 32, 987, 910, 643, 641, 12, 342, 498, 987, 965, 122, 121, 431, 350.

58. Zaprogramuj i przetestuj podaną na wykładzie implementację sortowania pozycyjnego.

59. Skonstruuj sieć sortującą poprawnie 6, 8, 10 liczb. Skonstruuj optymalną taką sieć.

60. Napisz program porównujący złożoność czasową algorytmów sortujących: przez kopcowanie, bąbelkowe, przez proste wstawianie, przez łączenie. Program powinien drukować tabelkę postaci:

n	Kopiec	Bąbel	Szybkie	Wstawianie	Łączenie
1					
2					
4					
8					
16					
$\vdots$					

61. Które z procedur sortujących:

    (a) quicksort,

    (b) heapsort,

    (c) mergesort,

    (d) insertionsort

    są stabilne. W każdym przypadku udowodnij stabilność lub znajdź konkretny przykład danych, dla których algorytm nie zachowa się stabilnie.

Drzewa

62. Udowodnij, że w każdym drzewie binarnym ponad połowa wskaźników ma wartość NULL.

63. Ile jest różnych drzew binarnych zawierających liczby $1, \ldots n$.

64. Jaką dodatkową informację należy przechowywać w każdym węźle drzewa binarnego, by łatwo znajdować medianę zawartych w nim elementów?

65. Napisz implementację usuwania węzła z drzewa binarnego wg następującego schematu:

    (a) jeśli usuwany węzeł nie ma dzieci, to go usuwamy a odpowiedni wskaźnik zmieniamy na NULL.

    (b) jeśli ma jedno dziecko, to go usuwamy, a odpowiedni wskaźnik w węźle rodzica zastępujemy wskaźnikiem na to dziecko.

    (c) jeśli ma dwoje dzieci, to nie usuwamy tego węzła, lecz najmniejszy element w jego prawym poddrzewie, a dane i klucz tego elementu wpisujemy do węzła, który miał być usunięty.

66. Napisz procedurę `void BSTdelete(BSTnode *t)`, która usuwa z pamięci komputera drzewo BST.

67. Zmodyfikujmy strukturę `BSTnoode` dodając dodatkowe pole `int left_size;`, które zapamięta ile aktualnie jest węzłów w lewym poddrzewie tego wierzchołka.

    (a) Napisz funkcję `BStnode * ity(BSTnode *t, int i)`, której wynikiem jest adres `i`-tego (w kolejności IN ORDER) węzła drzewa.

    (b) Jakie zmiany w funkcji `void insert(BSTnode *& t, int k)` są konieczne, by wartość w polach `left_size` była uaktualniana przy wstawianiu do drzwa nowych kluczy?

68. Napisz procedurę `int poziom(BSTnode * t, int klucz)`, której wynikiem jest poziom w drzewie `t`, na którym występuje `klucz`.

69. Napisz procedurę `void lewa_rotacja(BSTnode * & t)`.

70. Udowodnij, że ponad połowa wskaźników w drzwie binarnym ma wartość NULL.

71. Czy ponad połowa węzłów drzewa czerwono-czarnego musi być czarna? Dlaczego?

72. Ile jest różnych kształtów drzew BST o n węzłach?

73. Napisz program (lub wzór rekurencyjny) obliczający ilość różnych kształtów drzew czerwono czarnych o ustalnoej czarnej wysokości.

74. Niech $K(n)$ oznacza ilość różnych kształtów drzew binarnych o n węzłach. Znajdź wzór rekurencyjny wyrażający $K(n)$ przez $K(i)$ dla $i < n$.

75. Napisz klasę `priority_queue` opartą na kopcu zrealizowanym w tablicy. (Dziećmi elementu `t[i]` są `t[2*i+1]` oraz `t[2*i+2]`. Ojcem `t[i]` jest `t[(i-1)/2]`.) Zaimplemetuj metody `put(T)`, `T max()`, `T get_max()` oraz `bool is_empty()`.

76. Oznaczmy przez $f(n)$ ilość kształtów drzew BST o n węzłach. Napisz wzór wyrażający $f(n)$ przez $f(0), \ldots, f(n-1)$.

77. Napisz funkcję rekurencyjną `int F(BSTnode * t)`obliczającą:

    (a) ilość węzłów w drzewie `t`,

    (b) głębokość w drzewa `t`,

78. Napisz implementację usuwania węzła z drzewa binarnego wg następującego schematu:

    (a) jeśli usuwany węzeł nie ma dzieci, to go usuwamy a odpowiedni wskaźnik zmieniamy na NULL.

    (b) jeśli ma jedno dziecko, to go usuwamy, a odpowiedni wskaźnik w węźle rodzica zastępujemy wskaźnikiem na to dziecko.

    (c) jeśli ma dwoje dzieci, to nie usuwamy tego węzła, lecz najmniejszy element w jego prawym poddrzewie, a dane i klucz tego elementu wpisujemy do węzła, który miał być usunięty.

79. Jaką dodatkową informację należy przechowywać w każdym węźle drzewa binarnego, by łatwo znajdować medianę zawartych w nim elementów? Napisz implementację funkcji `BTSnode* ity(BSTnode *t, int i)`, korzystającą z tego dodatkowego pola, która będzie działała w czasie $O(\log n)$ dla drzew zrównoważonych.

80. Do pustego drzewa czerwono-czarnego wstaw kolejno 20 przypadkowych liczb. Następnie usuń je w tej samej kolejności.

81. Do pustego B-drzewa o parametrze $t = 2$ wstaw 20 przypadkowych liczb, a następnie usuń je w kolejności wstawiania.

82. Narysuj przykładowy kopiec dwumianowy o 17 kluczach. Wykonaj 4 razy operację `Delete_min`.

83. Jaka jest minimalna, a jaka maksymalna ilość kluczy w B-drzewie o ustalonym t (jednakowym na wszystkich poziomach) i głębokości k?

Rekurencja i techniki programowania

84. Napisz program, który znajdzie sposób, jak przejść konikiem po szachownicy odwiedzając każde pole dokładnie raz.

85. Użyj rekursji do rozwiązania problemu wież z Hanoi.

86. Napisz program rozwiązujący poroblem wież z Hanoi: Jaka jest liczba ruchów potrzebna do przełożenia n krążków różnej wielkości z patyczka A na patyczek C, jeśli można używać też patyczka B, a krążek większy nie może nigdy leżeć na mniejszym. Program powinien wypisywać wszystkie konieczne ruchy w postaci:
    1. A->C
    2. A->B
    3. C->B
    ...

87. Napisz program znajdujący wszystkie ustawienia 8 hetmanów na szachownicy takie, że żaden z nich nie szachuje innego.

88. Niech dana będzie tablica A zawierająca pary liczb: $(1, 0)$, $(2, 100)$, $(3, 100)$ ... $(n, 100)$. Pierwszy element każdej pary nazywamy nazwą, a drugi wartością. Tablicę tą traktujemy jako kopiec ze względu na wartość. Jaką dodatkową stukturę danych trzeba użyć, by w efektywny sposób wykonywać operację `Decrease_value(k,x)`, przypisującą nazwie `k` nową wartość `x` mniejszą od poprzedniej? Jak należy zmodyfikować operacje `Delete_min` oraz `Insert`, by ta nowa struktura nie dezaktualizowała się w trakcie ich działania?

89. (Algorytm z nawrotami) Napisz program, który (np. metodą prób i błędów) znajduje drogę konika szachowego po szachownicy, taką że każde pole jest odwiedzone dokładnie raz.

90. Jak skonstruować program liczący ułamek łańcuchowy

$$g = \cfrac{a}{b + \cfrac{a}{b + \cfrac{a}{b + \cfrac{a}{\ldots}}}}$$

z zadaną z góry dokładnością.

91. Udowodnij, że Z_7 jest ciałem tzn. w pierścieniu $\{0, 1, \ldots, 6\}$ z mnożeniem i dodawaniem modulo 7 jest jednoznacznie zdefiniowana operacja odwrotna do mnożenia. Uzasadnij, że jest tak dla każdego Z_p gdy p jest liczbą pierwszą.

92. Zakładając, że masz do dyspozycji algorytm na mnożenie dowolnie długich liczb całkowitych oraz znasz wartość pewnej dużej liczby pierwszej (np. 100-cyfrowej) zaproponuj sposób doskonałedo szyfrowania tekstu o długości do 100 znaków i niedoskonałego szyfrowania tekstów dłuższych.

Do zadań 1. 2. 3. podchodzimy kolejno - do każdej operacji następna osoba.