Zadanie 4. (0–2 pkt) ..... / 2

Przeanalizuj program. Uzupełnij tabelę – podaj wartości wypisane przez program.

```
import heapq
h = [8, 2, 6]
heapq.heapify(h)
heapq.heappush(h, 4)
a = heapq.heappop(h)
heapq.heappush(h, 1)
b = heapq.heappop(h)
c = heapq.heappop(h)
print(a, b, c, len(h))
```

a	b	c	len(h)

Zadanie 5. (0–2 pkt) ..... / 2

Program biblioteki szkolnej obsługuje 50 000 czytelników. Dla każdego wprowadzonego numeru karty sprawdza, czy taki numer istnieje. Uczeń trzyma numery w liście i sprawdza warunek numer in karty. Wskaż lepszą strukturę danych z biblioteki standardowej Pythona i uzasadnij wybór, porównując czas jednego sprawdzenia w obu strukturach.

Miejsce na odpowiedź

PIERWSZA ŁAWKA

Zadanie 6. (0–4 pkt) ..... / 4

Zapisz w pseudokodzie lub w wybranym języku programowania algorytm, który w ciągu A uporządkowanym niemalejąco znajdzie największy indeks i, dla którego A[i] = x, albo zwróci −1, gdy x w ciągu nie ma. Algorytm ma wykonać liczbę porównań rzędu log n.

Przykład: dla A = 2, 4, 4, 4, 7, 9 i x = 4 wynikiem jest 3, a dla x = 5 wynikiem jest −1.

Specyfikacja

Dane: n – liczba całkowita dodatnia; A[0..n−1] – ciąg liczb całkowitych uporządkowany niemalejąco; x – liczba całkowita

Wynik: i – największy indeks, dla którego A[i] = x, albo −1

Miejsce na odpowiedź

UWAGA Rozwiązanie przeglądające ciąg element po elemencie (liczba porównań rzędu n) nie spełnia warunku zadania.

PIERWSZA ŁAWKA

Jak korzystać z zestawu

- Karta 1 przypomina najważniejsze wiadomości i pokazuje przykłady rozwiązane krok po kroku (z kodem w Pythonie albo opisem kroków w aplikacji) – od niej warto zacząć.
- Karta 2 to ćwiczenia podstawowe, karta 3 – utrwalenie w formule zadań maturalnych CKE (ramka „Jak to sprawdza CKE").
- Karta 4 to praca przy komputerze: pliki danych są w archiwum ZIP dołączonym do zestawu, a odpowiedzi kontrolne dla plików *_przyklad (pliki o nazwie z dopiskiem „_przyklad") podano przy zadaniach. Wyniki zapisuje się w plikach wynikiN.txt – jak na maturze.
- Karty 5–6 to kartkówka (15 minut), karty 7–8 – sprawdzian (45 minut), obie w grupach A i B: te same typy zadań i ta sama punktacja, inne dane. Kartkówka i sprawdzian nie wymagają komputera.
- Klucz zawiera odpowiedzi, przykładowe programy, zasady oceniania w stylu CKE i numery wymagań podstawy. Osobny plik „Powtórka" zbiera teorię, typowe błędy i test „Sprawdź się".

Oznaczenia

ZP	zakres podstawowy
ZR	zakres rozszerzony (treści tylko dla ZR)
(0–2 pkt)	liczba punktów możliwych do zdobycia za zadanie
Zadanie 4.1.	wiązka zadań do wspólnego materiału źródłowego
Jak to sprawdza CKE	typ zadania i typowy błąd ze sprawozdań CKE
Na maturze	wymagania z podstawy programowej, których dotyczy temat

Wymagania z podstawy programowej (Dz.U. 2024 poz. 1019)

- **I.R1** wyróżnia w problemie podproblemy i charakteryzuje: metodę połowienia, stosuje podejście zachłanne i rekurencję;
- **I.R3** w zależności od problemu rozwiązuje go, stosując metodę wstępującą lub zstępującą;
- **I+II.R3b** objaśnia oraz porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przykłady problemów i algorytmów, w szczególności: b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),
- **I+II.R3c** objaśnia oraz porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przykłady problemów i algorytmów, w szczególności: c) metodę dziel i zwyciężaj (jednoczesne znajdowanie minimum i maksimum, sortowanie przez scalanie i szybkie),
- **I+II.R1j** zapisuje za pomocą listy kroków lub pseudokodu i implementuje w wybranym języku programowania algorytmy poznane na wcześniejszych etapach oraz algorytmy: j) rekurencyjnego tworzenia fraktali;

Materiał oryginalny – żadne zadanie nie jest kopią zadania CKE ani podręcznika; dane w plikach są fikcyjne. Podstawa programowa LO i technikum w brzmieniu z 2024 r. (Dz.U. 2024 poz. 1019). Wymagania: I.R1, I.R3, I+II.R3b, I+II.R3c, I+II.R1j. Miejsce w programach nauczania: „Informatyka na czasie" (Nowa Era) ZR – algorytmika, klasa 3 (nazwy serii podano tylko informacyjnie). Do użytku własnego i pracy z własnymi uczniami.

PIERWSZA ŁAWKA

Metody numeryczne: połowienie (miejsca zerowe), pierwiastek kwadratowy – w praktyce, przy komputerze

Pliki danych w archiwum ZIP · wyniki w plikach wynikiN.txt

ZR </> KARTA 4/8

Imię i nazwisko: ..... Klasa: ..... Data: ..... Wynik: ..... / 13 pkt

Ćwiczenia · 13 pkt

Zadanie 1. (0–7 pkt)

Równania trzeciego stopnia

Opis danych

Plik **rownania.txt** zawiera 200 wierszy. W każdym są dwie liczby całkowite p i q ($1 \le p \le 20$, $-500 \le q \le 500$) oddzielone spacją – współczynniki równania $x^3 + p \cdot x + q = 0$. Dla $p > 0$ funkcja $f(x) = x^3 + px + q$ jest rosnąca, więc każde równanie ma dokładnie jedno rozwiązanie i leży ono w przedziale $[-10, 10]$ ($f(-10) < 0 < f(10)$). Pierwsze wiersze pliku:

```
18 -342
18 -156
2 -275
```

Plik **rownania_przyklad.txt** zawiera 10 wierszy. Rozwiązania wyznacz metodą połowienia; nie korzystaj z bibliotek do rozwiązywania równań.

Do dyspozycji masz plik **rownania.txt** oraz mniejszy plik **rownania_przyklad.txt**. Odpowiedzi dla danych przykładowych podano przy zadaniach – najpierw sprawdź na nich swój program.

Napisz program (lub kilka programów), który(-e) znajdzie(-ą) i da(dzą) odpowiedzi do podanych zadań. Odpowiedzi zapisz w pliku **wyniki1.txt**, a każdą z nich poprzedź numerem odpowiedniego zadania.

Do oceny oddajesz: plik tekstowy **wyniki1.txt** z odpowiedziami do zadań 1.1–1.3 (każda poprzedzona numerem zadania) oraz pliki z kodem źródłowym programów, np. **zadanie1.py**.

Dane fikcyjne, przygotowane do zadania.

Zadanie 1.1. (0–2 pkt) ..... / 2

Podaj, ile równań ma rozwiązanie w przedziale (1, 2). *Dla danych z pliku przykładowego odpowiedź: 1.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 1.2. (0–2 pkt) ..... / 2

Podaj współczynniki p i q równania o największym rozwiązaniu oraz to rozwiązanie zaokrąglone do 3 miejsc po przecinku (trzy wartości oddzielone spacjami). *Dla danych z pliku przykładowego odpowiedź: 11 -356 6.571.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 1.3. (0–3 pkt) ..... / 3

Podaj sumę rozwiązań wszystkich równań z pliku zaokrągloną do 2 miejsc po przecinku. *Dla danych z pliku przykładowego odpowiedź: 6.42.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 5. (0–2 pkt) ..... / 2

Przeanalizuj program. Uzupełnij.

```
import heapq
h = []
for x in [2, 9, 6]:
    heapq.heappush(h, (x % 3, x))
print(heapq.heappop(h))
```

1. Program wypisze _____
2. Kopiec h po zakończeniu zawiera _____ elementy.

Zadanie 6. (0–3 pkt) ..... / 3

Aplikacja do wypożyczania laptopów musi wiele razy na sekundę sprawdzać, czy numer seryjny laptopa jest na liście zablokowanych (20 000 numerów). Którą strukturę danych wybierzesz – listę czy zbiór (set)? Uzasadnij, podając czas sprawdzenia w obu przypadkach.

Miejsce na odpowiedź

Zadanie 7. (0–4 pkt) ..... / 4

Na gałęzi main wykonano git merge sklepik i Git zgłosił konflikt w pliku ustawienia.py (fragment poniżej). Zespół ustalił, że obowiązuje wartość z gałęzi sklepik. Wyjaśnij, skąd pochodzi każdy z dwóch wariantów, zapisz, jak ma wyglądać ten fragment pliku po rozwiązaniu konfliktu, i podaj dwa polecenia, które kończą scalanie.

```
<<<<<<< HEAD
CENA = 4.50
=======
CENA = 5.00
>>>>>>> sklepik
```

Miejsce na odpowiedź

PIERWSZA ŁAWKA

Zadanie 6. (0–4 pkt) ..... / 4

Dla A = 20, 5, 16, 2, 25, 8, 13 wykonaj funkcję podziel(A, 0, 6) z sortowania szybkiego (element osiowy – ostatni). Podaj ciąg po podziale i zwrócony indeks. Następnie wyjaśnij, dlaczego dla ciągu już uporządkowanego rosnąco to sortowanie wykonuje liczbę porównań rzędu n^2.

```
def podziel(A, l, p):          # element osiowy: A[p]
    os = A[p]
    i = l
    for j in range(l, p):
        if A[j] < os:
            A[i], A[j] = A[j], A[i]
            i = i + 1
    A[i], A[p] = A[p], A[i]
    return i
```

Miejsce na odpowiedź

Zadanie 7. (0–3 pkt) ..... / 3

Uporządkowany słownik ma 500 000 haseł. Wyjaśnij, korzystając z pojęcia logarytmu, ile najwyżej porównań z hasłem środkowym wykona wyszukiwanie binarne, i porównaj to z wyszukiwaniem liniowym.

Miejsce na odpowiedź

Rekurencja i dziel i zwyciężaj, metoda wstępująca i zstępująca, fraktale – w praktyce, przy komputerze

ZR </> KARTA 4/8

Pliki danych w archiwum ZIP · wyniki w plikach wynikiN.txt

Imię i nazwisko: ..... Klasa: ..... Data: ..... Wynik: ..... / 12 pkt

Ćwiczenia · 12 pkt

Zadanie 1. (0–6 pkt)

Kody kontrolne

Opis danych

Kody kontrolne paczek w magazynie liczy funkcja F z karty 3: F(n) = n dla n < 10, a dla n ≥ 10 F(n) = F(n div 10 + n mod 10). Plik **kody.txt** zawiera 400 wierszy – w każdym jedną liczbę całkowitą n ($1 \le n \le 10^9$). Pierwsze wiersze pliku:

```
139512
571879
414782
248261
```

Plik **kody_przyklad.txt** zawiera 15 liczb w tym samym formacie. Zaprogramuj F **rekurencyjnie**.

Do dyspozycji masz plik **kody.txt** oraz mniejszy plik **kody_przyklad.txt**. Odpowiedzi dla danych przykładowych podano przy zadaniach – najpierw sprawdź na nich swój program.

Napisz program (lub kilka programów), który(-e) znajdzie(-ą) i da(dzą) odpowiedzi do podanych zadań. Odpowiedzi zapisz w pliku **wyniki1.txt**, a każdą z nich poprzedź numerem odpowiedniego zadania.

Do oceny oddajesz: plik tekstowy **wyniki1.txt** z odpowiedziami do zadań 1.1–1.3 (każda poprzedzona numerem zadania) oraz pliki z kodem źródłowym programów, np. **zadanie1.py**.

Dane fikcyjne, przygotowane do zadania.

Zadanie 1.1. (0–2 pkt) ..... / 2

Podaj, dla ilu liczb z pliku F(n) = 7. *Dla danych z pliku przykładowego odpowiedź: 2.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 1.2. (0–2 pkt) ..... / 2

Podaj pierwszą w pliku liczbę, dla której obliczenie F(n) wymaga największej liczby wywołań funkcji F (łącznie z pierwszym), oraz tę liczbę wywołań. Obie wartości oddziel spacją. *Dla danych z pliku przykładowego odpowiedź: 877757511 10.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 1.3. (0–2 pkt) ..... / 2

Podaj, dla ilu liczb n z pliku zachodzi F(n + 1) = F(n) + 1. *Dla danych z pliku przykładowego odpowiedź: 14.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 7. (0–3 pkt) ..... / 3

Wykonaj funkcję isqrt(n) (połowienie na liczbach całkowitych, lo = 0, hi = n + 1, dopóki hi − lo > 1) dla n = 500. Podaj kolejne wartości s, wynik i liczbę obrotów pętli.

```
def isqrt(n):                  # największe x, dla którego x * x <= n
    lo, hi = 0, n + 1            # lo * lo <= n < hi * hi
    while hi - lo > 1:
        s = (lo + hi) // 2
        if s * s <= n:
            lo = s
        else:
            hi = s
    return lo
```

Miejsce na odpowiedź

Zadanie 8. (0–2 pkt) ..... / 2

Porównaj funkcję isqrt(n) (połowienie na liczbach całkowitych) i metodę Herona na liczbach rzeczywistych przy sprawdzaniu, czy n jest kwadratem liczby całkowitej. Która metoda daje wynik dokładny i dlaczego?

Miejsce na odpowiedź

PIERWSZA ŁAWKA

PIERWSZA ŁAWKA · POWTÓRKA · INFORMATYKA · KLASA 3 LO I TECHNIKUM · ZAKRES ROZSZERZONY · PROGRAMOWANIE ZR: STRUKTURY DANYCH I BIBLIOTEKI, IDE, PROJEKT ZESPOŁOWY (REPOZYTORIUM)

Struktury danych, IDE i repozytorium – powtórka

Dobór struktur danych (list, deque, set, dict, Counter, heapq), środowisko programistyczne i debuger, praca zespołowa w repozytorium Git. Klasa 3 LO i technikum, zakres rozszerzony.

1. Jaką strukturę wybrać

Potrzebuję…	Struktura	Czas operacji
dostępu po indeksie, dopisywania na końcu	list	O(1)
kolejki FIFO, obu końców	collections.deque	O(1) na obu końcach
sprawdzania „czy jest”	set / dict	średnio O(1)
zliczania	Counter / dict	średnio O(1) na element
zawsze najmniejszego	heapq	push i pop – rzędu log n

- Lista z pop(0) albo insert(0, x) przesuwa wszystkie elementy – O(n). Do kolejki bierz deque.

2. Biblioteki w praktyce

```
from collections import Counter

gatunki = "sikora wróbel sikora kos wróbel sikora dzięcioł".split()
licz = Counter(gatunki)
print(licz["sikora"], licz["szpak"])
print(licz.most_common(2))
```

- sorted(dane, key=…, reverse=True) – nowa lista; dane.sort() – w miejscu, zwraca None. Sortowanie jest stabilne.
- heapq.heappush(h, (priorytet, nazwa)) – pary porównują się najpierw po priorytecie.

3. IDE i testy

- Punkt przerwania zatrzymuje program. step (s) – wchodzi do funkcji, next (n) – wykonuje wiersz w całości, continue (c) – do następnego punktu przerwania, p x – pokaż zmienną.
- assert warunek – gdy warunek fałszywy, program kończy się błędem AssertionError. Testuj przykład z treści i przypadki brzegowe.

PIERWSZA ŁAWKA · POWTÓRKA · INFORMATYKA · KLASA 3 LO I TECHNIKUM · ZAKRES ROZSZERZONY · WYSZUKIWANIE I SORTOWANIE: BINARNE, MIN-MAX JEDNOCZEŚNIE, PRZEZ SCALANIE, SZYBKIE; EFEKTYWNOŚĆ I LOGARYTM

Wyszukiwanie i sortowanie, efektywność – powtórka

Wyszukiwanie liniowe i binarne, logarytm, jednoczesne min i max, sortowanie przez scalanie i szybkie, porównywanie efektywności. Klasa 3 LO i technikum, zakres rozszerzony.

1. Wyszukiwanie binarne i logarytm

- Tylko w ciągu uporządkowanym. Porównaj x ze środkiem, odrzuć połowę: lo = s + 1 albo hi = s − 1.

```
def szukaj(A, x):
    lo, hi = 0, len(A) - 1
    while lo <= hi:
        s = (lo + hi) // 2
        if A[s] == x:
            return s
        if A[s] < x:
            lo = s + 1
        else:
            hi = s - 1
    return -1
```

- Pesymistycznie $\lfloor \log_2 n \rfloor + 1$ obrotów: n = 1000 → 10, n = 1 000 000 → 20. Podwojenie n = +1 obrót.
- bisect_left / bisect_right – granice wystąpień x; liczba wystąpień = bisect_right − bisect_left.

2. Min i max jednocześnie

- Osobno: 2(n − 1) porównań. Parami: 3 porównania na 2 elementy, dla parzystego n – 3n/2 − 2.
- Dziel i zwyciężaj: min i max połówek + 2 porównania; T(2) = 1, T(4) = 4, T(8) = 10.

3. Scalanie i sortowanie szybkie

- Scalanie dwóch uporządkowanych ciągów długości a i b – co najwyżej a + b − 1 porównań.
- Przez scalanie: O(n log n) zawsze, stabilne, potrzebuje dodatkowej pamięci.
- Szybkie: podział względem elementu osiowego, bez scalania; średnio O(n log n), pesymistycznie $O(n^2)$ (ostatni element jako osiowy w ciągu posortowanym).

CO JESZCZE JEST W ŚRODKU

Informatyka, klasa 3 LO i technikum (zakres rozszerzony) – pakiet 4 zestawów: kartkówki, sprawdziany, karty pracy i powtórki

Podgląd pokazał 10 z 136 stron. Pełna wersja: 136 stron A4, PDF do druku i DOCX do edycji.

- Programowanie ZR: struktury danych i biblioteki, IDE, projekt zespołowy (repozytorium) – kartkówka, sprawdzian i karty pracy z powtórką,…
- Wyszukiwanie i sortowanie: binarne, min-max jednocześnie, przez scalanie, szybkie; efektywność i logarytm – kartkówka, sprawdzian i karty pracy z…
- Rekurencja i dziel i zwyciężaj, metoda wstępująca i zstępująca, fraktale – kartkówka, sprawdzian i karty pracy z powtórką, informatyka, klasa 3 LO i…
- Metody numeryczne: połowienie (miejsca zerowe), pierwiastek kwadratowy – kartkówka, sprawdzian i karty pracy z powtórką, informatyka, klasa 3 LO i…

Kup pełną wersję – plik od razu na e-mail
pierwszalawka.pl/material/informatyka-liceum-klasa-3-zakres-rozszerzony-pakiet