```python
# Algorytm

# Lista wygenerowania do testowania
# - 20 losowych elementów
import random

my_list = []
for i in range(20):
    value = random.randint(1, 100)
    my_list.append(value)

# ------------------------------------------------------------------------
# Słownik - krótko
osoba = {
    "imie": "Jan",
    "nazwisko": "Kowalski",
    "wiek": 30,
    "adres": {
        "ulica": "Kwiatowa",
        "numer": 10,
        "miasto": "Kraków"
    }
}

# słownik: osoba, 4 klucze:
#   - imie (str)
#   - nazwisko (str)
#   - wiek (int)
#   - adres [słownik zagnieżdżony (3 klucze):
#               - ulica (str)
#               - numer (int)
#               - miasto (str)
#               ]

# ---------------- OPERACJE NA SŁOWNIKACH ----------------
# - dodanie nowej pary klucz - wartość:
osoba["email"] = "jan.kowalski@example.com"

# - usunięcie pary klucz-wartość:
del osoba["wiek"]

# - dostęp do wartości na podstawie klucza:
print(osoba["imie"])

# - iteracja przez pary klucz-wartość w słowniku:
for klucz, wartosc in osoba.items():
    print(klucz, wartosc)
```

```python
# ------------------------------------------------------------------------

# 1. SORTOWANIE BĄBELKOWE (bubble sort)
# wizualizacja:
# https://commons.wikimedia.org/wiki/File:Bubble-sort.gif
# https://www.sortvisualizer.com/bubblesort/

def bubble_sort(lista):
    n = len(lista)
    # iterujemy przez wszystkie elementy listy
    for i in range(n):
        # ostatnie i elementów są już posortowane
        for j in range(0, n-i-1):
            # porównujemy sąsiednie elementy
            if lista[j] > lista[j+1]:
            # zamieniamy miejscami, jeśli kolejność jest nieprawidłowa
                lista[j], lista[j+1] = lista[j+1], lista[j]
    return lista

print(my_list)
bubble_sort(my_list)
print("Posortowana lista: ")
print(my_list)

# ------------------------------------------------------------------------

# 2. WYSZUKIWANIA

# 2.1 WYSZUKIWANIE LINIOWE
# - jak sprawdzić czy dana wartość znajduje się w naszej liście? (robią dzieci)

def linear_search(lista, x):
    n = len(lista)
    # przechodzimy po wszytskich elementach
    for i in range(n):
        # sprawdzamy czy element jest naszym elementem szukanym
        if lista[i] == x:
            return i
    return -1
```

```python
# 2.2 WYSZUKIWANIE BINARNE
# - działa na posortowanym zbiorze
# - metoda dziel i zwyciężaj
# https://www.mathwarehouse.com/programming/gifs/binary-vs-linear-search.php

# (robimy wspólnie)
def binary_search(arr, x):
    low = 0 # początek podzbioru
    high = len(arr) - 1 # koniec podzbioru
    mid = 0
    while low <= high:
        # wyliczamy środek
        mid = (high + low) // 2

        if arr[mid] < x:
            low = mid + 1

        elif arr[mid] > x:
            high = mid - 1
        else:
            return mid
    return -1
```

```python
# ----------------------------------------------------------------------
```

```python
# ZADANIA DODATKOWE
# ZADANIE 1
# Zadaniem ucznia jest stworzenie programu, który będzie działał jak książka
# telefoniczna. Program powinien mieć następujące funkcjonalności:

#   ● Dodawanie nowego kontaktu - program powinien pytać użytkownika o imię i
#   nazwisko oraz numer telefonu i dodać te dane do listy kontaktów. Lista
#   kontaktów powinna być przechowywana w postaci listy słowników, gdzie
#   każdy słownik reprezentuje jeden kontakt.

#   ● Sortowanie kontaktów za pomocą metody sortowania bąbelkowego -
#   program powinien sortować listę kontaktów alfabetycznie według nazwisk z
#   wykorzystaniem funkcji bubble_sort.

#   ● Wyświetlanie listy kontaktów - program powinien wyświetlić listę kontaktów
#   w formacie: "imię nazwisko - numer telefonu". Kontakty powinny być
#   posortowane alfabetycznie według nazwisk.

print("\n")
# lista kontaktów
kontakty = []
# pętla programu
while True:
    print("1. Dodaj nowy kontakt")
    print("2. Wyświetl listę kontaktów")
    print("3. Wyjdź z programu")
    wybor = input("Wybierz opcję: ")

    # dodawanie nowego kontaktu
    if wybor == "1":
        imie = input("Podaj imię: ")
        nazwisko = input("Podaj nazwisko: ")
        numer = input("Podaj numer telefonu: ")
        kontakt = {"imie": imie, "nazwisko": nazwisko, "numer": numer}
        kontakty.append(kontakt)
        kontakty = bubble_sort(kontakty)
        print("Kontakt został dodany.")

    # wyświetlanie listy kontaktów
    elif wybor == "2":
        if len(kontakty) == 0:
            print("Brak kontaktów.")
        else:
            print("Lista kontaktów:")
            for kontakt in kontakty:
                print(f"{kontakt['nazwisko']} {kontakt['imie']} - {kontakt['numer']}")

    # wyjście z programu
    elif wybor == "3":
        break

    # błąd - nieznana opcja
    else:
        print("Nieznana opcja.")
```

```python
# ------------------------------------------------------------------------

# ZADANIE 2
# Zadaniem ucznia jest napisanie programu, który losuje liczbę z zakresu od 1 do
# 100, a następnie komputer będzie zgadywał tę liczbę, a my będziemy mu udzielać
# podpowiedzi w postaci "za mało" lub "za dużo" w zależności od tego, czy
# zgadnięta liczba jest mniejsza czy większa od wylosowanej liczby.

# Komputer będzie korzystał z algorytmu binary search, a program zakończy się, gdy
# komputer zgadnie liczbę.

# * Rekurencja - proces wywoływania funkcji przez samą siebie.
#
# W tym konkretnym kodzie, funkcja binary_search można zastosować rekurencje.
# Funkcja sortowania binarnego może być rekurencyjna
# będzie wywoływać samą siebie w dwóch warunkach:
# - kiedy odpowiedź jest "za mało",
# - kiedy odpowiedź jest "za dużo".
# Proces rekurencyjny trwa tak długo, aż odpowiedź jest "tak",
# wtedy funkcja zwraca wartość guess.

import random
def binary_search(low, high, guess):
    mid = (low + high) // 2
    print("Czy to jest liczba", guess, "?")
    response = input("Podaj odpowiedź (za mało/za dużo/tak): ")
    if response == "za mało":
        return binary_search(mid + 1, high, (mid + 1 + high) // 2)
    elif response == "za dużo":
        return binary_search(low, mid - 1, (low + mid - 1) // 2)
    else:
        return guess

# losujemy liczbę z zakresu 1-100
number = random.randint(1, 100)
print("Wylosowana liczba to", number)
# komputer zgaduje
guess = 50
result = binary_search(1, 100, guess)
print("Zgadłem! Wylosowana liczba to", result)

# ------------------------------------------------------------------------
# PODSUMOWANIE:
#   ● Wyszukiwanie binarne - podział uporządkowanej listy na połowy i
#   iteracyjnE przeszukiwaniE jednej z nich w poszukiwaniu szukanej wartości.

#   ● Sortowanie bąbelkowe - porównywaniE sąsiednich elementów listy
#   i zamianie ich kolejności, jeśli są w niewłaściwej kolejności.

# Pytania powtórzeniowe:
#   ● W jaki sposób działa wyszukiwanie binarne?
#   ● Jak działa sortowanie bąbelkowe?
```