Билеты к осеннему зачёту

Список основных вопросов

1. Наивный поиск подстроки в строке. Реализация на Python без использования стандартных методов str. (конспект 5)

2. Алгоритм обращения чисел в массиве. Реализация на Python. (лек 5, лек 7)

3. Алгоритм циклического сдвига в массиве. Реализация на Python. (лек 7)

4. Поиск корня уравнения методом бисекции. Требования алгоритма к функции. Реализация на Python и ассимптотика алгоритма. (лек 7)

5. Поиск значения в упорядоченном массиве методом бисекции. Алгоритм и его реализация на Python. (лек 7)

6. Изложить алгоритм, его особенности, ассимптотику и применимость для алгоритмов сортировки выбором, вставками и пузырьком. Без реализации на Python. (google)

7. Изложить алгоритм, его особенности, ассимптотику и применимость для алгоритмов пирамидальной сортировки, сортировки обезьяны и дурака. Без реализации на Python.

8. Изложить алгоритм, его особенности, ассимптотику и применимость для алгоритмов сортировки слиянием и подсчётом. Без реализации на Python.

9. Изложить алгоритм, его особенности, ассимптотику и применимость для алгоритмов сортировки Хоара и поразрядной сортировки. Без реализации на Python.

10. Сортировка обезьяны. Ассимптотика алгоритма. Реализация на Python. (лек 7)

```
from random import shuffle
def monkey_sort(A):
```

```
while A != sorted(A):
    shuffle(A)
```

11. Сортировка вставками. Ассимптотика алгоритма. Реализация на Python. (лек 8)

```
def insertion_sort(A):
for i in range(1, len(A)):
    new_elem = A[i]
    j = i — 1
    while j >= 0 and A[j] > new_elem:
        A[j + 1] = A[j]
        j -= 1
    A[j + 1] = new_elem
```

12. Сортировка выбором. Ассимптотика алгоритма. Реализация на Python. (лек 8)

```
for pos in range(N - 1):
    for i in range(pos + 1, N):
        if A[i] < A[pos]:
            A[i], A[pos] = A[pos], A[i]
```

13. Сортировка методом пузырька. Ассимптотика алгоритма. Реализация на Python. (лек 8)

```
for prohod in range(1, N):
    for i in range(N - prohod):
        if A[i] > A[i + 1]:
            A[i + 1], A[i] = A[i], A[i + 1]
```

14. Сортировка дурака. Ассимптотика алгоритма. Реализация на Python. (лек 8)

```
i = 0
while i < N - 1:
    if A[i] < A[i + 1]:
        i += 1
    else:
        A[i + 1], A[i] = A[i], A[i + 1]
        i = 0
```

15. Сортировка подсчётом. Применимость и ассимптотика алгоритма. Реализация на Python. (лек 8, *подсчёт - а) однопроходная (O(N+M), где M — мощность множества значений), б) неуниверсальная сортировка – нужно заранее знать диапазон значений. хорошо работает ,например, для цифр*)

```
frequency = [0]*10
digit = int(input())
while 0 <= digit <= 9:
    frequency[digit] += 1
    digit = int(input())
for digit in range(10):
    print(digit*frequency[digit], end=' ')
```

16. Поразрядная сортировка. Применимость и ассимптотика алгоритма. Реализация на Python. (лек 8, *для коротких чисел и строк*)

```
A = [int(x) for x in input().split()]
max_num_len = max([len(str(x)) for x in A])
for radix in range(0, max_num_len):
    B = [[] for i in range(10)]
    for x in A:
        digit = (x // (10 ** radix)) % 10
        B[digit].append(x)
        A[:] = []
    for digit in range(10):
    A += B[digit]
```

17. Быстрая сортировка Хоара. Ассимптотика алгоритма. Реализация на Python.(лек 11)

```
from random import choice
def hoar_sort(A):
    if len(A) <= 1:
        return A
    barrier = choice(A)
    left = [x for x in A if x < barrier]
    middle = [x for x in A x == barrier]
    right = [x for x in A x > barrier]
    left = hoar_sort(left)
    right = hoar_sort(right)
    return left + middle + right
```

18. Сортировка слиянием. Ассимптотика алгоритма. Реализация на Python. (лек 11)

```
def merge_sort(A):
    if len(A) <= 1:
        return A
    left = A[:len(A) // 2]
    right = A[len(A) // 2:]
    left = merge_sort(left)
    right = merge_sort(right)
    return merge(left, right)
```

```
def merge(A, B):
    Res = []
    i = 0
    j = 0
    while i < len(A) and j < len(B):
        if A[i] < B[j]:
            Res.append(A[i])
            i += 1
        else:
            Res.append(B[j])
            j += 1
    Res += A[i:] + B[j:]  # один из срезов пуст
    return Res
```

19. Рекурсия. Прямой и обратный ход рекурсии. Стек вызовов при рекурсии. (лек 9, *рекурсия — решение задачи с использованием решения аналогичной подзадачи*)

20. Алгоритм Евклида. Реализация на Python через цикл и через рекурсию. (google, лек 9 БМ)

```
def my_gcd(a, b):
    if b == 0:
        return a
    else:
        return my_gcd(b, a%b)
```

21. Быстрое возведение в степень. Ассимптотика алгоритма. Реализация на Python. (лек 9)

```
def fast_power(a, n):
    if n == 0:
        return 1
    elif n%2 == 1:
        return a*fast_power(a, n — 1)
    else: # n%2 == 0
        return fast_power(a*a, n//2)
```

22. Вычисление чисел Фибоначчи. Реализация на Python через цикл и через рекурсию. (через цикл лек 3, через рекурсию лек 9, через динамику лек 10)

```
fib1 = 1
fib2 = 1
n = int(input())
i = 2
while i < n:
   fib_sum = fib2 + fib1
```

```
    fib1,  fib2 = fib2, fib_sum
    i += 1
print (fib_sum)
```

```
def fib(n):
    if n < 2:
        return n
    else:
        return fib(n - 1) + fib(n - 2)
```

23. Ханойские башни. Алгоритм и его реализация на Python. (лек 9)

24. Динамическое программирование. Сходство с рекурсией и отличие от неё. Когда рекурсия применима, а динамическое программирование нет.

25. Задача о количестве траекторий Кузнечика на числовой прямой. Реализация на Python. (лек 10)

26. Задача о траектории наименьшей стоимости для Кузнечика. Восстановление траектории наименьшей стоимости. Реализация на Python. (лек 10)

27. Двумерное динамическое программирование. Задача о количестве траекторий шахматного короля. Реализация на Python. (лек 10)

28. Наибольшая общая подпоследовательность. Ассимптотика алгоритма. Реализация.

29. Наибольшая возрастающая подпоследовательность. Ассимптотика алгоритма. Реализация на Python. (11 лек)

```
F = [0]*len(A)
for i in range(len(A)):
    for j in range(i):
        if A[j] < A[i] and and F[j] > F[i]:
            F[i] = F[j]
    F[i] += 1
print(max(F))
```

Генерация комбинаторных объектов. (лек 11)

30. Перегрузка операторов для классов в Python. (лаба 10)

31. Конструктор класса в Python. Классовые и экземплярные атрибуты. (лек 12)

32. Наследование классов в Python. Вызов конструктора надкласса. (лек 12)

33. Исключения в Python. Генерирование и перехват исключений. (лек 12)

34. Односвязный список на Python. Реализация при помощи класса LinkedList. Ассимптотика операций.

35. Стек. Использование стека для проверки корректности скобочной последовательности.

36. Двусвязный список на Python. Очередь.

37. Пирамида (куча). Реализация на Python. Ассимптотика добавления и удаления элемента в кучу.

38. Пирамидальная сортировка. Ассимптотика алгоритма. Реализация.

39. Открытая и закрытая хеш-таблица. Описать добавление элемента. Ассимптотика поиска. Без реализации на Python.

Вопросы по синтаксису Python 3

1. Ссылочная модель данных и динамическая типизация в Python. Сборщик мусора. (google)

2. Литералы чисел. Поддержка позиционных систем счисления в Python. (лек 3)

```
number = input()
x = int(number, 7)
```

3. Строки в Python. Экранируемые символы. Виды литералов строк в Python и их особенности. (\\, \", \', \r, \n, \t)(лек 5)

4. Условный оператор if и каскадная условная конструкция elif в Python. (конспект 4)

```
x = int(input())
if x < 0:
        print('negative')
elif x == 0:
        print('null')
elif x < 1:
```

```
        print('0 < x < 1')
elif x == 1:
        print('1')
else:  # x > 1
        print('x > 1')
```

5. Цикл while и управляющие операторы break и continue. Использование else после while. (лаба 3, конспект 1 лек)

6. Цикл for и его особенности в Python. Функция range(). (лаба 3)

7. Строки в Python. Методы find, count, replace, strip, upper, lower. (лаба 1, конспект 5)

8. Строки в Python. Срезы с двумя и тремя параметрами. (лаба 1, лек 5)

9. Кортежи переменных и множественное присваивание. (конспект 2 лек)

```
(x, y, z) = (1, 2, 3)
(x, y) = (y, x)
(x, y, z) = (z, x, y)
```

10. Списки в Python. Методы списков и операции со списками. (лек 5, лаба 5)

11. Списки в Python. Срезы списков. Присваивание в срез. Проблема копирования списка. (лек 5, лаба 5, проблема присваивания лек 5)

12. Списки в Python. List comprehensions: генерация списков. (лаба 5, лек 6)

13. Двумерные массивы (списки списков). Вложенная генерация. (лек 6)

14. Именованные параметры функций. Значения параметров по умолчанию. (лек 6)

Алгоритмические и теоретические вопросы

1. Позиционные системы счисления. Перевод числа из 10-й в произвольную систему счисления и наоборот. (лек 3, *преимущество позиционных СС в наличии "нуля" и в простоте алгоритмов уножения/деления*)

2. Связь двоичной, восьмеричной и шестнадцатеричной систем счисления. Примеры и обоснование. (лек 3)

3. Основы алгебры логики. Таблицы истинности И, ИЛИ, НЕ, XOR, импликации и эквиваленции.

Базовые логические операции НЕ, И, ИЛИ

A	не A
0	1
1	0

A	B	A и B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A или B
0	0	0
0	1	1
1	0	1
1	1	1

Дополнительные логические операции

Исключающее ИЛИ

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Импликация

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Эквивалентность

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

4. Основы алгебры логики. Свойства операций И и ИЛИ.

$$A \wedge \overline{A} = 0$$
$$A \wedge A = A$$
$$A \wedge 1 = A$$
$$A \wedge 0 = 0$$

$$A \vee \overline{A} = 1$$
$$A \vee A = A$$
$$A \vee 1 = 1$$
$$A \vee 0 = A$$

Формулы склеивания:

$$(A \wedge B) \vee (A \wedge \overline{B}) = A$$
$$(A \vee B) \wedge (A \vee \overline{B}) = A$$

Формулы поглощения:

$$A \vee (A \wedge B) = A$$
$$A \wedge (A \vee B) = A$$
$$A \vee (\overline{A} \wedge B) = A \vee B$$
$$A \wedge (\overline{A} \vee B) = A \wedge B$$

Переместительный закон:

$$A \vee B = B \vee A$$
$$A \wedge B = B \wedge A$$

Сочетательный закон:

$$(A \vee B) \vee C = A \vee (B \vee C)$$
$$(A \wedge B) \wedge C = A \wedge (B \wedge C)$$

5. Основы алгебры логики. Операция НЕ. Законы де Моргана. (*Эти законы связывают пары логических*

операций с помощью функции логического отрицания, то есть позволяют выразить одну логическую операцию с помощью другой)

6. Табличное задание логической функции. Дизъюнктивная нормальная форма. (лек 4)

7. Однопроходные алгоритмы: подсчёт, сумма, произведение. (лек 3)

8. Среднеквадратическое отклонение: однопроходный алгоритм. (лек 3)

9. Однопроходные алгоритмы: поиск максимума и подсчёт количества элементов, равных максимальному. (лек 4)

10. Однопроходные алгоритмы: нахождение трёх максимальных элементов. (лек 4)

11. Однопроходные алгоритмы: поиск местоположения максимума. (лек 4)

12. Алгоритм прверки простоты числа. Обоснование возможности остановки перебора на корне из числа. (лек 4 БМ)

```
def isPrime(x):
    "x is a prime"
    delitel = 2
    while delitel**2 < x:
        if x % delitel == 0:
            return False
        delitel += 1
    return True
```

13. Алгоритм разложения числа на множители. (лек 4 БМ)

14. Алгоритм обращения массива. (лек 5)

15. Структурное программирование. Декомпозиция задачи и проектирование «сверху-вниз». (лек 6)