1. Сложность жадного алгоритма множественного выравнивания

Ответ: $O(k^2n^2)$.
Решение:
Алгоритм включает в себя $k-1$ операций объединения двух последоватеьностей или профилей в один профиль (т.к. в итоге должен остаться один профиль), каждая операция включает в себя выравнивание всех оставшихся профилей на полученный на предыдущем шаге, каждое выравнивание имеет сложность $O(n^2)$, на первом шаге число пар - $\frac{k(k-1)}{2}$ = C_k^2 , на следующих - $(k-1), (k-2), \cdots, 1$. Такая сумма пропорциональна квадрату, так как пропорциональна интегралу $\int_0^k (k-i)di$ (можем дать интегральную оценку сверху и снизу, отняв от подынтегральной функции константу), первый член тоже пропорционален квадрату по k, значит, всё вместе пропорционально квадрату по k, и квадрат по n получаем из сложности каждого выравнивания, всё вместе - $O(k^2n^2)$, Ч.Т.Д.

7.

CGATG

```
                    C         G            A              T        G
State_1 0.5  0.1*0.5 0.1*0.1*0.5 0.4*0.1*0.1*0.5  0.4*2.e-3  8.e-5
                    \                       =2.e-3       /          \
State_2 0.5  0.4*0.5-0.5*0.4*0.5-0.05*0.5*0.4*0.5 0.05*5.e-3 1.25e-4

=5.e-3
The resulting sequence of hidden states:
22212
```

8.

3 скрытых и 4 открытых состояния:

Начальное распределение на скрытых состояниях - 2 параметра (сумма вероятностей 1, поэтому не три)
Матрица переходов: 3 состояния *2 вероятности переходов для каждого = 6 независимых переменных (опять -1 за счёт нормировки вероятности)*
Матрица эмиссии: 3→4, 3 3 вероятности эмиссии, 9 параметров.

Всего: 2+6+9=17 параметров

9.BWT

```
def bwt(s):
        R = list(range(len(s)))
    R.sort(key = lambda x: s[x:]+s[:x])
        return ''.join(list(map(lambda x: s[x-1], R)))
```

```
>>> bwt('TAGAGCTA')
'TGTGAAAC'
```

```
def sufarr(s):
        sarr = list(map(lambda x: x[1], [(s[0:i], i) for i in
range(len(s))].sort(key = lambda x: x[0])))
        return sarr
```

Где-то здесь ошибка, но у меня уже нет времени дебажить, и это сложно делать в командной строке, думаю, из кода понятно, какой алгоритм

3. Скобочное представление

..(((((.((....)))))))