Домашнее задание

1. На вход задачи подаётся число n и последовательность целых чисел $a_1, \ldots, a_n$. Необходимо найти номера i и j $(1 \leqslant i < j \leqslant n)$, такие что сумма $\sum_{k=i}^{j} a_k$ максимальна.

1. Постройте линейный алгоритм, решающий задачу.

2. Постройте онлайн-алгоритм (достаточно выполнить только этот пункт).

2. Постройте алгоритм, который выводит все двоичные строки длины n, в которых нет двух единиц подряд.

3. Рассмотренный нами алгоритм вычисления расстояния редактирования строк длины m и n заполняет таблицу размера $O(mn)$. При больших m и n такому алгоритму просто не хватит памяти. Как можно обойтись меньшим объёмом памяти?

1. Допустим, что нас интересует только расстояние редактирования, но не соответсвующее оптимальное выравнивание. Покажите, что тогда в каждый момент не нужно хранить всю таблицу и можно обойтись объёмом памяти $O(n)$.

2. Теперь допустим, что мы хотим найти и оптимальное выравнивание. Легко видеть, что это эквивалентно поиску кратчайшего пути из вершины $(0, 0)$ в вершину (n, m) в соответсвующем графе: вершины графа — клетки таблицы, а стоимость ребра — 1 или 0, в зависимости от конкретного перехода. Любой такой путь должен проходить через вершину $(k, m/2)$ для некоторого k. Модифицируйте алгоритм поиска расстояния редактирования, чтобы он заодно выдавал и k. Считайте, что m — степень двойки.

3*. Рассмотрим теперь следующий рекурсивный алгоритм.

```
1 Function FindPath((0,0) → (n,m)) :
2 |   Вычислить k (см. предыдущий пункт);
3 |   FindPath((0,0) → (k,m/2));
4 |   FindPath((k,m/2) → (n,m));
5 |   return объединение найденных двух путей
6 end
```

Покажите, что по данной схеме алгоритм можно реализовать так, чтобы время его работы было $O(nm)$, а память — $O(n)$.

4. Постройте алгоритм, который, получив на вход две строки x и y, находит их самую длинную общую подстроку (непрерывную подпоследовательность $x[i]x[i+1]\ldots x[i+m] = y[j]y[j+1]\ldots y[j+m]$).

5. На столе лежат в ряд n банковских карт, на счету которых находится $v_1, v_2, \ldots v_n$ долларов. Два игрока по очереди берут карты, причём можно брать только крайнюю карту (с любой стороны), до тех пор, пока не кончатся все карты. Каждый из игроков, заинтересован в максимальной суммарной стоимости взятых им карт. Считайте, что n чётно.

1. Покажите, что жадная стратегия (брать ту из карт, которая дороже) не оптимальна: уже первый ход по этому правилу может помешать достичь оптимума.

2. Постройте алгоритм отыскания оптимальной стратегии для первого игрока за время $O(n^2)$. Получив на вход $v_1, \ldots, v_n$, алгоритм должен произвести препроцессинговые вычисления

за $O(n^2)$, а вычисление каждого следующего хода должно выполняться за $O(1)$ шагов (с использованием сохранённой информации).

3. Представьте теперь, что задача игрока не максимизировать стоимость карт, а просто набрать стоимость больше, чем у соперника. Постройте жадное решение для этой задачи.