Программирование алгоритмов на C++. Задачи

- **Задача 1**
  Найти максимальное расстояние (модуль разности) между чётными числами в последовательности, так что оба числа окружены нечётными (если крайний элемент, то проверяем только одного соседа).

- **Задача 2. Ходы коня**
  Вычислите, сколько полей на шахматной доске могут быть конечной точкой пути коня за M ходов из заданной точки (её можно задавать случайно).

- **Задача 3. Случайные точки на сфере**
  Сгенерируйте M случайных точек на единичной сфере в пространстве относительно равномерного распределения. Вычислите, сколько из них находятся на расстоянии $< a(= 0.1)$ от треугольника, построенного на каких-либо трёх других точках. Как оценивается сложность вашего алгоритма?

- **Задача 4. Максимум справа за один проход**
  Дан массив a длины n. Найдите за один проход по массиву $max_j \sum_{k=j}^{n} a[k]$. Ограничение по памяти – $O(1)$.

- **Задача 5. Максимум максимумов**
  Дан массив a длины n без нулей. Для k-ого подмассива с элементами одного знака (пусть его начальный и конечный индексы – i_1 и i_2) обозначим $m_k = max_j\{S'_j, S''_j\}$, где $S'_j = \sum_{r=i_1}^{j}(x_r mod\ 5-2)$, а $S''_j = \sum_{r=j}^{i_2}(x_r mod\ 7-3)$. Найти $max_k\{m_k\}$ Ограничение по времени – $O(n)$, по памяти – $O(1)$.

- **Задача 6. Минимум максимумов.**
  Дан массив a длины n без нулей. Для k-ого подмассива с элементами одного знака (пусть его начальный и конечный индексы – i_1 и i_2) обозначим $m_k = max_j\{S'_j, S''_j\}$, где $S'_j = \sum_{r=i_1}^{j} x_r$,

а $S''_j = \sum_{r=j}^{i_2} x_r$. Найти $min_k\{m_k\}$ Ограничение по времени – $O(n)$, по памяти – $O(1)$.

- **Задача 7. Окна**
  Есть большая прямоугольная таблица символов, могут встречаться: '#' и пробел. Пустые области - это окна. Написать алгоритм, который максимально быстро определяет, являются ли все окна прямоугольными. Какова его сложность?

- **Задача 8. Лабиринт**
  Есть большая прямоугольная таблица символов, могут встречаться: '#' и пробел. '#' соответствует стенкам лабиринта, пробелы - дорожкам. В левой нижней и правой верхней позициях пробелы. Написать алгоритм, который определяет, есть ли путь из левого нижнего угла в правый верхний.