Задание 1. Разработайте и выполните эксперимент по сравнению эффективности работы двух алгоритмов сортировки. Оцените устойчивость алгоритмов*.

АЛГОРИТМ $SelectionSort\,(A\,[0..n-1])$

// Сортировка массива методом выбора
// **Входные данные:** Массив $A[0..n-1]$ упорядочиваемых
// элементов
// **Выходные данные:** Массив $A[0..n-1]$, отсортированный
// в неубывающем порядке
for $i \leftarrow 0$ **to** $n-2$ **do**
  $min \leftarrow i$
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[j] < A[min]$
      $min \leftarrow j$
  Обмен $A[i]$ и $A[min]$

АЛГОРИТМ $Mergesort\,(A\,[0..n-1])$

// Рекурсивно сортирует массив $A[0..n-1]$
// **Входные данные:** Массив $A[0..n-1]$ упорядочиваемых
// элементов
// **Выходные данные:** Отсортированный в неубывающем порядке
// массив $A[0..n-1]$
if $n > 1$
  Копировать $A[0..\lfloor n/2 \rfloor - 1]$ в $B[0..\lfloor n/2 \rfloor - 1]$
  Копировать $A[\lfloor n/2 \rfloor..n-1]$ в $C[0..\lfloor n/2 \rfloor - 1]$
  $Mergesort(B[0..\lfloor n/2 \rfloor - 1])$
  $Mergesort(C[0..\lfloor n/2 \rfloor - 1])$
  $Merge(B, C, A)$

АЛГОРИТМ $Merge\,(B\,[0..p-1]\,.C\,[0..q-1]\,,A\,[0..p+q-1])$

// Слияние двух отсортированных массивов в один
// **Входные данные:** Сортированные массивы $B[0..p-1]$
// и $C[0..q-1]$
// **Выходные данные:** Сортированный массив $A[0..p+q-1]$,
// состоящий из элементов B и C
$i \leftarrow 0$; $j \leftarrow 0$; $k \leftarrow 0$
while $i < p$ **and** $j < q$ **do**
  if $B[i] \leqslant C[j]$
    $A[k] \leftarrow B[i]$; $i \leftarrow i+1$
  else
    $A[k] \leftarrow C[j]$; $j \leftarrow j+1$
  $k \leftarrow k+1$
if $i = p$
  Копировать $C[j..q-1]$ в $A[k..p+q-1]$
else
  Копировать $B[i..p-1]$ в $A[k..p+q-1]$

* Алгоритм сортировки называется устойчивым, если в нем сохраняется относительный порядок любых двух равных элементов, находящихся во входном списке. Другими словами, если во входном списке есть два одинаковых элемента, номера которых равны i и j, причем i < j, то в отсортированном списке, где их номера будут, соответственно, равны i' и j', будет выполняться отношение i' < j'.

Задача 2. Разработайте и выполните эксперимент по сравнению эффективности работы двух версий последовательного поиска.

АЛГОРИТМ $SequentialSearch\,(A\,[0..n-1]\,,K)$
// **Входные данные:** массив чисел $A[0..n-1]$ и ключ поиска K
// **Выходные данные:** возвращается индекс первого элемента
// массива A, который равен K, либо -1,
// если заданный элемент не найден
$i \leftarrow 0$
while $i < n$ **and** $A[i] \neq K$ **do**
$i \leftarrow i + 1$
if $i < n$
return i
else
return -1

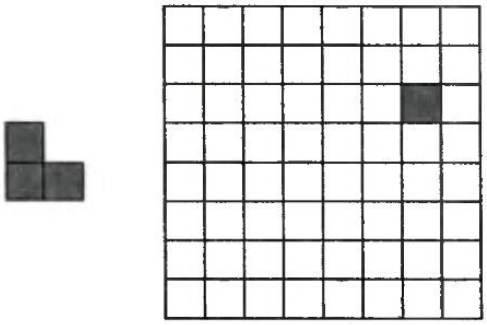
Версия 1.

Версия 2.

АЛГОРИТМ $SequentialSearch2\,(A\,[0..n]\,,K)$
// Алгоритм реализует последовательный поиск
// с использованием ключа поиска в качестве ограничителя
// **Входные данные:** Массив A из n элементов и ключ поиска K
// **Выходные данные:** Позиция первого элемента массива
// $A[0..n-1]$, значение которого совпадает
// с K; если элемент не найден, возвращается
// значение -1
$A[n] \leftarrow K$
$i \leftarrow 0$
while $A[i] \neq K$ **do**
$i \leftarrow i + 1$
if $i < n$
return i
else
return -1

Задача 3.

Триомино — элемент мозаичного заполнения в форме L, образованный тремя квадратами шахматной доски. Задача состоит в покрытии триомино шахматной доски размером 2^n x 2^n с одной вырезанной в произвольном месте клеткой. Триомино должны покрывать все клетки, за исключением вырезанной, без пропусков и перекрытий. Разработайте алгоритм методом декомпозиции.



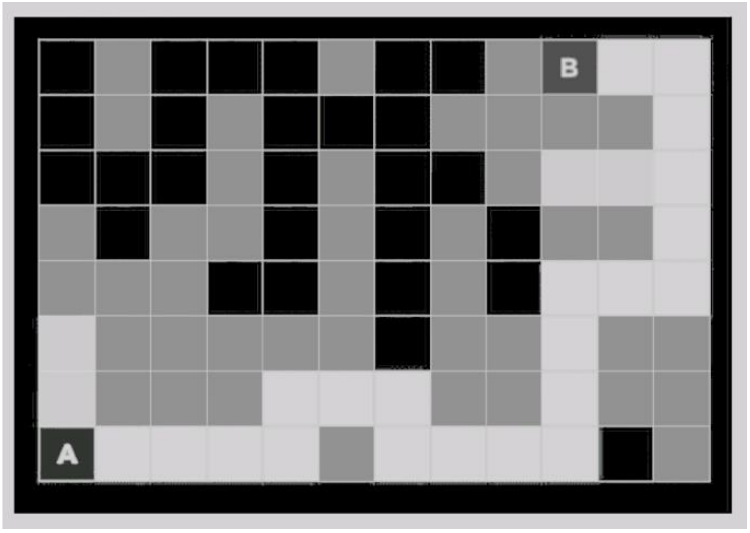
Задача 4.

Какой из алгоритмов (DFS) или (BFS) найдет более короткий путь через лабиринт?

1. **DFS всегда найдет более короткий путь, чем BFS**
2. **BFS всегда найдет более короткий путь, чем DFS**
3. **DFS иногда, но не всегда, находит более короткий путь, чем BFS**
4. **BFS иногда, но не всегда, находит более короткий путь, чем DFS**
5. **Оба алгоритма всегда будут находить пути одинаковой длины**

Задача 5.

Серые клетки (см. рис.) указывают на стены. Алгоритм поиска был запущен в этом лабиринте и нашел выделенный путь из точки A в точку B. При этом выделенные черным цветом клетки были исследованными состояниями, но это не привело к цели**.**



Какой из четырех алгоритмов поиска, рассмотренных в лекции, — поиск по глубине, поиск по ширине, жадный поиск по наилучшему варианту с эвристикой Манхэттенского расстояния и поиск A* - (или несколько, если возможно несколько) может быть использован?

1. Может быть только A*

2. Может быть только GBFS

3. Может быть только DFS

4. Может быть только BFS

5. Это может быть либо A*, либо GBFS

6. Это может быть либо DFS, либо BFS

7. Это может быть любой из четырех алгоритмов

8. Не может быть ни одного из четырех алгоритмов