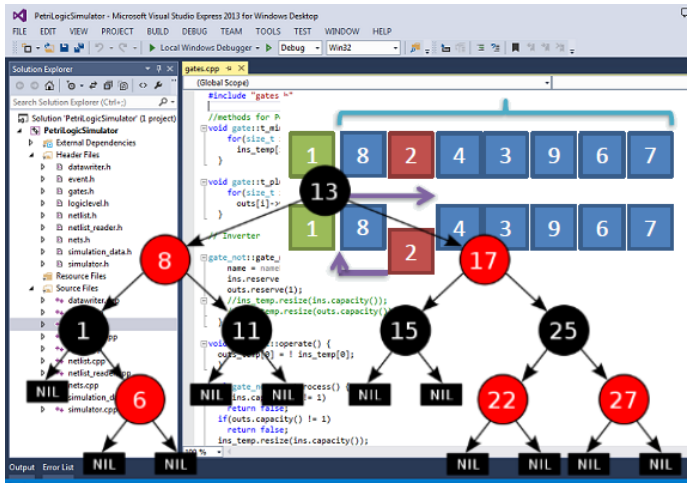
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники (группы ЭН-24-25, каф. ПКИМС)

Теория алгоритмов



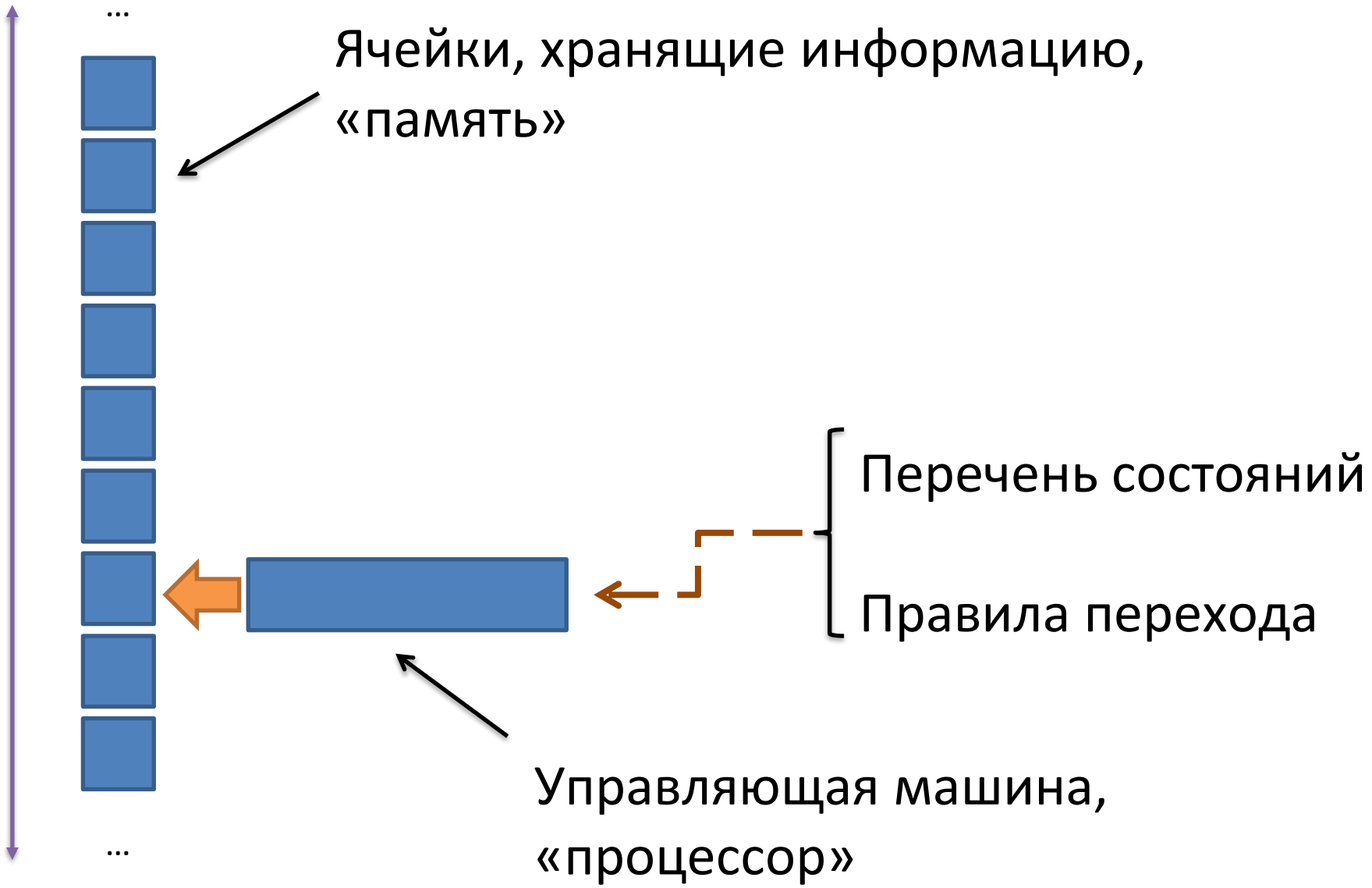
Лекция 1

Общие сведения об алгоритмах

Алгоритм: определение

Алгоритм - это организованная последовательность действий, понятных для некоторого исполнителя, ведущая за конечное число шагов к решению поставленной задачи.

Машина Тьюринга



Алан Мэтисон Тьюринг
(1912 - 1954)

Тезисы Чёрча-Тьюринга

1. Любая функция, которая может быть вычислена физическим устройством, может быть вычислена машиной Тьюринга

Алан Мэтисон Тьюринг
(1912 - 1954)

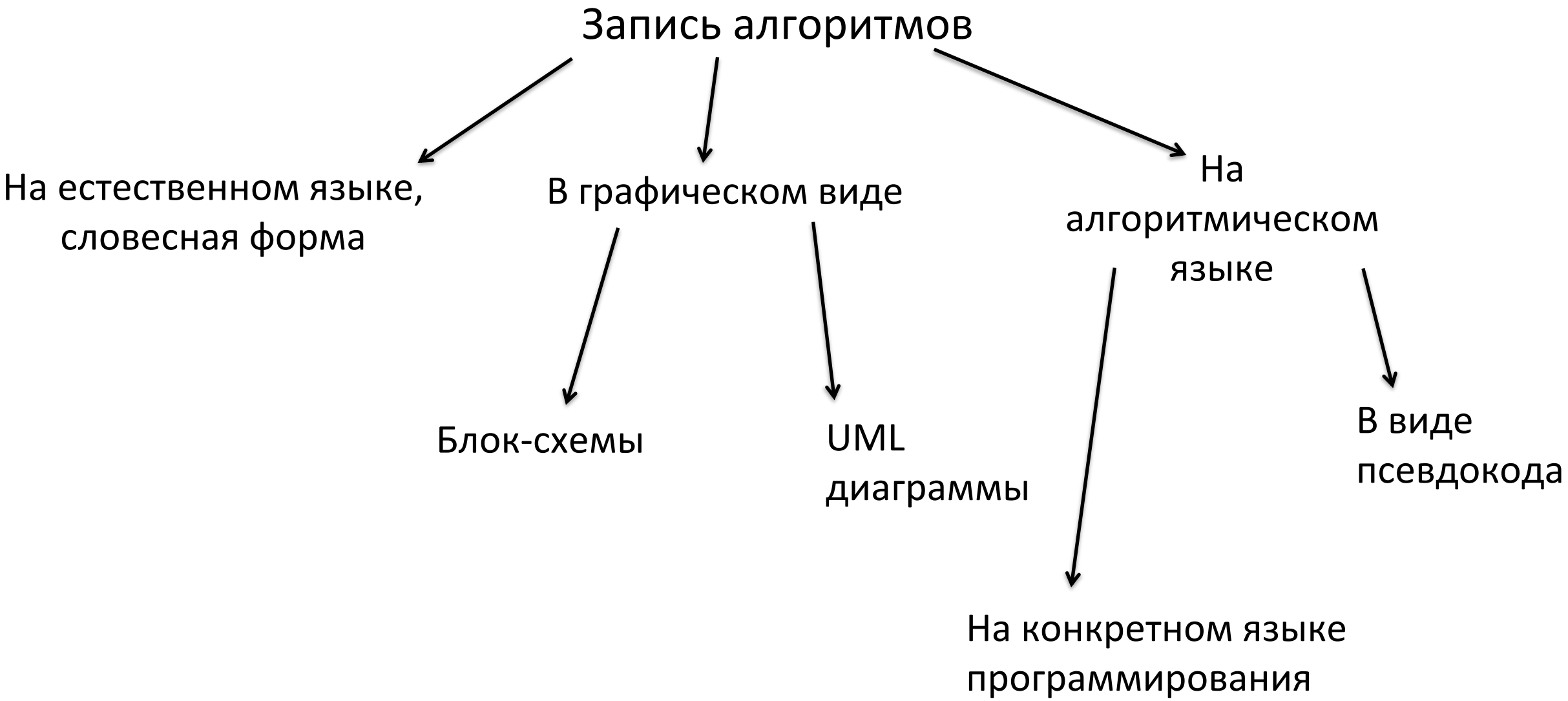
Алонзо Чёрч
(1903 - 1995)

2. Любой конечный физический процесс, не использующий аппарат, связанный с непрерывностью и бесконечностью, может быть вычислен физическим устройством

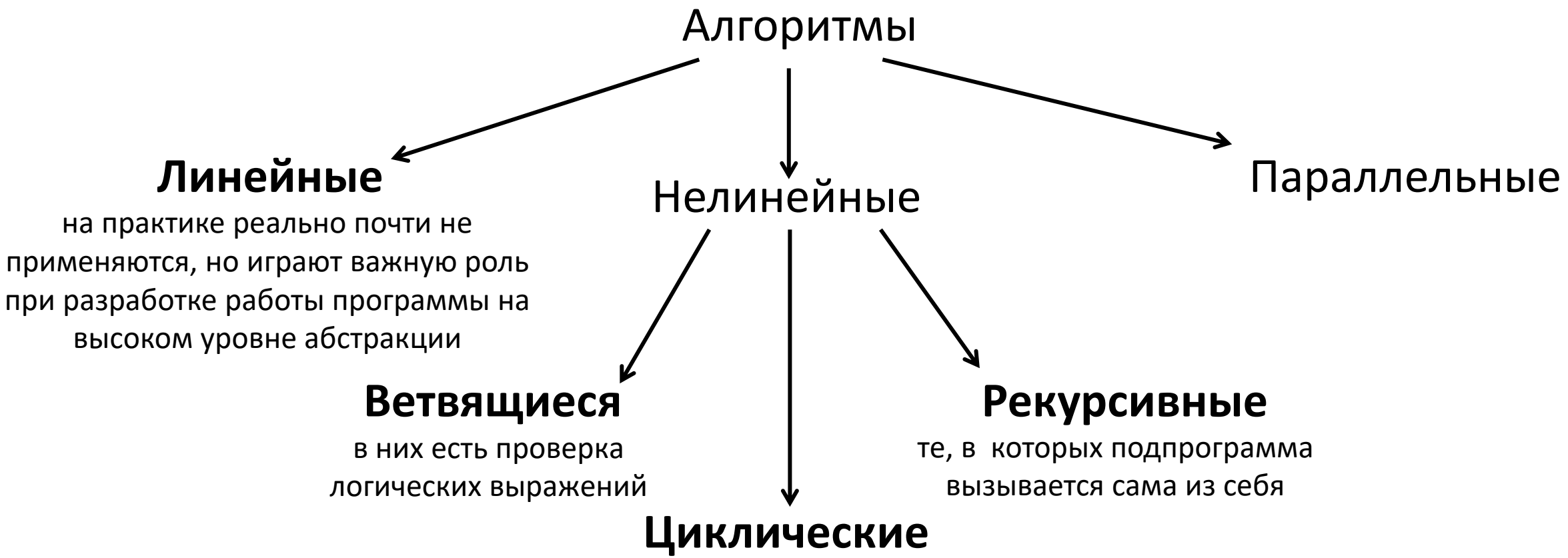
Свойства алгоритмов

- **Дискретность** - алгоритм состоит из последовательности отдельных шагов - элементарных действий, выполнение которых на заданном уровне абстракции не представляет сложности и вполне понятно.
- **Корректность** - если алгоритм создан для решения определенной задачи, то для всех исходных данных он должен всегда давать правильный результат и ни для каких исходных данных не будет получен неправильный результат
- **Детерминированность** - при задании одних и тех же исходных данных несколько раз алгоритм будет выполняться абсолютно одинаково и всегда будет получен один и тот же результат
- **Массовость** - с помощью алгоритма можно решать не одну конкретную задачу, а любую задачу из некоторого класса однотипных задач при всех допустимых значениях исходных данных
- **Конечность** - последовательность элементарных действий алгоритма не может быть бесконечной, неограниченной, хотя может быть очень большой
- **Результативность** - выполнение алгоритма обязательно должно привести к решению поставленной задачи, либо к сообщению о том, что при заданных исходных величинах задачу решить невозможно

Формы записи алгоритмов



Виды алгоритмов с точки зрения их исполнения



Виды алгоритмов по принципу работы

Детерминированные (жёсткие) алгоритмы – предполагают одну и ту же последовательность действий вне зависимости от входных данных.

Вероятностные (гибкие, адаптивные) алгоритмы – предполагают различное решение в зависимости от некоторых случайным образом генерируемых данных.

Эвристические (статистически верные) алгоритмы – не имеют чёткого обоснования, но дают верный ответ в большинстве случаев.

Парадигмы разработки алгоритмов: Линейная

Линейная

Разделяй и властвуй

Динамическое программирование

Реализация «в ширину» и «в глубину»

К линейной парадигме разработки относят:

1. большинство вычислительных алгоритмов: методы решения СЛАУ;
2. «медленные» алгоритмы поиска;
3. «медленные» алгоритмы сортировки данных.

Парадигмы разработки алгоритмов: «Разделяй и властвуй» (1)

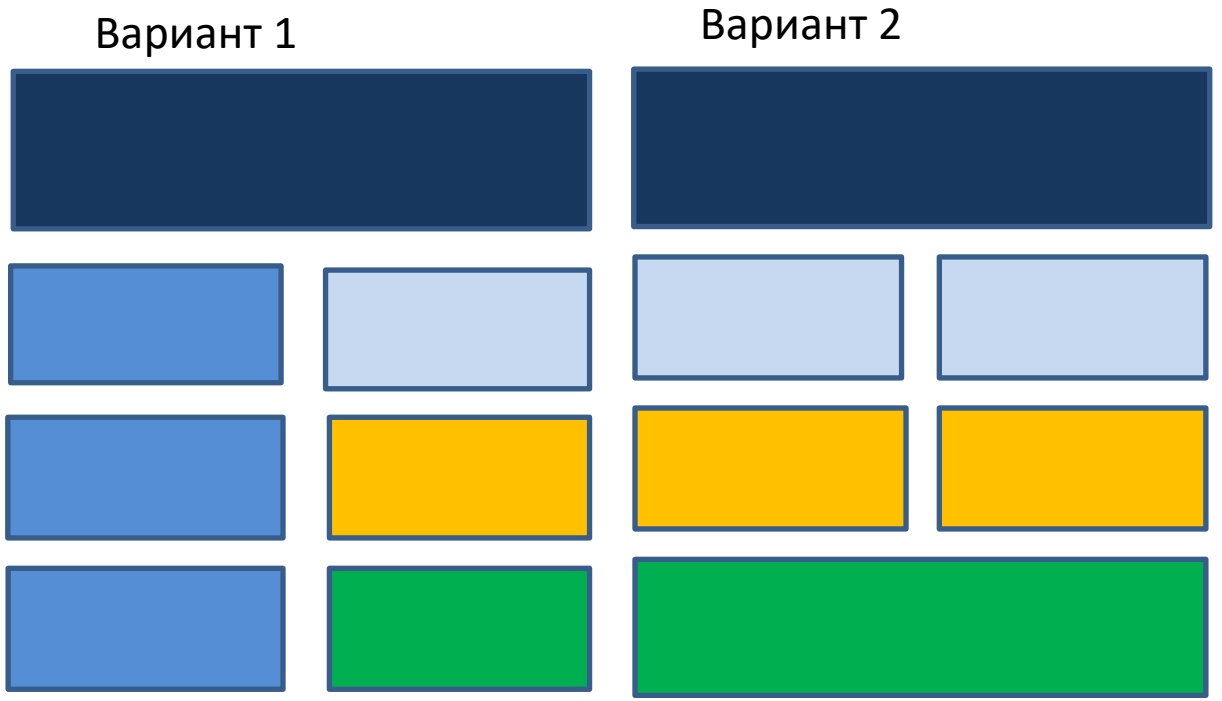
Линейная

Разделяй и властвуй

Динамическое программирование

Реализация «в ширину» и «в глубину»

Жадные алгоритмы



Основные этапы:

1. разделить целостные данные на части;
2. обработать выбранный/каждый блок данных;
3. при необходимости – объединить результаты обработки разных блоков;

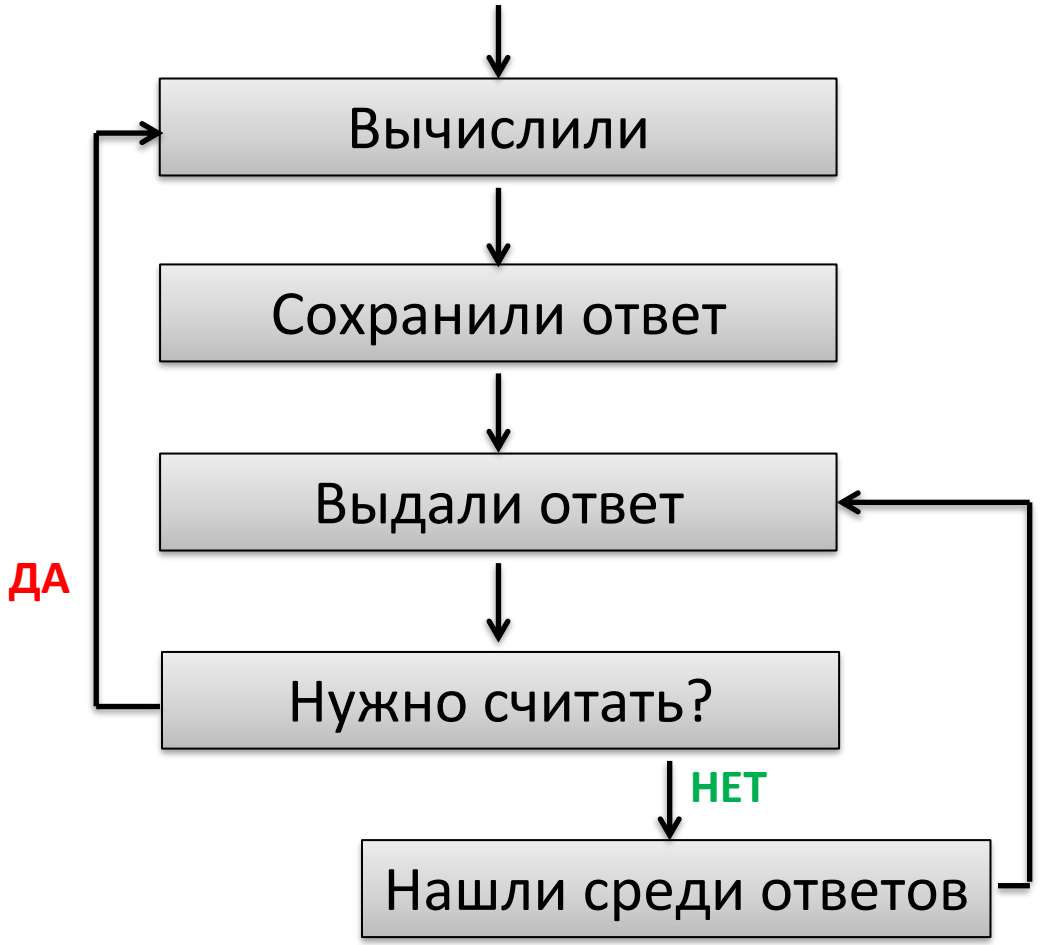
К парадигме «разделяй и властвуй» относят:

1. некоторые алгоритмы вычислительной математики – например, метод деления отрезка пополам;
2. быстрые алгоритмы поиска (на основе отсортированных данных, хэшей, деревьев);
3. быстрые алгоритмы сортировки данных;
4. все алгоритмы, имеющие параллельную реализацию.

Парадигмы разработки алгоритмов: динамическое программирование (2)

Линейная
Разделяй и властвуй
Динамическое программирование
Реализация «в ширину» и «в глубину»
Жадные алгоритмы

Динамическая парадигма предполагает сохранение результатов предыдущих вычислений.



Парадигмы разработки алгоритмов: реализация «в ширину» и «в глубину»

Линейная

Разделяй и властвуй

Динамическое программирование

Реализация «в ширину» и «в глубину»

Жадные алгоритмы

```
long Fib_NoRec(long A) {
  long a = 0, b = 1, c = 0;
  while (A - 1) {
    c = a + b;
    a = b;
    b = c;
    --A;
  }
  return c;
}
```

«в ширину»
(без применения рекурсии)

```
long Fib_Rec(long A) {
  if (A == 0 || A == 1)
    return A;

  return (Fib_Rec(A - 1) + Fib_Rec(A - 2));
}
```

«в глубину»
(с применением рекурсии)

Парадигмы разработки алгоритмов: жадные алгоритмы

Линейная
Разделяй и властвуй
Динамическое программирование
Реализация «в ширину» и «в глубину»
Жадные алгоритмы

Жадные алгоритмы, получая первое же подходящее (и, вполне вероятно, не оптимальное) решение, отказываются от рассмотрения всех остальных, принимая его за конечное решение

Чем определяется сложность алгоритма?

Сложность алгоритма определяется числом некоторых операций в зависимости от числа обрабатываемых элементов.

«О большое» – математическое обозначение для сравнения асимптотического поведения функций

$$f(n) = O(g(n))$$

Такая запись читается следующим образом: функция f(n) асимптотически ограничена сверху функцией g(n)

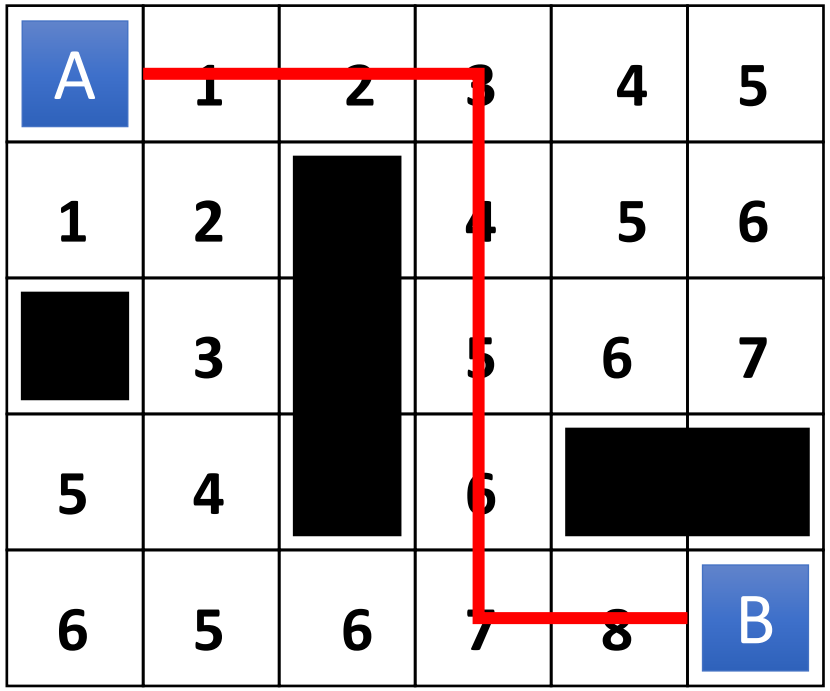
O(N)
O(N·logN)
O(N+M)
O(N^2)

Алгоритмическая сложность может измеряться для:
- худшего случая;
- среднего случая;
- лучшего случая.

Алгоритмическая сложность может измеряться для:
- числа операций;
- требуемого объёма памяти.

Волновой алгоритм (алгоритм Ли) нахождения кратчайшего пути

Задача: найти кратчайшее расстояние между двумя точками на плоскости



Шаг 1:
нанести на поле сетку, препятствия, начальную (A) и конечную (B) точки

Шаг 2:
проставить первую волну вокруг начальной точки A

Шаг 3:
запустить цикл обхода всей матрицы, проставляя следующие номера волны вокруг текущего номера, пока не дойдём до финиша (B)

Шаг 4:
найти обратный путь по наименьшим номерам, начиная с точки финиша (B)