Алгоритмы и структуры данных

Даниленко Иван Николаевич, доцент кафедры АиКС

Алгоритмы и структуры данных

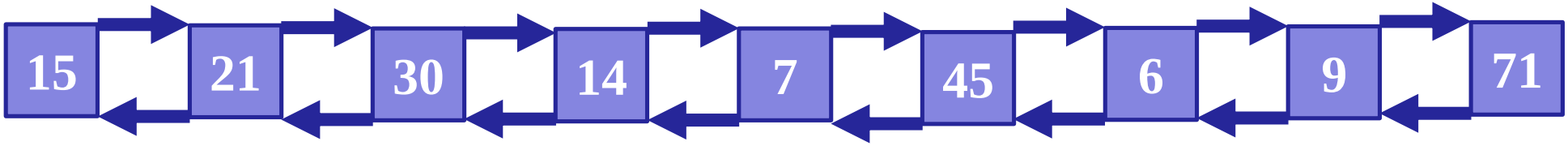
- Структуры данных и абстрактные структуры данных
  - Асимптотическая сложность
  - Анализ алгоритмов
- Алгоритмы сортировки
  - Сортировка, основанная на сравнениях
  - Цифровые сортировки
- Алгоритмы поиска
  - «Линейный» поиск
  - Поиск строк
- Деревья
  - Общие сведения о деревьях
  - Деревья поиска, основанные на сравнениях
  - Цифровые деревья
- Графы
  - представление графов
  - базовые алгоритмы на графах

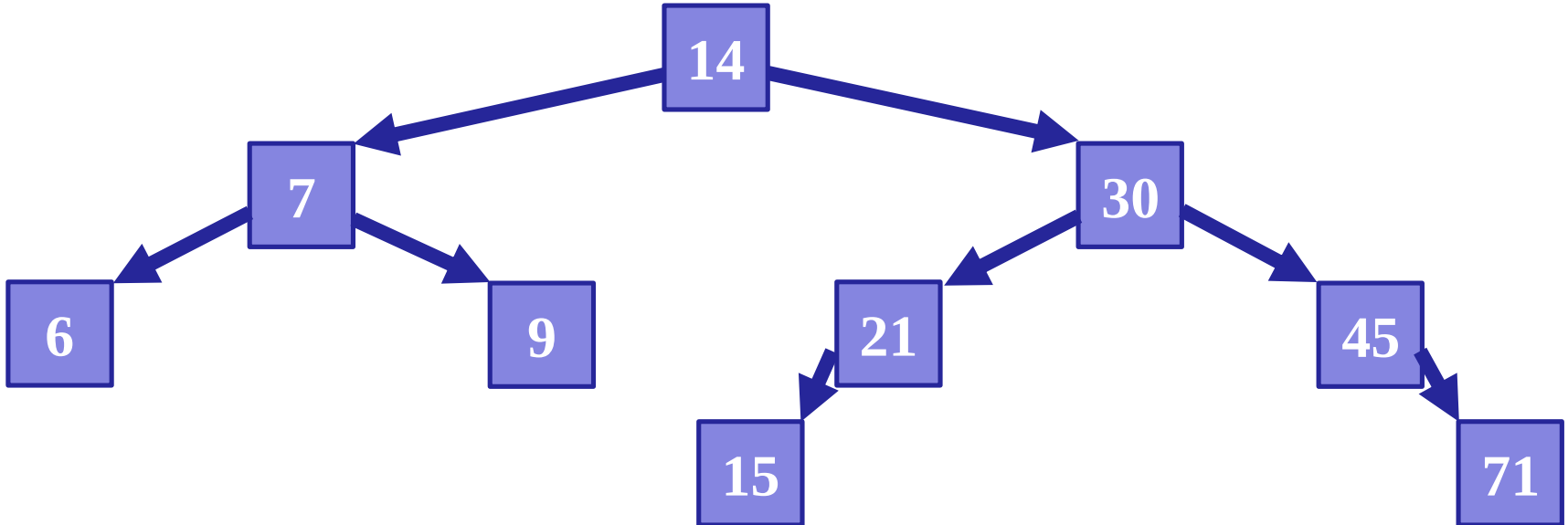
Какой алгоритм «лучше»?

```
void Alg_1(Key A[], int Size){
  int i, j;
  for(i = 0; i < Size; i++)
     for(j = 1; j < Size-i; j++)
        if(A[j-1]>A[j])
           Swap(A[j-1],A[j]);
  }
```

```
void Alg_2(Key A[], int Size) {
  int i, j;
  Key V;
  for (i = 1; i < Size; i++){
     V = A[i];
     for(j = i; j > 0 && A[j-1] > V; j--)
        A[j] = A[j-1];
     A[j] = V;
     }
  }
```

Какая структура данных «лучше»?





Литература

(из рабочей программы дисциплины)

основная

- Вирт, Никлаус. Алгоритмы и структуры данных [Электронный ресурс] / Никлаус Вирт ; пер. Ф. В. Ткачева. - Саратов : Профобразование, 2019. - 272 с. ил.
- Сундукова, Т. О. Структуры и алгоритмы компьютерной обработки данных [Электронный ресурс] : Учебное пособие / Т. О. Сундукова, Г. В. Ваныкина. Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. - 804 с.

дополнительная

- Самуйлов, С. В. Алгоритмы и структуры обработки данных : учебное пособие [Электронный ресурс] / С. В. Самуйлов. — Саратов : Вузовское образование, 2016. — 132 с.
- Медведев, Д. М. Структуры и алгоритмы обработки данных в системах автоматизации и управления : учебное пособие / Д. М. Медведев. — Саратов : Ай Пи Эр Медиа, 2018. — 100 с.
- Назаренко, П. А. Алгоритмы и структуры данных : учебное пособие / П. А. Назаренко. — Самара : Поволжский государственный университет телекоммуникаций и информатики, 2015. — 130 с.
- Белов, В. В. Алгоритмы и структуры данных: Учебник / Белов В.В., Чистякова В.И. - Москва :КУРС, НИЦ ИНФРА-М, 2020. - 240 с.

методическая

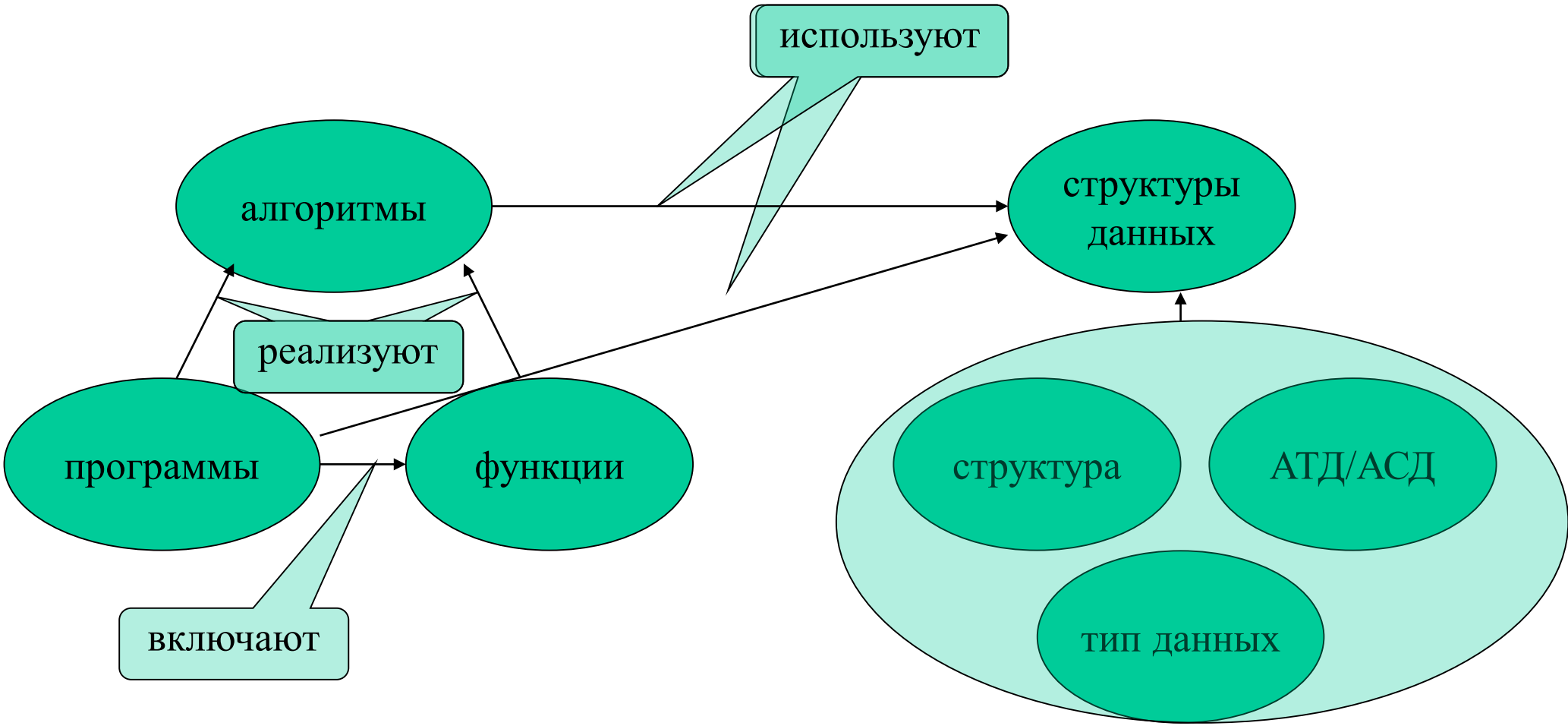
- Синюк, В. Г. Алгоритмы и структуры данных : лабораторный практикум. Учебное пособие / В. Г. Синюк, Ю. Д. Рязанов. — Белгород : Белгородский государственный технологический университет им. В.Г. Шухова, ЭБС АСВ, 2013. — 204 с.

Литература

(прочие источники)

- Ахо, А. Структуры данных и алгоритмы / А. Ахо, Дж. Хопкрофт, Д. Ульман. – М.: Вильямс, 2003. – 384 с.: ил.
- Кубенский, А. А. Структура и алгоритмы обработки данных: объектно-ориентированный подход и реализация на C++: учебное пособие / А. А. Кубенский. – СПб.: БХВ-Петербург, 2004. – 464 с.: ил.
- Программирование алгоритмов обработки данных: учебное пособие / О. Ф. Ускова [и др.]. – СПб.: БХВ-Петербург, 2003. – 188 с.: ил.
- Новиков, Ф. А. Дискретная математика для программистов: учебное пособие для студентов высших учебных заведений / Ф. А. Новиков. – 2-е изд. – СПб.: Питер, 2005. – 363 с.: ил.
- Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. - М.: Изд-во Моск. Центра непрерыв. мат. образования, 2001. - 955 с.
- Ахо, А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. - М.: Мир, 1979. - 536с.
- Гудрич, М. Т. Структуры данных и алгоритмы в Java / М. Т. Гудрич, Р. Тамассия. - Минск: Новое знание, 2003. - 670 с.: ил.
- Кнут Д. Искусство программирования для ЭВМ. Т.1. Основные алгоритмы / Д. Кнут. – М.: Вильямс, 2000. – 720 с.: ил.
- Хусаинов, Б. С. Структуры и алгоритмы обработки данных: примеры на языке Си: учебное пособие / Б. С. Хусаинов.- М.: Финансы и статистика, 2004.- 463, [1] с.: ил.
- Вирт, Н. Алгоритмы и структуры данных / Н. Вирт ; [пер. с англ. Д. Б. Подшивалова] .— [2-е изд., испр.] .— СПб. : Невский диалект, 2007 .— 351 с. : ил.
- Седжвик, Р. Фундаментальные алгоритмы на С. Части 1-5. / Р. Седжвик. Киев : ДиаСофтЮП, 2003. 1136 с. : ил.
- Топп, У. Структуры данных в C++: Пер. с англ. - М. : ЗАО "Издательство БИНОМ", 2000. - 816 с. Ж ил.
- Algolist – алгоритмы, методы, исходники [Электронный ресурс]. – 200-. – Режим доступа: http://algolist.manual.ru/, свободный. – Загл. с экрана

Основные понятия и их взаимосвязи



Абстрактные типы данных (АТД)

Структура – множество объектов и связей между ними.

Структура данных – множество элементов данных, объединенных и упорядоченных определенным образом.

Тип данных – это множество значений и операций над этими значениями (Седжвик).

Абстрактный тип данных – это тип данных, определенный только операциями, применимыми к объектам данного типа, без учета его внутренней организации и внутренней реализации.

Абстрактные типы данных предназначены для удобного хранения и доступа к информации.

Абстрактный тип данных – это тип данных, доступ к которому осуществляется только через интерфейс (Седжвик).

Абстрактный тип данных – группа тесно связанных между собой данных и методов (функций), которые могут осуществлять операции над этими данными.

Алгоритм

Алгоритм – одно из основных понятий математики, не обладающих формальным определением в терминах более простых понятий, а абстрагируемое непосредственно из опыта. *Алгоритм* – это точное предписание, которое задает [алгоритмический] вычислительный процесс, начинающийся с произвольного исходного данного и направленный на получение полностью определяемого этим исходным данным результата. Для каждого алгоритма можно выделить 7 характеризующих его (не независимых!) параметров: 1) совокупность возможных исходных данных, 2) совокупность возможных результатов, 3) совокупность промежуточных результатов, 4) правило начала, 5) правило непосредственной переработки, 6) правило окончания, 7) правило извлечения результата (БСЭ).

Алгоритм – это точное предписание исполнителю совершить определенную последовательность действий для достижения поставленной цели за конечное число шагов (ГЕС).

Алгоритм – это последовательность действий (операций) и правил их выполнения или команд, предназначенных для решения определенной задачи или группы задач.

Алгоритм – описание метода решения задачи, пригодного для реализации в виде компьютерной программы. (Седжвик)

…

Черч, Тьюринг, Марков, Пост

Программа

Программа – последовательность машинных команд, предназначенная для достижения конкретного результата (ГЕС).

Программа – упорядоченная последовательность действий для ЭВМ, реализующая алгоритм решения некоторой задачи (БСЭ).

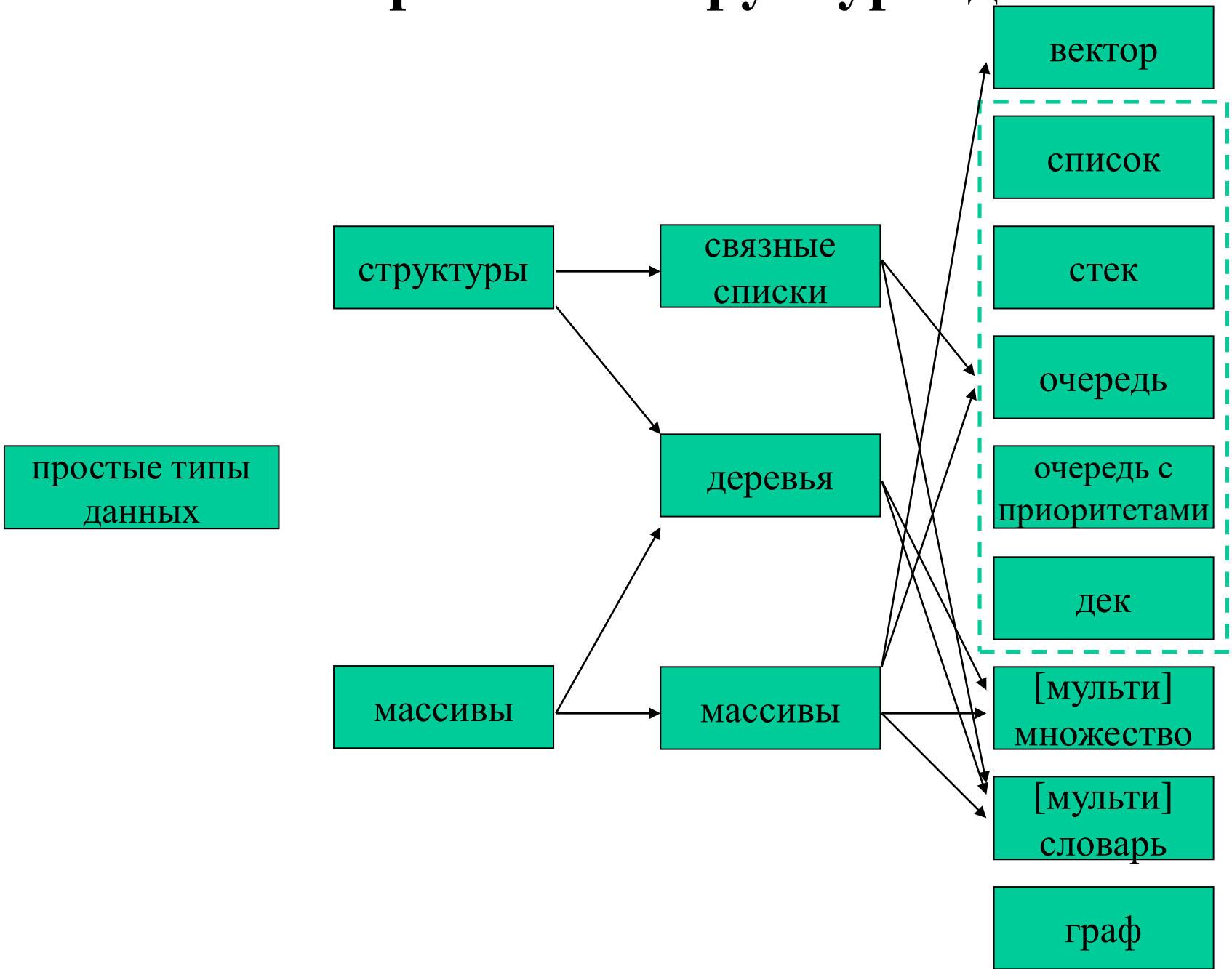
***Программа* – данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма (ГОСТ 19781-90).**

Программа – это описание на языке программирования действий, которые должна выполнить ЭВМ в соответствии с алгоритмом решения конкретной задачи или группы задач.

Абстрактные типы (структуры) данных

- Массив
- Список
- Стек
- Очередь
- Дек (двухсторонняя очередь)
- Очередь с приоритетами
- Множество
- Мультимножество
- Словарь (хеш-таблица, ассоциативный массив)
- Мультисловарь
- Строка (динамический массив знаков)
- Вектор (динамический массив)
- Дерево
- Граф

Абстрактные структуры данных



Анализ алгоритмов

Анализ – *математический* и *эмпирический*!

Данные:
- реальные;
- случайные;
- искаженные.

Для чего анализ:

- сравнение разных алгоритмов;
- прогнозирование оценки производительности алгоритма в новой среде;
- выбор значений параметров алгоритмов.

Задача аналитика: рабочие характеристики алгоритмов.

Задача программиста: использовать информацию о рабочих характеристиках при выборе алгоритмов для конкретных приложений.

Анализ алгоритмов

Необходимо анализировать алгоритмы (для сравнения, прогнозирования, настройки), следовательно необходимы критерии оценки и «методика» оценки.

Сложность алгоритмов:

- сложность реализации;
- пространственная (емкостная);
- временная.

Сложность алгоритмов:

- в лучшем (зачем?);
- в среднем (как это?);
- в худшем (а надо ли?).

Временная сложность алгоритма – это количество шагов (инструкций алгоритмов, для функций, программ – единиц времени), которые необходимо выполнить алгоритму для достижения запланированного результата.

Функциональную зависимость временной сложности от размера исходных (обрабатываемых) данных обозначают $T(N)$. Единицами измерения могут служить время, количество шагов, количество определенных операций (например, присваивания, сравнения).

Какой алгоритм лучше?

```
void SortBubble(Key A[], int Size)
  {
  int i, j;
  for(i = 0; i < Size; i++)
     for(j = 1; j < Size-i; j++)
        if(A[j-1]>A[j])
           Swap(A[j-1],A[j]);
  }
```

```
void SortInsert(Key A[], int Size)
  {
  int i, j;
  Key V;
  for (i = 1; i < Size; i++)
     {
     V = A[i];
     for(j = i; j > 0 && A[j-1] > V; j--)
        A[j] = A[j-1];
     A[j] = V;
     }
  }
```

Асимптотическая оценка

Для того, чтобы выделить существенные свойства алгоритма или программы используется асимптотическая оценка временной и/или пространственной сложности. В частности, используется O-символика.

Для описания временной сложности алгоритмов используется O-символика. Например, если время выполнения $T(N)$ некоторой программы имеет порядок $O(N^2)$ (читается «о-большое от N в квадрате» или просто «о от N в квадрате»), это означает, что существуют положительные константы c и N_0 такие, что для всех N, больших или равных N_0, выполняется неравенство $T(N) \leq cN^2$.

Определение. Запись $O(f_c(N))$ для временной сложности $T(N)$ означает, что $\exists$ c > 0, $N_0 > 0$ такие, что для всех $N \geq N_0$, выполняется неравенство $f_c(N) \geq T(N)$.

Правило сумм. Пусть две программы P_1 и P_2 имеют асимптотическую сложность $O(f(N))$ и $O(g(N))$ соответственно, тогда асимптотическая сложность последовательно выполняющихся программ будет $O(\max(f(N), g(N)))$.

Правило произведений. Пусть две программы P_1 и P_2 имеют асимптотическую сложность $O(f(N))$ и $O(g(N))$ соответственно, тогда асимптотическая сложность $T_1(N)*T_2(N)$ будет $O(f(N)*g(N))$.

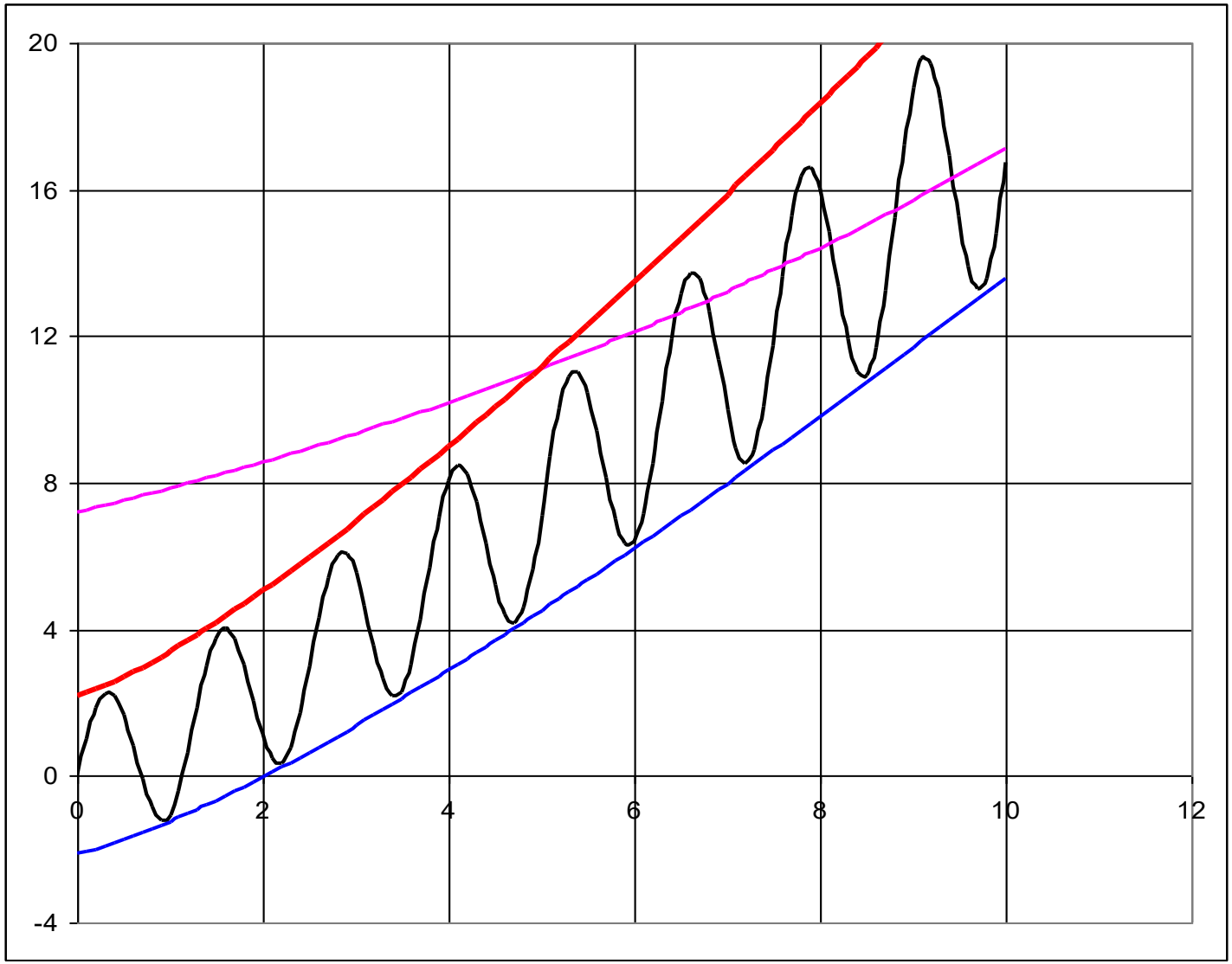
Правило констант.

Пример. Перемножение матриц – $O(N^3)$, алгоритм Штрассена $O(N^{2,81})$, самый $O(N^{2,376})$

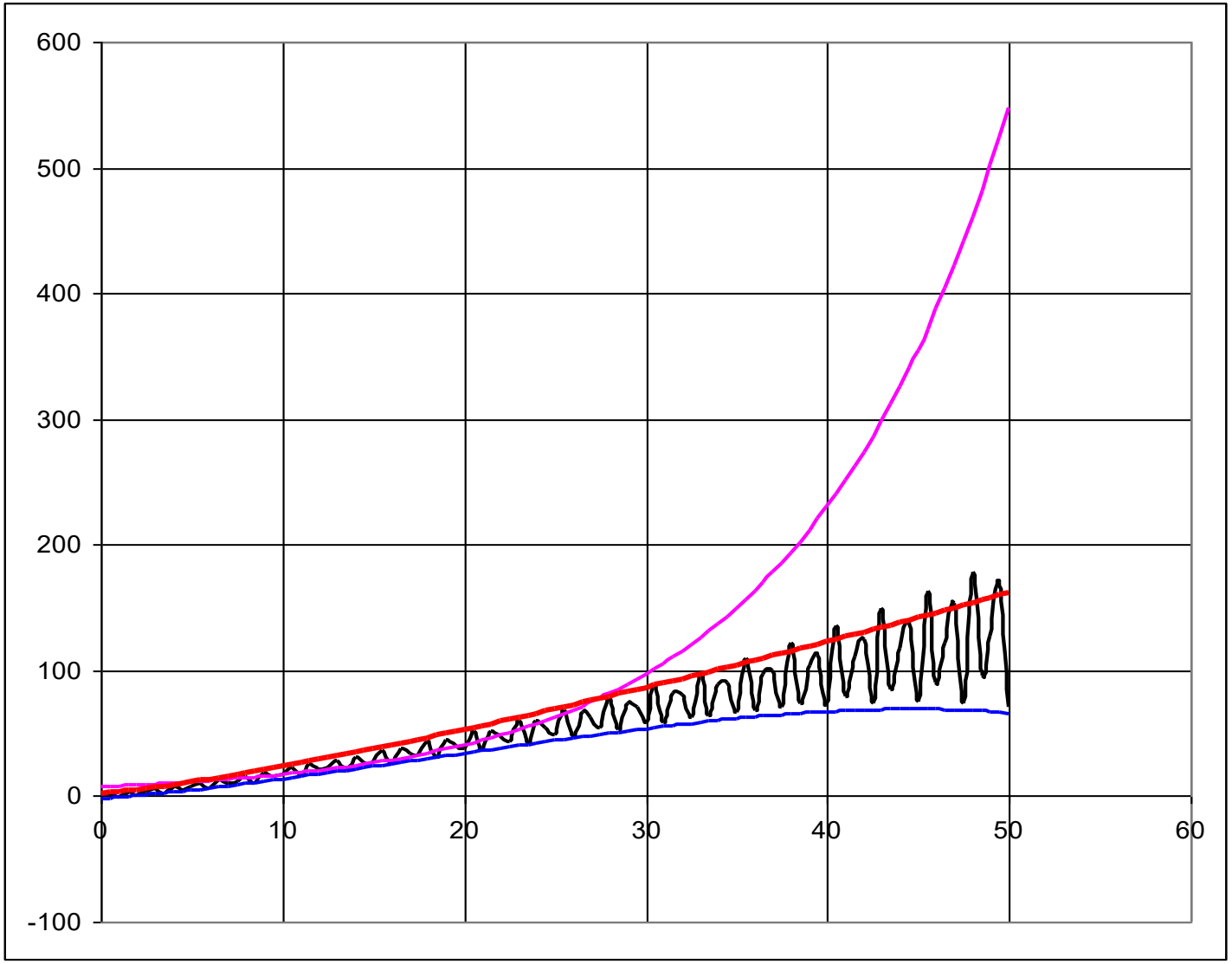
Асимптотические оценки

Обозначение	Описание	Определение
$f(n) \in O(g(n))$ или $f(n) = O(g(n))$	f ограничена сверху функцией g (с точностью до постоянного множителя) асимптотически	$\exists\, C > 0, n_0 : \forall\, n > n_0\ f(n) \le Cg(n)$
$f(n) \in \Omega(g(n))$ или $f(n) = \Omega(g(n))$	f ограничена снизу функцией g (с точностью до постоянного множителя) асимптотически	$\exists\, C > 0, n_0 : \forall\, n > n_0\ f(n) \ge Cg(n)$
$f(n) \in \Theta(g(n))$ или $f(n) = \Theta(g(n))$	f ограничена снизу и сверху функцией g асимптотически	$\exists\, C,\ C' > 0, n_0 :$ $\forall\, n > n_0\ Cg(n) \le f(n) \le C'g(n)$
$f(n) \in o(g(n))$ или $f(n) = o(g(n))$	g доминирует над f асимптотически	$\forall\, C > 0, \exists\, n_0 : \forall\, n > n_0\ f(n) < Cg(n)$
$f(n) \in \omega(g(n))$ или $f(n) = \omega(g(n))$	f доминирует над g асимптотически	$\forall\, C > 0, \exists\, n_0 : \forall\, n > n_0\ f(n) > Cg(n)$
$f(n) \sim g(n)$	f эквивалентна g асимптотически	$\lim\limits_{n\to\infty} \dfrac{g(n)}{f(n)} = 1$

Асимптотические оценки



Асимптотические оценки



Типовые зависимости

Функция	Название	Пример
1	постоянная, константная	обращение к элементу массива
$\log N$	логарифмическая	бинарный поиск
N	линейная	последовательный поиск
$N \log N$	квазилинейная	быстрая сортировка
N^2	квадратичная	алгоритм Дейкстры
N^3	кубическая	алгоритм Флойда
$2^N (a^N)$	экспоненциальная	NP-полные задачи
$N!$	факториальная	перестановки

Асимптотическая сложность в типовых рекурсивных алгоритмах

Рекуррентное соотношение	O($f(N)$)	Вывод	Описание
$T_N = T_{N-1} + N$, $N \geq 2$, $T_1 = 1$		$T_N = T_{N-1} + N = T_{N-2} + N + N - 1 = = \ldots = N(N + 1)/2$	На каждом шаге рекурсии (итерации) размер обрабатываемых данных уменьшается на 1.
$T_N = T_{N/2} + 1$, $N \geq 2$, $T_1 = 1$			-//- уменьшается вдвое
$T_N = T_{N/2} + N$, $N \geq 2$, $T_1 = 0$			-//- уменьшается вдвое, но предварительно обрабатывается каждый элемент.
$T_N = 2T_{N/2} + N$, $N \geq 2$, $T_1 = 0$			На каждом шаге и.(р.) обрабатываемые данные разделяются пополам, но до, в течение или после разбиения обр-ся каждый элемент
$T_N = 2T_{N/2} + 1$, $N \geq 2$, $T_1 = 1$			-//- данные разделяются пополам и выполняются некоторые другие действия, не зависящие от N.

Асимптотическая сложность в типовых рекурсивных алгоритмах

Рекуррентное соотношение	O($f(N)$)	Вывод	Описание
$T_N = T_{N-1} + N$, $N \geq 2$, $T_1 = 1$	$\approx N^2/2$	$T_N = T_{N-1} + N = T_{N-2} + N + N - 1 = = \ldots = N(N+1)/2$	На каждом шаге рекурсии (итерации) размер обрабатываемых данных уменьшается на 1.
$T_N = T_{N/2} + 1$, $N \geq 2$, $T_1 = 1$	$\approx \log N$		-//- уменьшается вдвое
$T_N = T_{N/2} + N$, $N \geq 2$, $T_1 = 0$	$\approx 2N$		-//- уменьшается вдвое, но предварительно обрабатывается каждый элемент.
$T_N = 2T_{N/2} + N$, $N \geq 2$, $T_1 = 0$	$\approx N \log N$		На каждом шаге и.(р.) обрабатываемые данные разделяются пополам, но до, в течение или после разбиения обр-ся каждый элемент
$T_N = 2T_{N/2} + 1$, $N \geq 2$, $T_1 = 1$	$\approx 2N$		-//- данные разделяются пополам и выполняются некоторые другие действия, не зависящие от N.

Асимптотическая сложность в типовых рекурсивных алгоритмах

•**Теорема.** Пусть имеется рекуррентное соотношение

$$T_N = aT_{N/b} + f(N),$$

где $a >= 1$, $b > 1$ – константы, $f(N)$ – функция, тогда

$$f(N) = \begin{cases} O(N^{\log_b a - \varepsilon}), mo\ T(N) = \Theta(N^{\log_b a}), \\ \Theta(N^{\log_b a}), mo\ T(N) = \Theta(N^{\log_b a} \lg N), \\ \Omega(N^{\log_b a + \varepsilon})\ u\ af(N/b) \le cf(N), c < 1, mo\ T(N) = \Theta(f(N)), \end{cases}$$

где $\varepsilon > 0$.