БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Алгоритмы и структуры данных

специальности:

1-31 03 03 «Прикладная математика»

1-31 03 07-01 «Прикладная информатика»

Лектор

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Соболевская Елена Павловна

доцент кафедры дискретной математики и алгоритмики,

кандидат физико-математических наук, доцент

Лауреат премии имени А.Н. Севченко в номинации «Образование»

за цикл пособий по дискретной математике, проектированию и анализу алгоритмов

http://fpmi.bsu.by/main.aspx?guid=30051

Практические занятия

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Буславский Александр Андреевич

старший преподаватель кафедры дискретной математики и алгоритмики,

Лауреат специального фонда Президента Республики Беларусь по социальной поддержке одарённых учащихся и студентов.

http://fpmi.bsu.by/main.aspx?guid=39321

Соболевская Елена Павловна

доцент кафедры дискретной математики и алгоритмики,

кандидат физико-математических наук, доцент

Лауреат премии имени А.Н. Севченко в номинации «Образование»

за цикл пособий по дискретной математике, проектированию и анализу алгоритмов

http://fpmi.bsu.by/main.aspx?guid=30051

Информационно-коммуникационные технологии:

1. Образовательный портал БГУ https://edufpmi.bsu.by
2. Образовательная платформа Insight Runner https://acm.bsu.by/
3. Группы в мессенджере Telegram, сервисы Google.

Разработчик Образовательной платформы Insight Runner

Соболь Сергей Александрович

инженер-программист ООО ЯндексБел,

старший преподаватель кафедры дискретной математики и алгоритмики (2014-2020 год),

магистр математики и информационных технологий,

серебряная медаль ACM ICPC 2013.

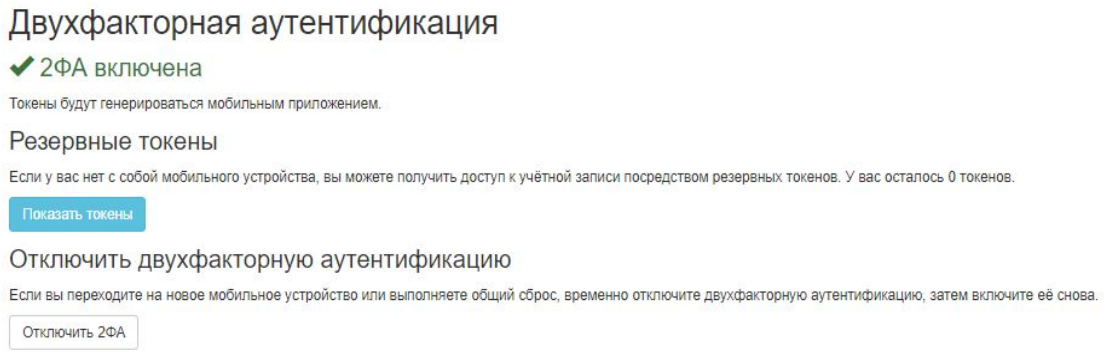
Insight Runner

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

https://acm.bsu.by/

логин номер вашего студенческого (семь цифр)
пароль (по умолчанию): 11111

После первого входа в iRunner необходимо сменить пароль и установить двухфакторную аутентификацию (для исключения противоправных действий в системе при несанкционированном доступе.)



Insight Runner **Wiki** (руководство по работе)

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

iRunner Wiki

Искать в iRunner Wiki

iR2

Навигация

- Система iRunner 2
- Система iRunner (старая)
- Заглавная страница
- Свежие правки
- Случайная статья
- Справка

Вики-инструменты

- Загрузить файл
- Служебные страницы

Заглавная страница

Заглавная | Обсуждение

Содержание скрыть

Система iRunner [править]

- Руководство по работе с системой iRunner.

Алгоритмы и структуры данных / Теория алгоритмов

Образовательный портал БГУ [править]

- Ссылка на курс АиСД (ПИ, ПМ) на портале БГУ
- Ссылка на курс ТА (ИНФ) на портале БГУ

Учебные программы [править]

- Теория алгоритмов. Учебная программа (Информатика)

Работа в системе — iRunner Wiki

Искать в iRunner Wiki

Работа в системе

Статья | Обсуждение

Содержание скрыть

Общий принцип [править]

В систему посылаются только файлы с исходным кодом, а сама посылаемая программа должна состоять только из одного файла: `*.dpr`, `*.cpp`, Нельзя отправить в систему скомпилированный exe-файл, файл проекта Visual Studio и т. п.

В решениях запрещается:

- осуществлять доступ к сети;
- выполнять любые операции ввода/вывода, кроме открывания, закрывания, чтения и записи стандартных потоков stdin, stdout, stderr и файлов прописанными в условии задачи;
- сознательно «ломать» тестирующую систему;
- выполнять другие программы и порождать новые процессы;
- изменять права доступа к файловой системе;
- работать с поддиректориями;
- создавать и манипулировать ресурсами GUI (окна, диалоговые сообщения и т. д.);
- работать со внешними устройствами (звук, принтер и т. д.);
- выполнять прочие действия, призванные нарушить ход учебного процесса.

Решения выполняются в специальном окружении, обеспечивающем безопасный запуск, и попытка выполнить какие-либо из указанных действий : всего, получением вердикта «Ошибка во время выполнения».

Ввод и вывод данных [править]

Insight Runner **Wiki**

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

acm.bsu.by

iRunner Wiki

Искать в iRunner Wiki

Система iRunner править

- Руководство по работе с системой iRunner.

Алгоритмы и структуры данных / Теория алгоритмов править

Образовательный портал БГУ править

- Ссылка на курс АиСД (ПИ, ПМ) на портале БГУ
- Ссылка на курс ТА (ИНФ) на портале БГУ

Учебные программы править

- Теория алгоритмов. Учебная программа (Информатика)
- Алгоритмы и структуры данных. Типовая учебная программа (ПИ,ПМ, ЭК, АМ, КБ)
- Алгоритмы и структуры данных. Учебная программа (Прикладная информатика)
- Алгоритмы и структуры данных. Учебная программа (Прикладная математика)
- Алгоритмы и структуры данных. Учебная программа (Экономическая кибернетика, Актуарная математика, Компьютерная безопасность)

Задачи iRunner править

- Описание тем задач.
- Система оценок по ТА (подгруппа Иржавского).

Теоретический материал (учебные пособия, статьи) править

- Учебное пособие «Алгоритмы и структуры данных» (В. М. Котов, Е. П. Соболевская, А. А. Толстиков).
- Учебное пособие «Теория алгоритмов» (П. А. Иржавский, В. М. Котов, А. Ю. Лобанов, Ю. Л. Орлович, Е. П. Соболевская).
- Материалы по структурам данных (И. С. Метельский).
- Сборник задач по теории алгоритмов. Бинарные поисковые деревья. Алгоритмы на графах. 2017 год.» (В. М. Котов, Ю. Л. Орлович, Е. П. Соболевская, С. А. Соболь
- Сборник задач по теории алгоритмов. Структуры данных. 2020 год.» (С. А. Соболь, К. Ю. Вильчевский,В. М. Котов, Е. П. Соболевская).
- Бинарные поисковые деревья.
- Программная реализация бинарных поисковых деревьев.
- Терминология теории графов.
- Максимальное и наибольшее.

ФПМИ БГУ

Insight Runner (тесты для)

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Для самоконтроля усвоения теоретического материала в Insight Runner разработана **система тестов.**

Тесты разработаны для большинства разделов учебной дисциплины.

Ответы к тестам (на усмотрение преподавателя) могут быть открыты после прохождения теста всеми учащимися.

Так как тесты в iRunner генерируются автоматически, то это позволяет бороться со списыванием.

Итоговый тест включает в себя 10 тестовых вопросов (20 минут) и его результат учитывается в рейтинговой оценке за работу в семестре.

Insight Runner (тесты для

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

- 2-й курс 4-я группа ТА 2019–2020
- *В архиве*
- Задачи по курсу
- Сообщения
- Назначение задач
- Журнал
- Ведомость
- Все решения
- Мои решения
- Компиляторы
- **Тесты**
- Электронная очередь
- Настройки

ТА. Самоконтроль (heap, mst, классы гр.)

Удалить

Студент: Евгений **Ганкович** VA
Оценка: 10
Баллы: 10,0 из 10,0
Время начала: 2 апреля 2020 г. 18:39
Длительность: 0:14:34

Верно
Неверно, но выбрано
Верно, но не выбрано

Вопрос 1 — 2,0 из 2,0

Связный взвешенный граф задан матрицей смежности. Если элемент $[i, j]$ в матрице равен ∞, то ребра $\{i, j\}$ в графе не существует. Определите вес минимального остовного дерева.

$$A = \begin{pmatrix} 0 & \infty & 9 & 8 & \infty \\ \infty & 0 & 5 & 2 & 1 \\ 9 & 5 & 0 & 7 & 2 \\ 8 & 2 & 7 & 0 & 4 \\ \infty & 1 & 2 & 4 & 0 \end{pmatrix}$$

13

Вопрос 2 — 4,0 из 4,0

Какие операции выполняются за $\Theta(1)$ в min-heap?

- добавление элемента
- уменьшение элемента по ключу
- уменьшение элемента по ссылке
- определение наименьшего элемента
- удаление наименьшего элемента

Вопрос 3 — 4,0 из 4,0

Для графа, заданного матрицей смежности A, определить, какие из следующих утверждений верны.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

- граф связный
- граф содержит не менее двух компонент связности

ФПМИ БГУ

Insight Runner (общие задачи)

Для закрепления на практике теоретических знаний в Insight Runner разработаны **общие задачи** (их должны выполнить все студенты).

Общие задачи открываются в iRunner, как правило, после каждой лекции и нацелены на проработку базовых знаний по пройденному на лекции материалу.

Эти задачи достаточно простые, не предполагают разработки сложных алгоритмов решения, а готовят студентов к решению индивидуальных задач.

Преподаватель может установить крайний срок выполнения задания. Задания, выполненные с нарушением этого срока, в системе iRunner имеют особые пометки.

Insight Runner
(общие задачи)

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Тема 1. Бинарные поисковые деревья:
0.1 построение дерева
0.2 удаление вершин из дерева
0.3 проверка является ли бинарное дерево поисковым
Тема 2. Динамическое программирование
0.1 Оптимальное перемножение группы матриц (двухмерное ДП)
0.2 Единицы - число сочетаний из n по k(одномерное ДП, модульная арифметика)
0.3. Единицы (большие ограничения, только для желающих)
20. Палиндром (двухмерное ДП, строки)
69. Кувшинки (простейшее одномерное ДП)
Тема 3. Структуры данных
0.1. Бинарный поиск (массив, уметь реализовать BinarySearch, LowerBound, UpperBound)
0.2 Задача о сумме (реализация структур для интервальных запросов - сумма на отрезке)
0.3. Бинарная куча (проверка на соответствие структуре)
0.4. Биномиальная куча (понимание структуры)
0.5. Хеш-таблица (разрешение коллизий метом открытой адресации)
Тема 4. Алгоритмы на графах
0.1 Строительство дорог (DSU + алг. на графах)
0.2 Разрушение дорог (DSU + алг. на графах)
0.3 Разрушение дорог (большие ограничения, только для желающих)
0.4 Матрица смежности
0.5. Канонический вид (по списку дуг)
0.6. Список смежности
0.7 Канонический вид (по матрице смежности)
0.8 BFS (поиск в ширину)
0.9 DFS (поиск в глубину)
0.10 Кратчайший путь. Алгоритм Дейкстры
0.11 Максимальный поток в сети (простая версия)
0.12 Максимальный поток в сети (большие ограничения, только для желающих)

Insight Runner (общие задачи)

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Общие задачи

№	Студент	Деревья поиска				Рекуррентные соотношения до 02.03.2020				Структуры данных до 15.03.2020			К.р. №1 (ДП). до 12.03.2020		DSU			Сумма	Алгоритмы на графах									Задачи	Баллы
		0.0	0.1. Построит	0.2. Удалить	0.3. is-bst	0.1. Порядок	69. Кувшинки	6. Без разрывов	32. Черепаш	0.1. Бин. поиск	0.3. Куча?	0.4. Бином. куча	82.1. dividing-sequence	82.2. dividing-sequence	0.1. roads	0.2. roads_2	0.3. roads_3	0.2. sum	0.4. Матрица смежн.	0.6. Список смежн.	0.5. Канон. вид	0.7. Канон. вид	0.8. BFS	0.9. DFS	0.10. Dijkstra	0.11. maximal-flow	0.12. maximal-flow-hard		
1																												24	17,0
2																									36 / 47			19	17,0
3																												26	17,0
4																												26	17,0
5																									46 / 47			23	17,0
6																												26	17,0
7																		21 / 24										21	17,0
8																		16 / 24										7	4,0
9																												20	17,0
10																												17	10,5
11																												22	17,0
12																												16	10,5
13																												26	17,0
14					51 / 55											37 / 41	27 / 36	16 / 24										20	10,5
15																												20	17,0
16								23 / 28																				5	2,0
17																												26	17,0
18																												21	13,0
19					51 / 55																							23	17,0
20																												24	17,0
21																												22	17,0
22																												24	17,0
23																												24	17,0
24																												24	17,0
25																												26	17,0
26																												26	17,0
27																												22	17,0
28																												24	17,0

Insight Runner 2 #9da975e

О системе Язык / Language Обратная связь

ФПМИ БГУ

Insight Runner (индивидуальные

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

В рамках учебной дисциплины студентами также выполняются **индивидуальные задачи.**

Число индивидуальных задач - по каждой теме не менее одной.

Индивидуальная задача предполагает разработку эффективного алгоритма решения задачи с последующей реализацией его в Insight Runner на любом языке программирования.

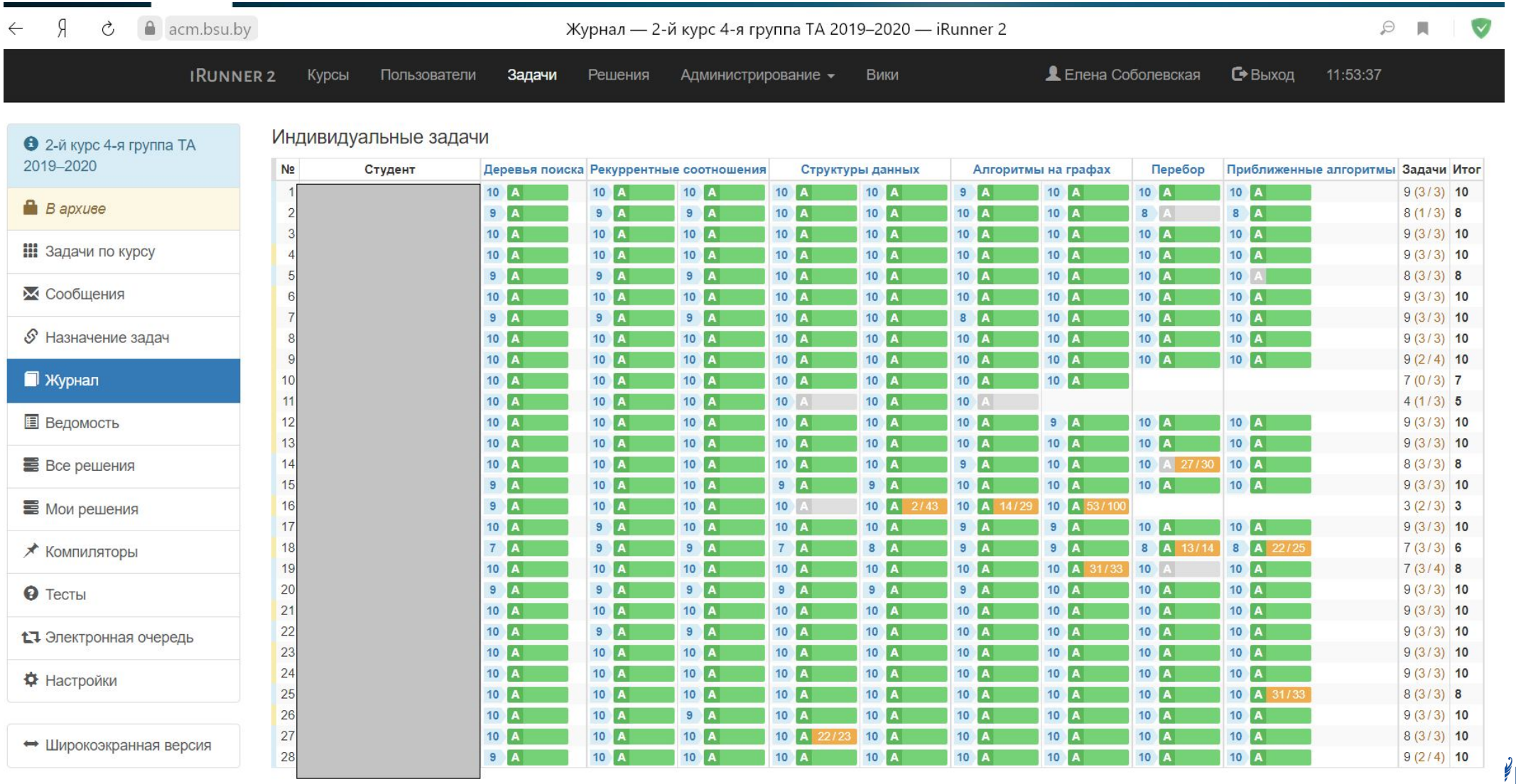
С 2016 г. в учебных курсах ограничивается до пяти число решений, которые можно отправить по каждой из назначенных задач в течение суток. На протяжении любого 24-часового отрезка времени разрешается отправить не более чем пять решений по каждой задаче. Ограничение не привязано к наступлению новых суток в 00:00. Решения с вердиктом «Ошибка компиляции» при подсчёте оставшихся попыток игнорируются.

Отметим, что в течение многих лет для всех студентов, которые работали в старой версии системы Insight Runner, действовало ограничение в двадцать попыток по задаче в семестр. Это ограничение было нельзя увеличить индивидуально. Политика ограничения числа посылок в день вместо ограничения общего числа посылок является более гибкой и применяется во многих системах, например на платформе Kaggle.

Призываем вас более качественно тестировать свои решения перед отправкой!!!

Insight Runner (индивидуальные

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ



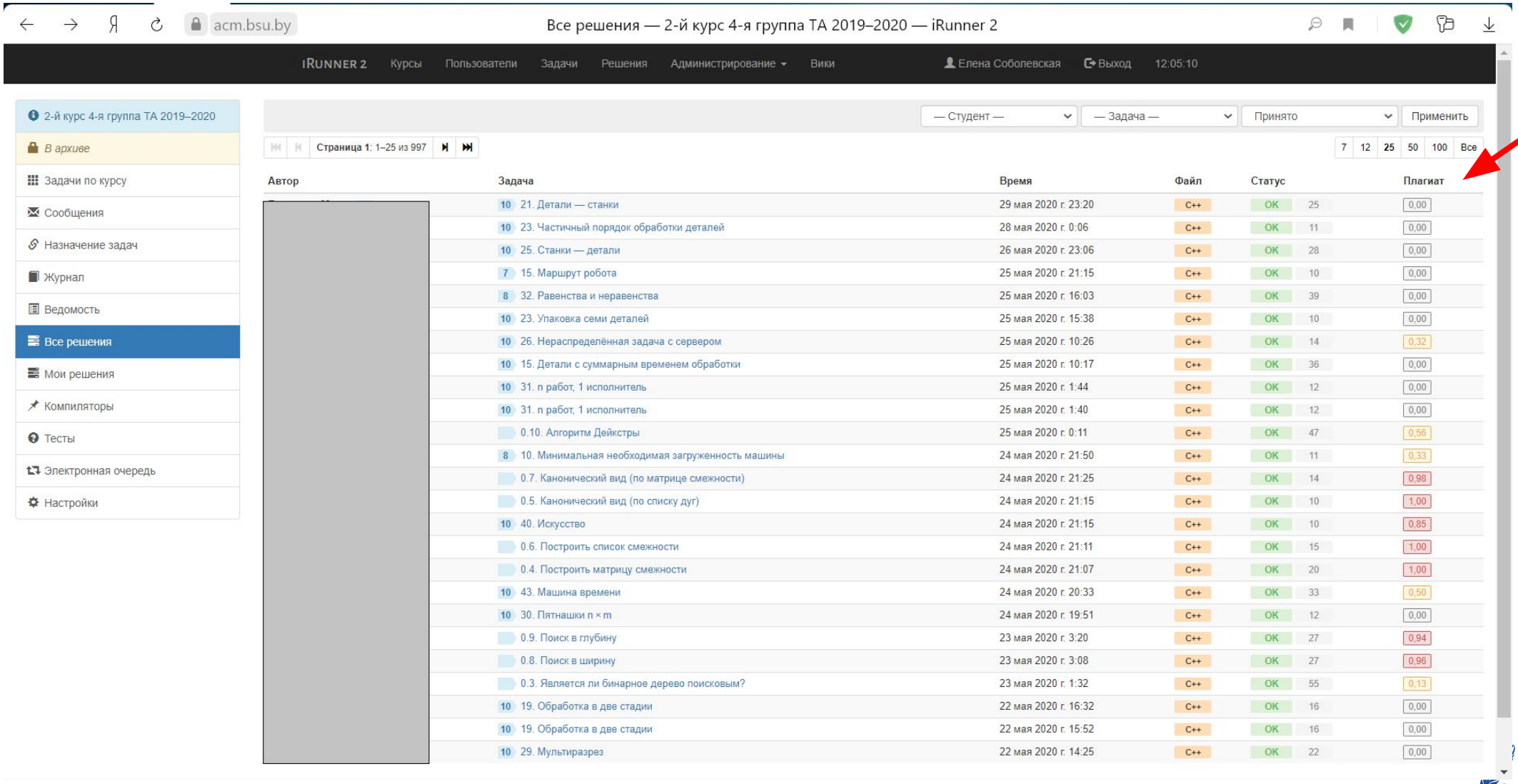
№	Студент	Деревья поиска	Рекуррентные соотношения		Структуры данных		Алгоритмы на графах		Перебор	Приближенные алгоритмы	Задачи	Итог
1		10 A	10 A	10 A	10 A	10 A	9 A	10 A	10 A	10 A	9 (3 / 3)	10
2		9 A	9 A	9 A	10 A	10 A	10 A	10 A	8 A	8 A	8 (1 / 3)	8
3		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
4		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
5		9 A	9 A	9 A	10 A	10 A	10 A	10 A	10 A	10 A	8 (3 / 3)	8
6		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
7		9 A	9 A	9 A	10 A	10 A	8 A	10 A	10 A	10 A	9 (3 / 3)	10
8		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
9		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (2 / 4)	10
10		10 A	10 A	10 A	10 A	10 A	10 A	10 A			7 (0 / 3)	7
11		10 A	10 A	10 A	10 A	10 A	10 A				4 (1 / 3)	5
12		10 A	10 A	10 A	10 A	10 A	10 A	9 A	10 A	10 A	9 (3 / 3)	10
13		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
14		10 A	10 A	10 A	10 A	10 A	9 A	10 A	10 A 27 / 30	10 A	8 (3 / 3)	8
15		9 A	10 A	10 A	9 A	9 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
16		9 A	10 A	10 A	10 A	10 A 2 / 43	10 A 14 / 29	10 A 53 / 100			3 (2 / 3)	3
17		10 A	9 A	10 A	10 A	10 A	9 A	9 A	10 A	10 A	9 (3 / 3)	10
18		7 A	9 A	9 A	7 A	8 A	9 A	9 A	8 A 13 / 14	8 A 22 / 25	7 (3 / 3)	6
19		10 A	10 A	10 A	10 A	10 A	10 A	10 A 31 / 33	10 A	10 A	7 (3 / 4)	8
20		9 A	9 A	9 A	9 A	9 A	9 A	10 A	10 A	10 A	9 (3 / 3)	10
21		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
22		10 A	9 A	9 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
23		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
24		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
25		10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A 31 / 33	8 (3 / 3)	8
26		10 A	10 A	9 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (3 / 3)	10
27		10 A	10 A	10 A	10 A 22 / 23	10 A	10 A	10 A	10 A	10 A	8 (3 / 3)	10
28		9 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	10 A	9 (2 / 4)	10

ФПМИ БГУ

Insight Runner

проверка на плагиат

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ



Insight Runner

проверка на плагиат

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

```
19
20          Scanner in = new Scanner(new File("input.txt"));
21          PrintWriter out = new PrintWriter("output.txt");
22          int n = in.nextInt();
23
24          if(n==3){
25              out.println(0);
26              out.flush();
27
28          }
29
30          else {
31              while (in.hasNext()) {
32                  points.add(new Point(in.nextInt(), in.nextInt()));
33              }
34              if(n==4){
35                  out.format("%.2f%n",
Math.min(points.get(0).distance(points.get(2)),points.get(1).distance(points.get(3))));
36                  out.flush();
37              }else {
38
39                  double[][] m = new double[n + 1][n];
40
41                  n--;
42                  for (int i = 1; i <= n; i++)
43                      m[i][i] = 0;
44
45                  for (int p = 2; p <= n; p++) {
46                      for (int i = 1; i <= n - p + 1; i++) {
47                          int j = i + p - 1;
48                          m[i][j] = Integer.MAX_VALUE;
49
50                          for (int k = i; k <= j - 1; k++) {
51                              double d = calcDiagonals(i - 1, k, j, points);
52                              double dMax = Math.max(Math.max(m[i][k], m[k + 1][j]), d);
53                              if (dMax < m[i][j]) {
```

```
19
20          int n = fin.nextInt();
21
22          if(n==3){
23              fout.println(0);
24              fout.flush();
25
26          }
27
28          else {
29              while (fin.hasNext()) {
30                  points.add(new Point(fin.nextInt(), fin.nextInt()));
31              }
32              if(n==4){
33                  fout.format("%.2f%n",
Math.min(points.get(0).distance(points.get(2)),points.get(1).distance(points.get(3))));
34                  fout.flush();
35              }else {
36
37                  double[][] memor = new double[n + 1][n];
38
39                  n--;
40                  for (int i = 1; i <= n; i++)
41                      memor[i][i] = 0;
42
43                  for (int p = 2; p <= n; p++) {
44                      for (int i = 1; i <= n - p + 1; i++) {
45                          int j = i + p - 1;
46                          memor[i][j] = Integer.MAX_VALUE;
47
48                          for (int k = i; k <= j - 1; k++) {
49                              double d = countDiagonals(i - 1, k, j, points);
50                              double dMax = Math.max(Math.max(memor[i][k], memor[k +
1][j]), d);
51                              if (dMax <= memor[i][j])
```

ФПМИ БГУ

Литература

1. В.М. Котов, Е. П. Соболевская, А. А. Толстиков. «Алгоритмы и структуры данных»: учеб. пособие Минск: БГУ, 2011г. – 267 с. – (Классическое университетское издание).
2. Теория алгоритмов: учеб. пособие / П.А. Иржавский, В.М. Котов, А.Ю. Лобанов, Ю.Л. Орлович, Е.П. Соболевская – Минск : БГУ, 2013. – 159 с.
3. Сборник задач по теории алгоритмов : учеб.-метод. пособие / В.М. Котов, Ю.Л. Орлович, Е.П. Соболевская, С.А. Соболь – Минск : БГУ, 2017.- 183с
4. Соболь С.А., Вильчевский К.Ю., Котов В.М., Соболевская Е.П. Сборник задач по теории алгоритмов. Структуры данных: – Минск : БГУ, 2020. – 159 с. (с грифом УМО по естественнонаучному образованию).
5. Алгоритмы: построение и анализ / Т. Кормен [и др.] – М.: Вильямс, 2005. – 1296 с.

Insight Runner
(индивидуальное задание № 1)

Для выполнения первой индивидуальной задачи, необходимо обладать навыками работы **с бинарными поисковыми деревьями**.

Частично вы уже получили эти навыки в рамках дисциплин по программированию, поэтому сейчас систематизируем их.

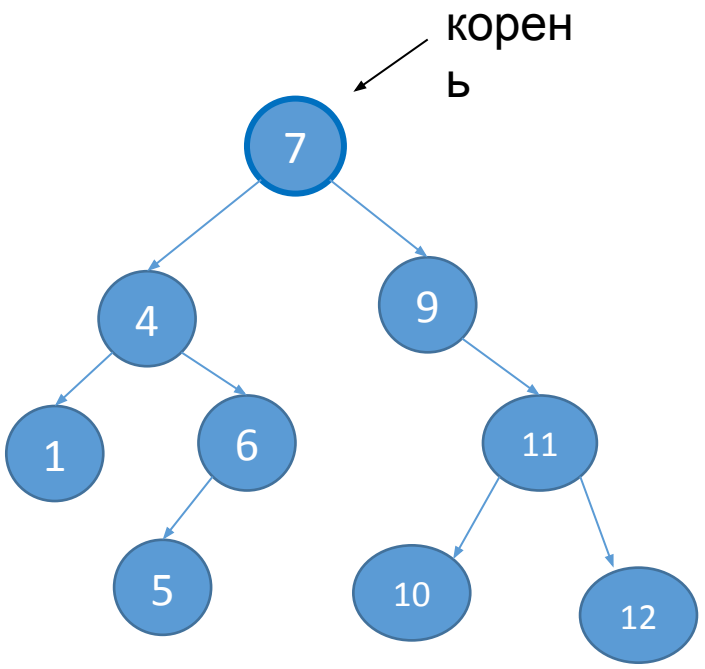
Словарные операции

- поиск элемента с заданным ключом x
- добавление нового элемента с заданным ключом x
- удаление элемента с заданным ключом x

Бинарное поисковое дерево

Корневое дерево

1. Ориентированный граф, в котором существует ровно одна вершина без входящих дуг (корень).
2. В каждую вершину, за исключением корня, входит ровно одна дуга.
3. Из корня дерева существует единственный путь в любую вершину.



Бинарное

4. Каждая вершина содержит не более 2-х сыновей.

Поисковое

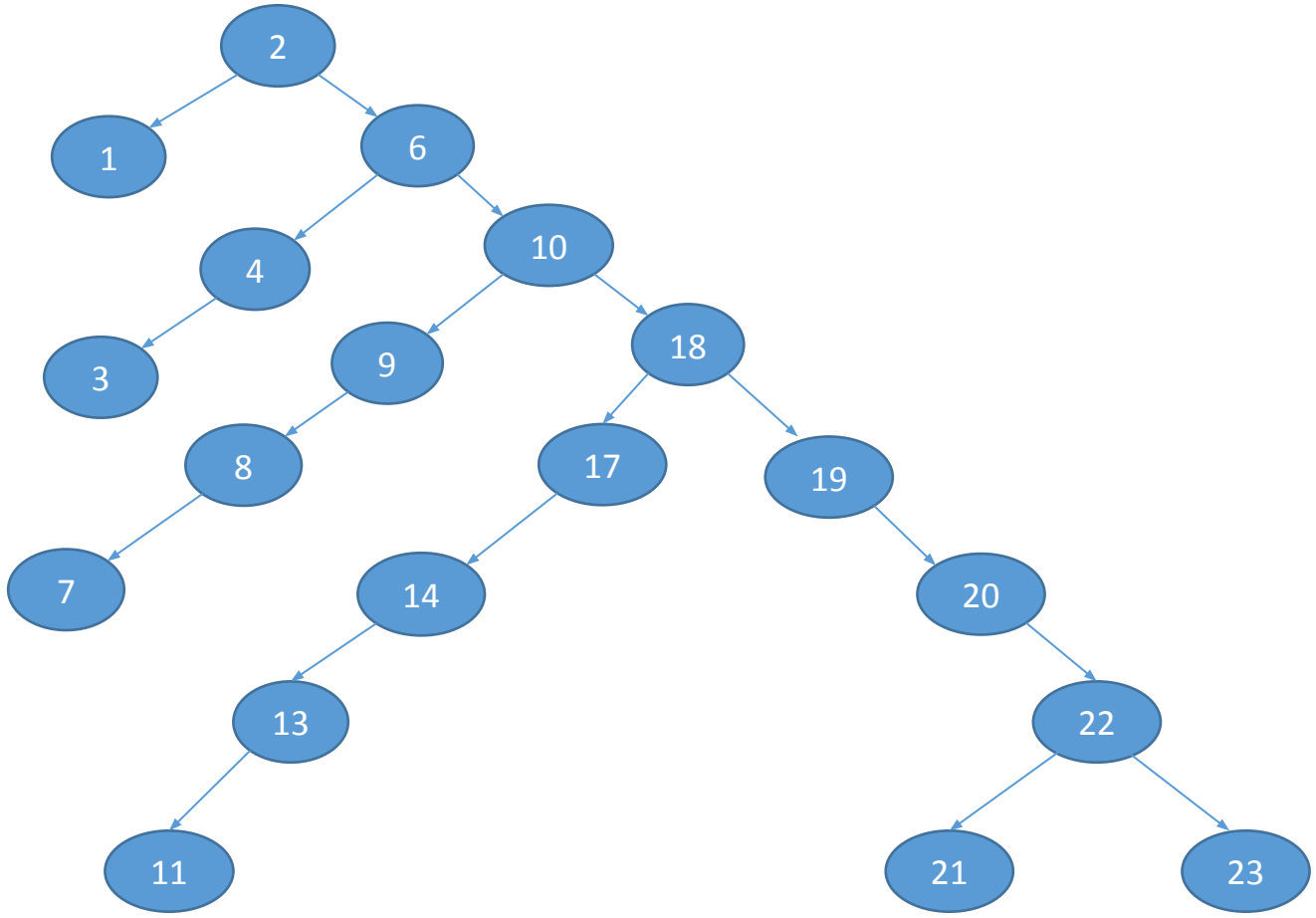
5. Каждой вершине поставлено в соответствие некоторое целое число - *ключ*. Для каждой вершины ***v*** все ключи в её левом поддереве строго меньше ключа вершины ***v***, а в правом – строго больше.

n – число вершин

m=n-1 – число дуг

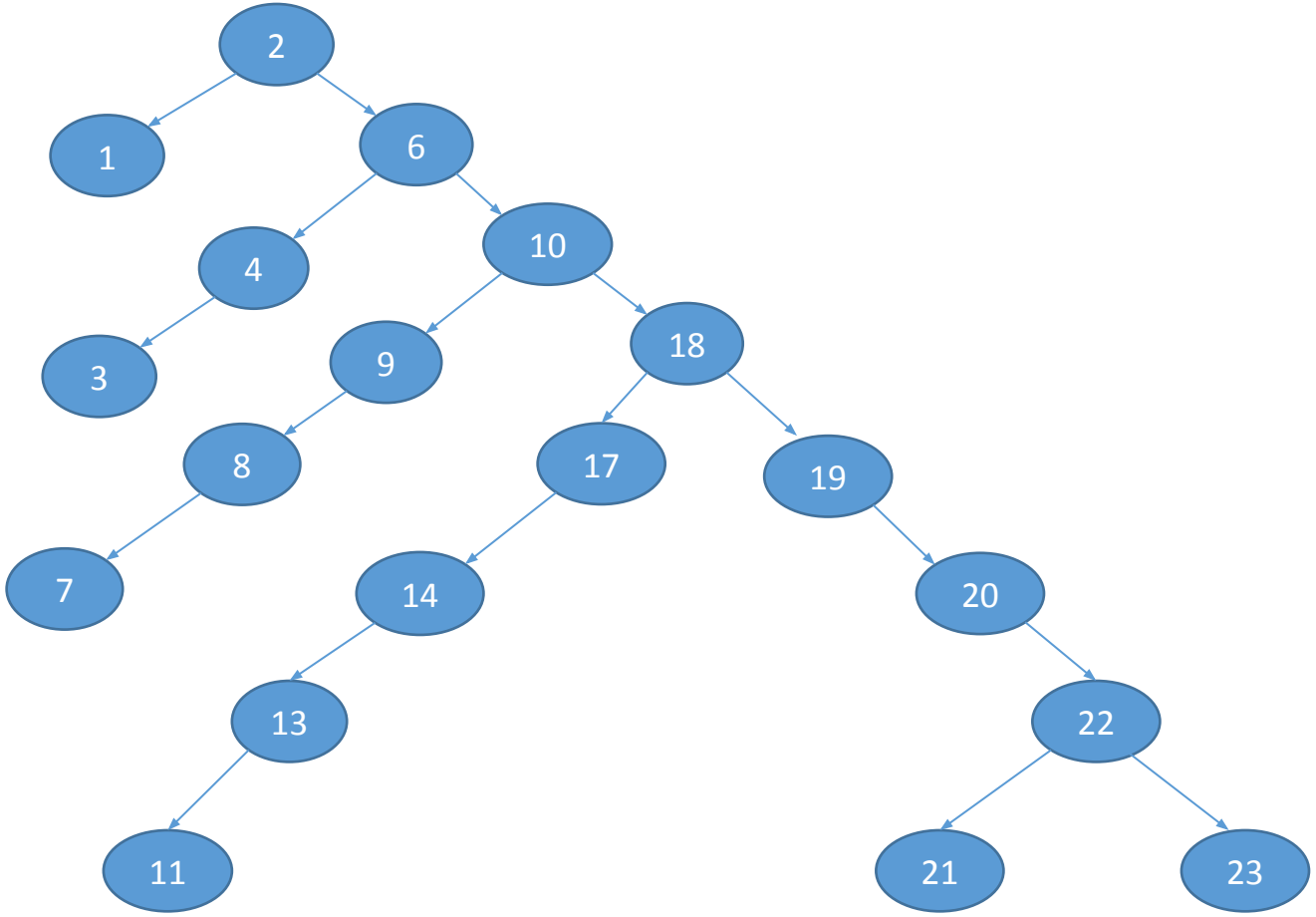
Построить бинарное поисковое дерево для последовательности элементов последовательным добавлением нового элемента:

2, 1, 6, 4, 3, 10, 9, 8, 7, 18, 17, 14, 13, 11, 19, 20, 22, 23, 21



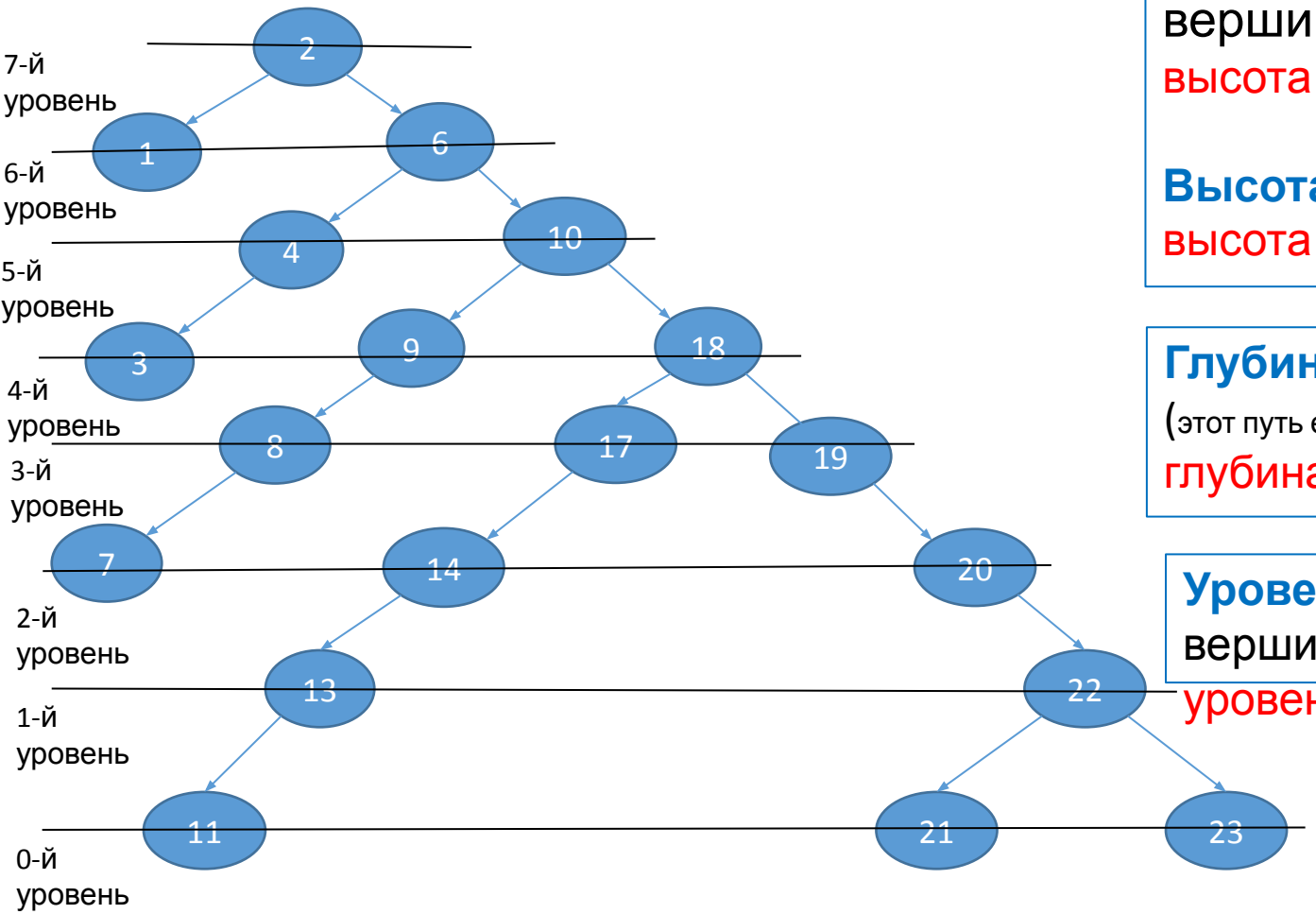
??? **6**

добавить



так как мы договорились (см. определение) , что в дереве все ключи различны, то одинаковые элементы при добавлении будем игнорировать

Высота, глубина, уровень



Высота вершины: длина наибольшего пути от вершины до одного из её потомков.

высота (10) = **5**

Высота дерева: высота корня.

высота (дерева) = **7**

Глубина вершины: длина пути из корня в вершину (этот путь единственный).

глубина (10) = **2**

Уровень вершины: высота дерева минус глубина вершины

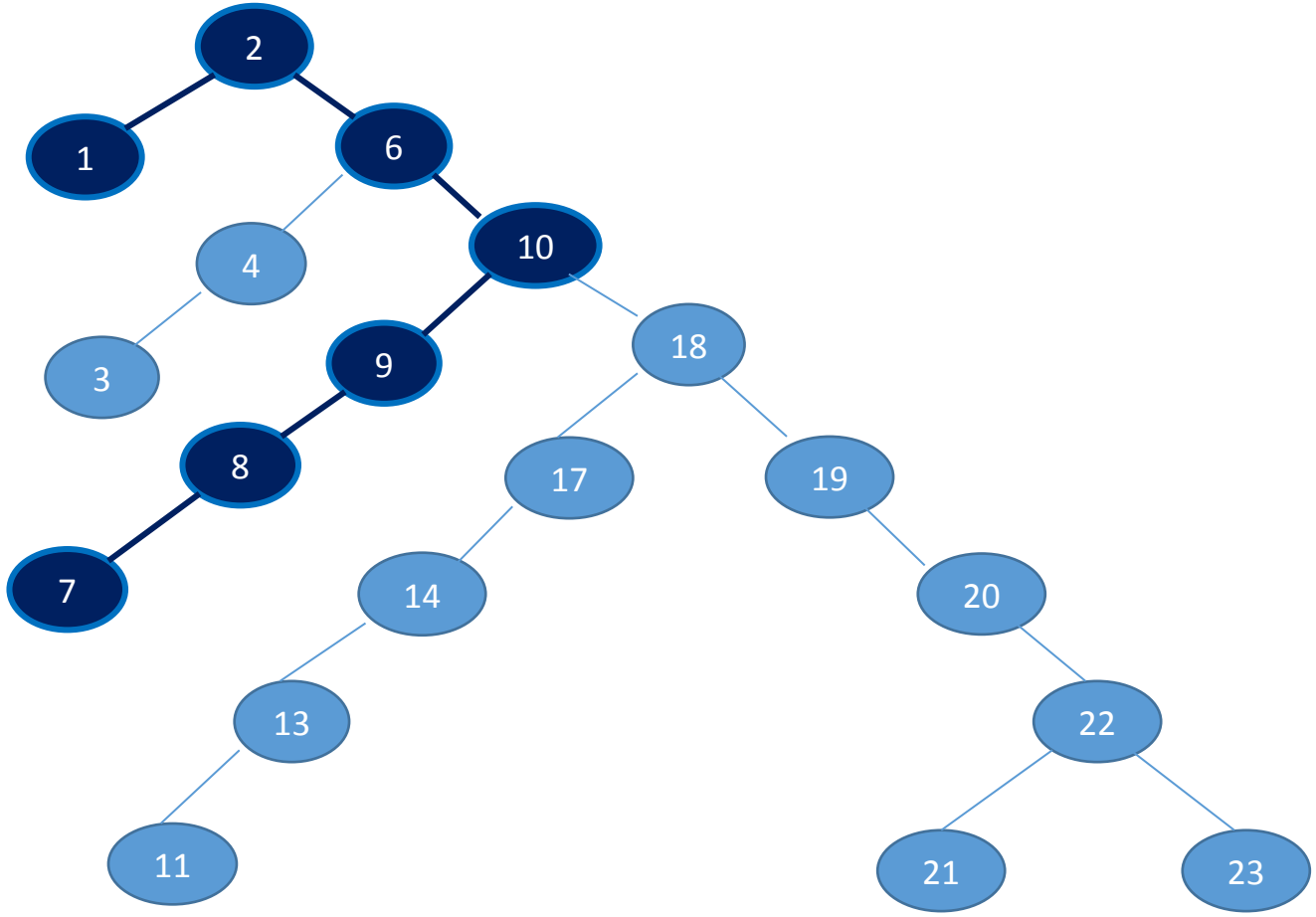
уровень (10) = высота (дерева)- глубина (10) =7 – 2 = **5**

ФПМИ БГУ

Путь, полупуть

Для полупути снимается ограничение на направление дуг.

Пример полупути, соединяющего вершины 1 и 7: **1 - 2 - 6 - 10 - 9 - 8 - 7**



Центральная и средняя вершины полупути

Центральная вершина полупути -
такая вершина, что количество вершин в полупути до неё равно количеству вершин после неё.

Средняя по значению (медиана) вершина полупути -такая вершина, что у половины из оставшихся вершин этого полупути ключ меньше, а у половины – больше.

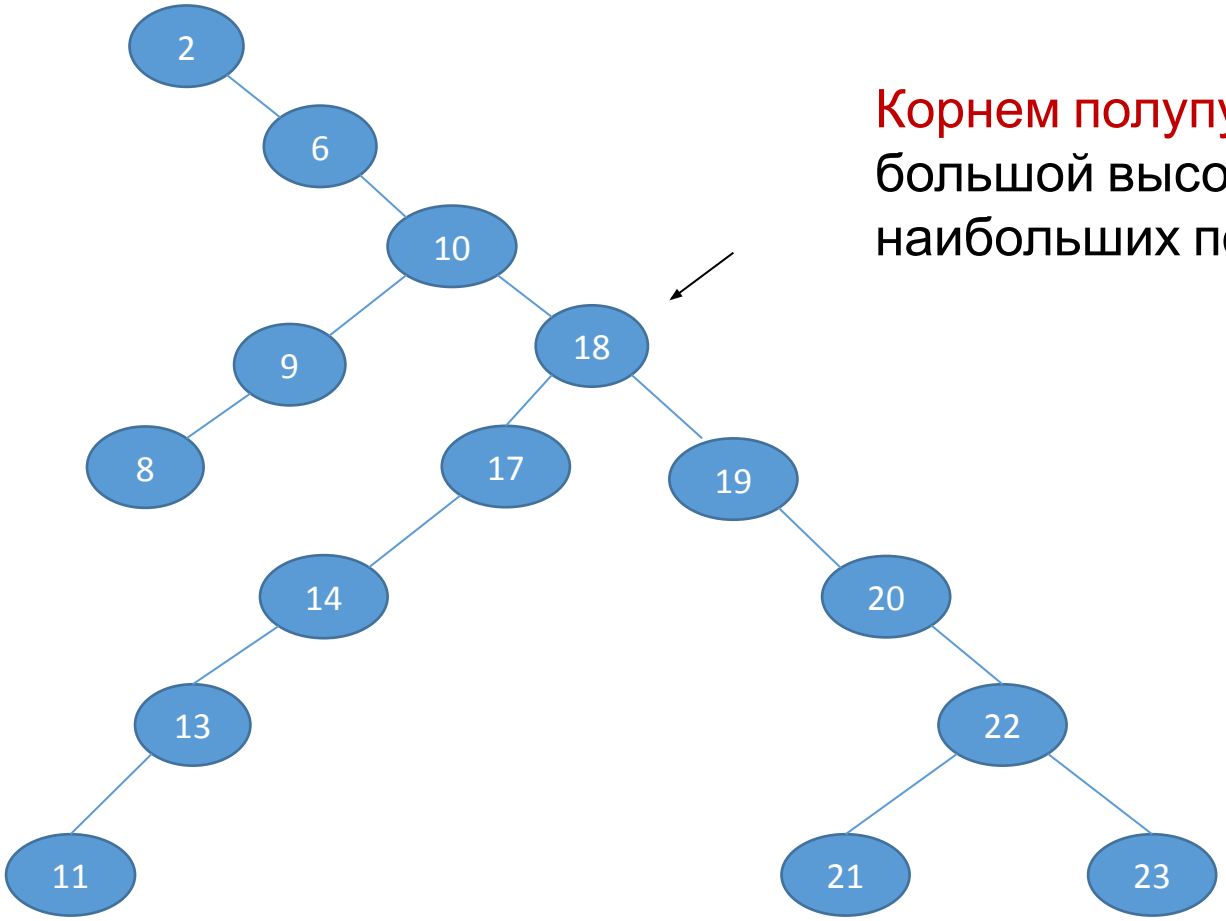
?

Что делать, если число вершин, среди которых надо найти центральную или среднюю ЧЁТНО?

↓

центральной и средней вершины
НЕ СУЩЕСТВУЕТ

Наибольшим полупутём в дереве будем называть полупуть наибольшей длины (напомним, что длина пути измеряется в рёбрах, а не вершинах).



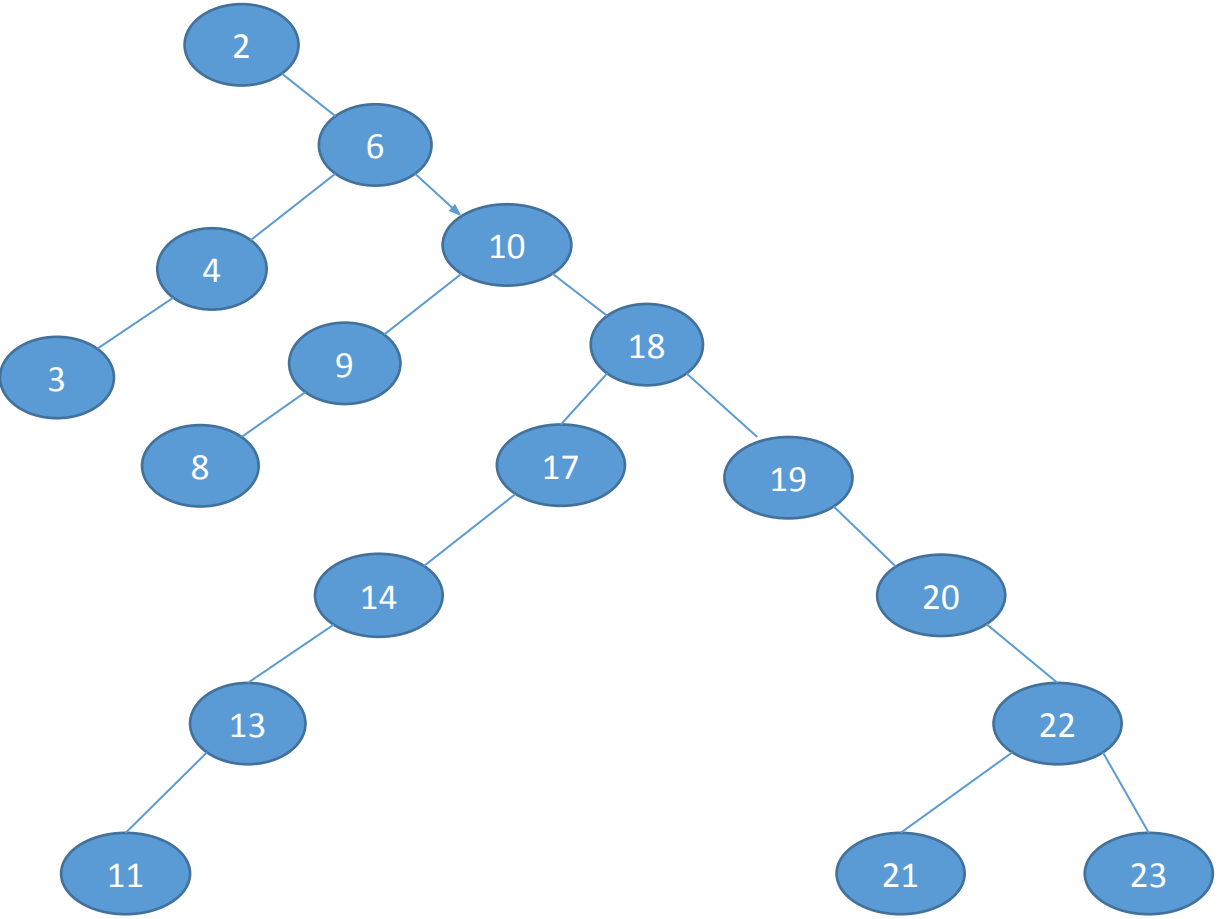
Корнем полупути назовём вершину полупути с самой большой высотой (на рисунке выделены два наибольших полупути и у них общий корень 18).

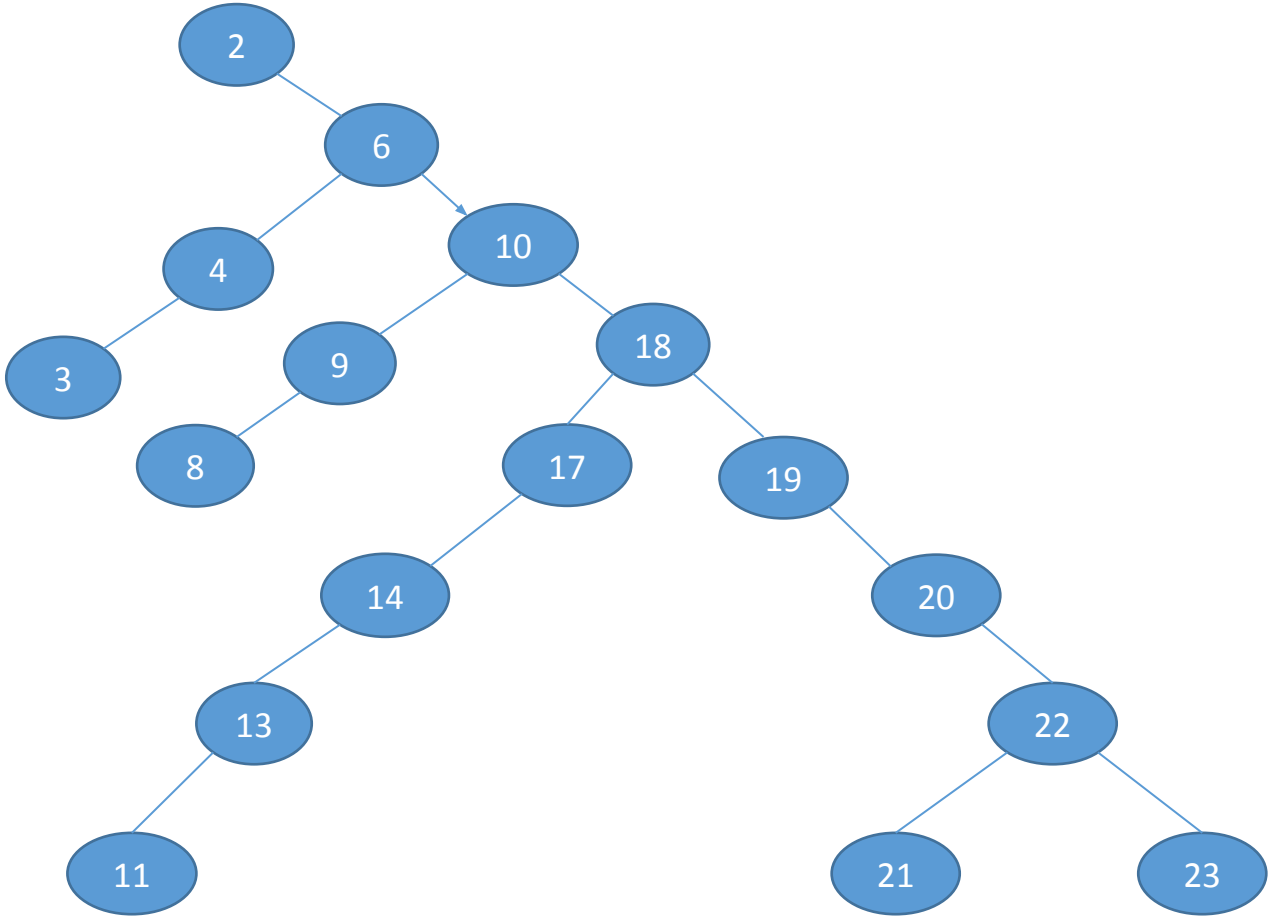
Пример.

1) Длина наибольшего полупути? =8

2) Какие вершины являются корнями полупутей наибольшей длины? =18 и 6

3) Сколько попарно различных полупутей наибольшей длины проходит через вершину **18**? =5





Обходы

прямой (левый, правый) - **PreOrderTraversal** (v)

обратный (левый, правый) - **PostOrderTraversal** (v)

внутренний (левый, правый) - **InOrderTraversal** (v)

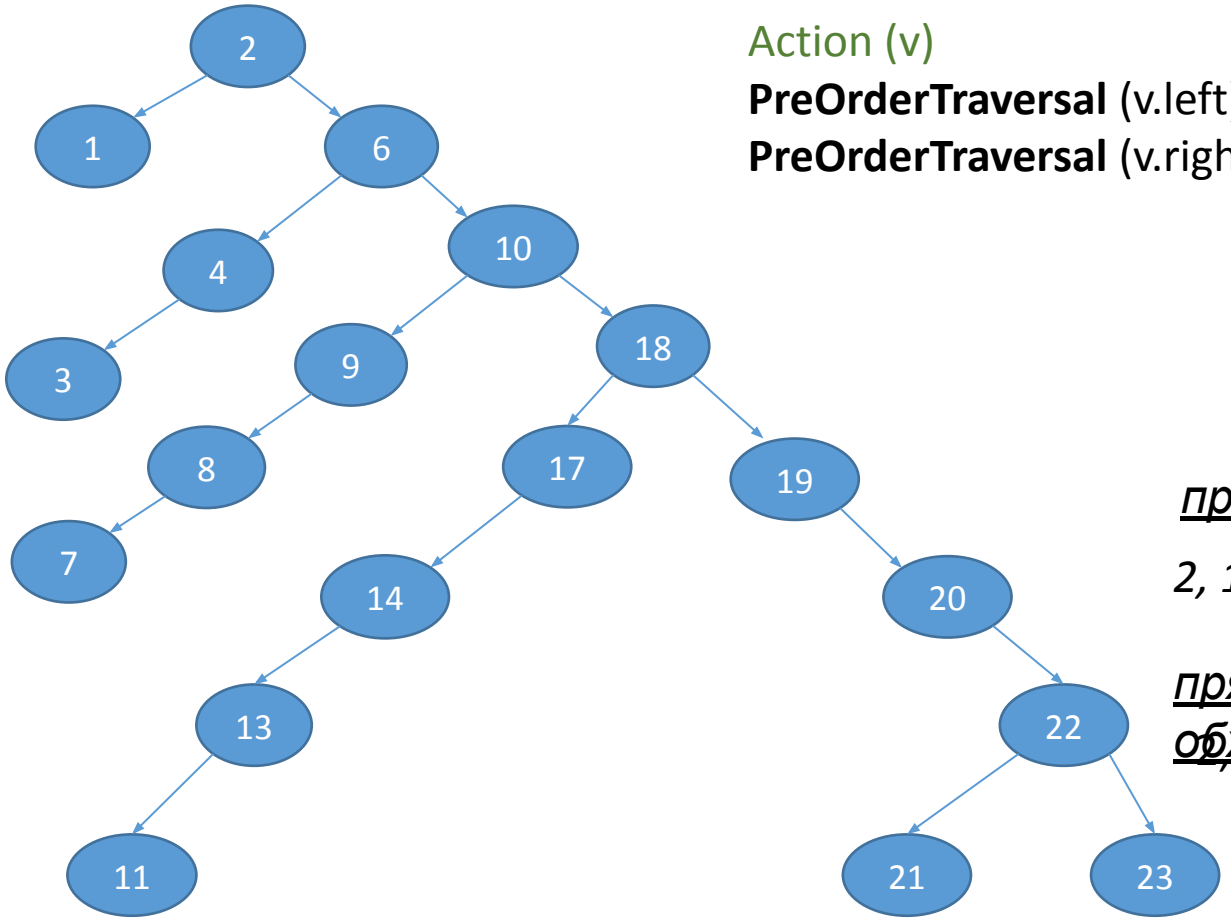
Время выполнения обхода: пропорционально числу вершин в дереве (=**n)**

прямой левый обход

```
def PreOrderTraversal (v):
    if v is not None:
        Action (v)
        PreOrderTraversal (v.left)
        PreOrderTraversal (v.right)
```

прямой правый обход

```
def PreOrderTraversal (v):
    if v is not None:
        Action (v)
        PreOrderTraversal (v.right)
        PreOrderTraversal (v.left)
```



прямой левый обход

2, 1, 6, 4, 3, 10, 9, 8, 7, 18, 17, 14, 13, 11, 19, 20, 22, 21, 23

прямой правый обход

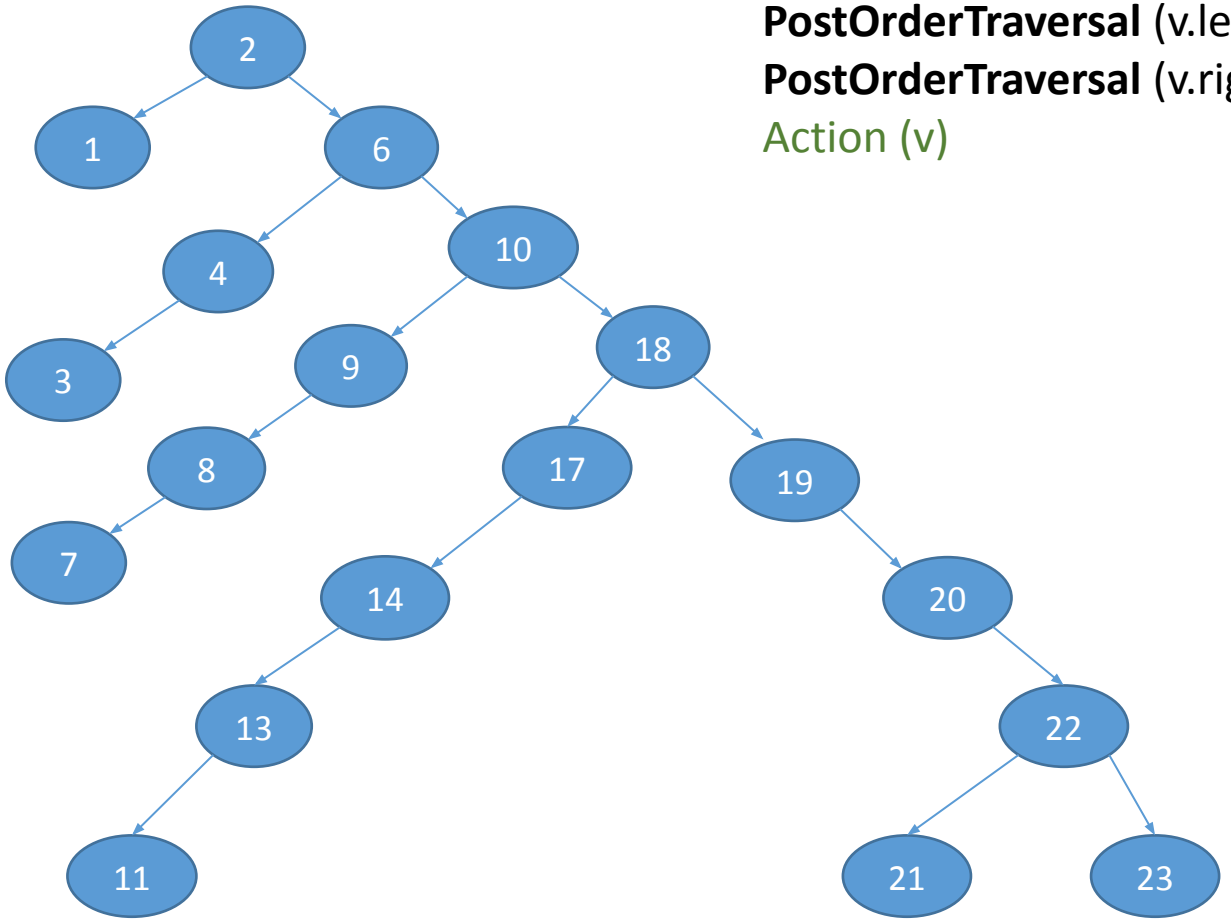
2, 6, 10, 18, 19, 20, 22, 23, 21, 17, 14, 13, 11, 9, 8, 7, 4, 3, 1

обратный левый обход

```
def PostOrderTraversal (v):
    if v is not None:
        PostOrderTraversal (v.left)
        PostOrderTraversal (v.right)
        Action (v)
```

обратный правый обход

```
def PostOrderTraversal (v):
    if v is not None:
        PostOrderTraversal (v.right)
        PostOrderTraversal (v.left)
        Action (v)
```



обратный левый обход

1 ,3, 4, 7, 8, 9, 11, 13, 14, 17, 21, 23, 22, 20, 19, 18, 10, 6, 2

обратный правый обход

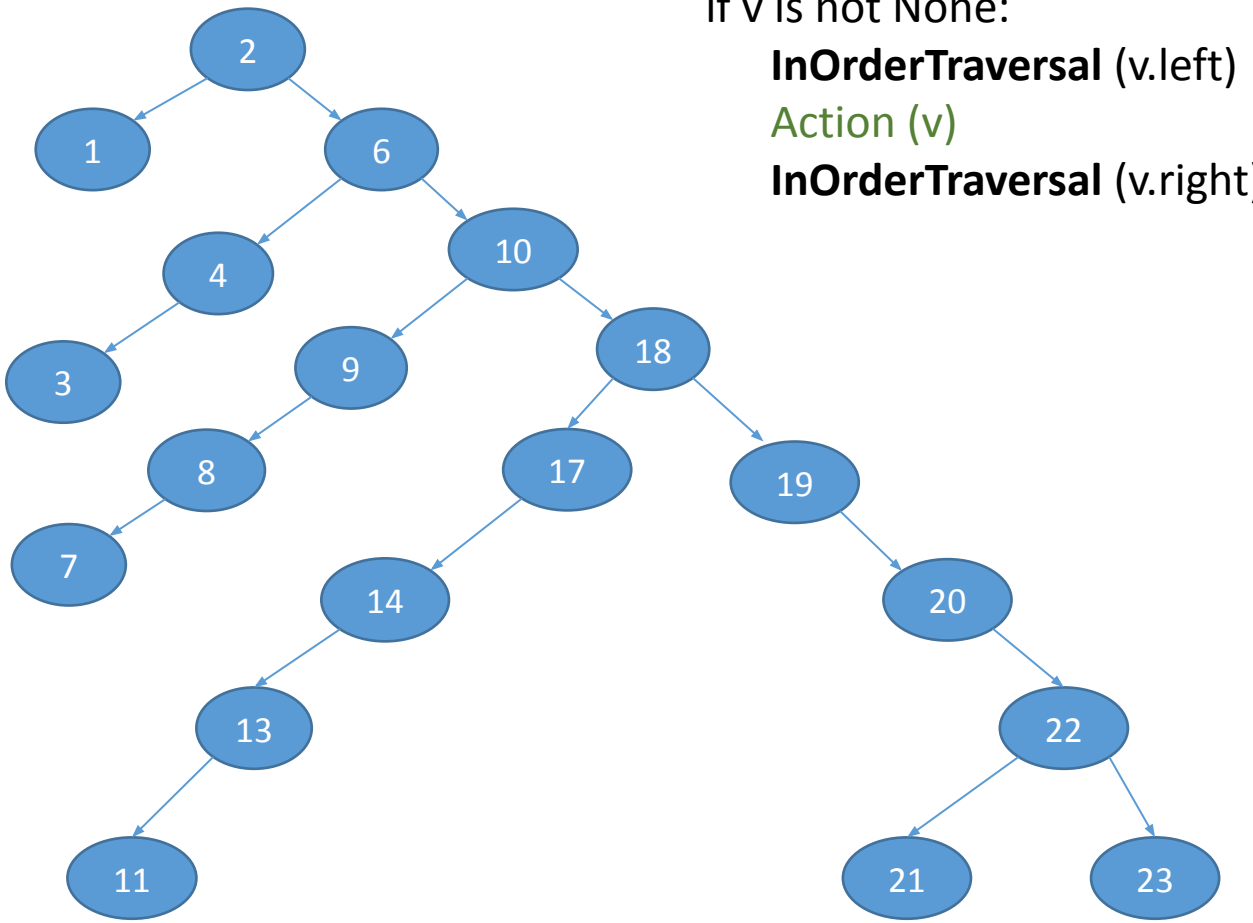
23, 21, 22, 20, 19, 11, 13, 14, 17, 18, 7, 8, 9, 10, 3, 4, 6, 1, 2

внутренний левый обход

```
def InOrderTraversal (v):
    if v is not None:
        InOrderTraversal (v.left)
        Action (v)
        InOrderTraversal (v.right)
```

внутренний правый обход

```
def InOrderTraversal (v):
    if v is not None:
        InOrderTraversal (v.right)
        Action (v)
        InOrderTraversal (v.left)
```



внутренний левый обход

(ключи отсортированы по возрастанию)

1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13, 14, 17, 18, 19, 20, 21, 22, 23

внутренний правый обход

(ключи отсортированы по убыванию)

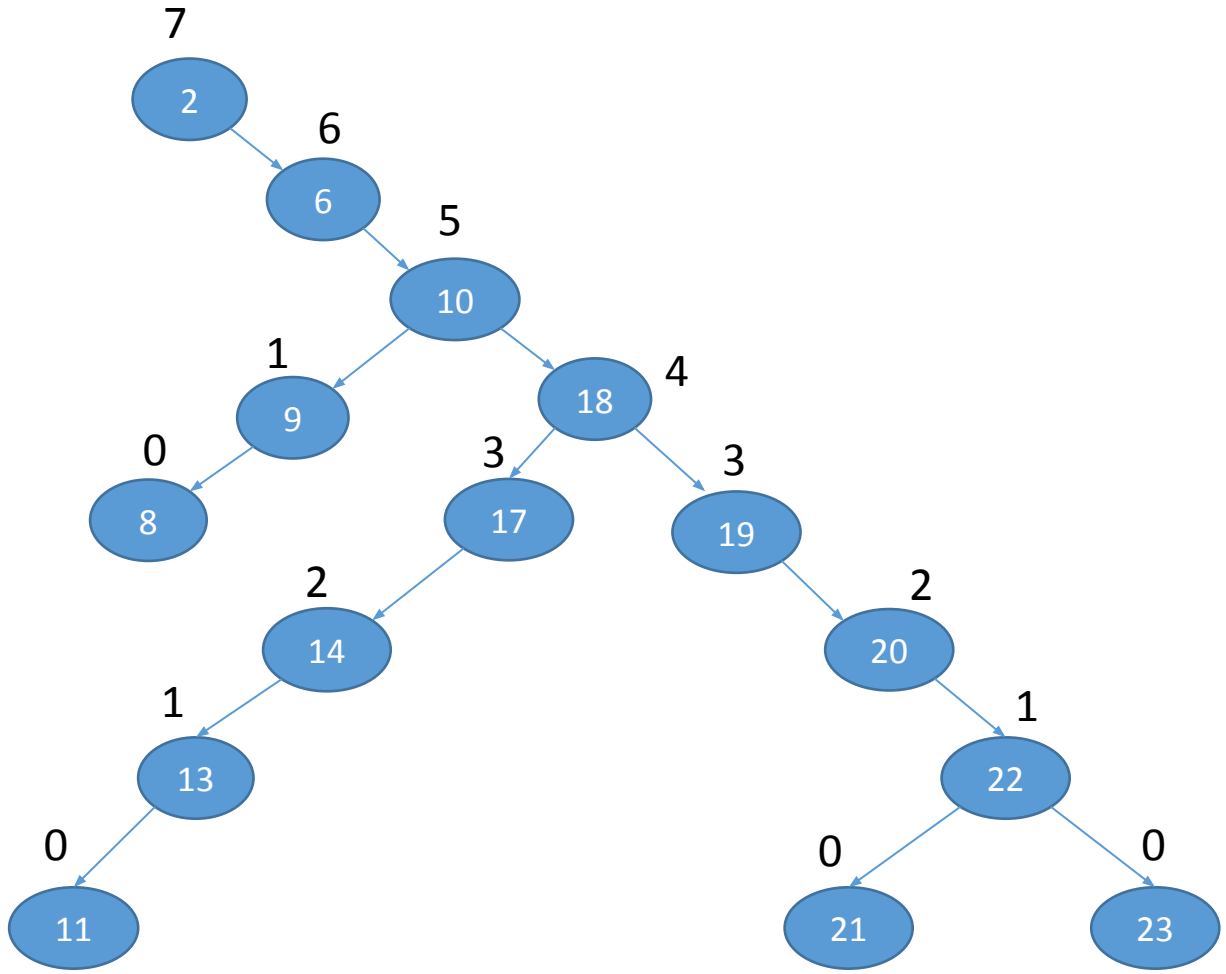
23, 22, 21, 20, 19, 18, 17, 14, 13, 11, 10, 9, 8, 7, 6, 4, 3, 2, 1

Примеры задач

1) Найти высоту дерева.
2) Определить, является ли дерево сбалансированнным по высоте.
3) Найти длину наибольшего полупути (корни полупутей наибольшей длины).
4) Проверить, является ли дерево идеально-сбалансированным по числу вершин.
5) Найти среднюю по значению вершину в дереве.
6) Найти среднюю по значению вершину среди вершин, у которых высоты поддеревьев совпадают.
7) Найти среднюю по значению вершину среди вершин некоторого пути.

1) Найти высоту дерева.

2) Определить, является ли дерево сбалансированнным по высоте (для всех вершин высоты их подеревьев должны отличаться не более, чем на 1.



Нужно знать высоту каждой вершины.

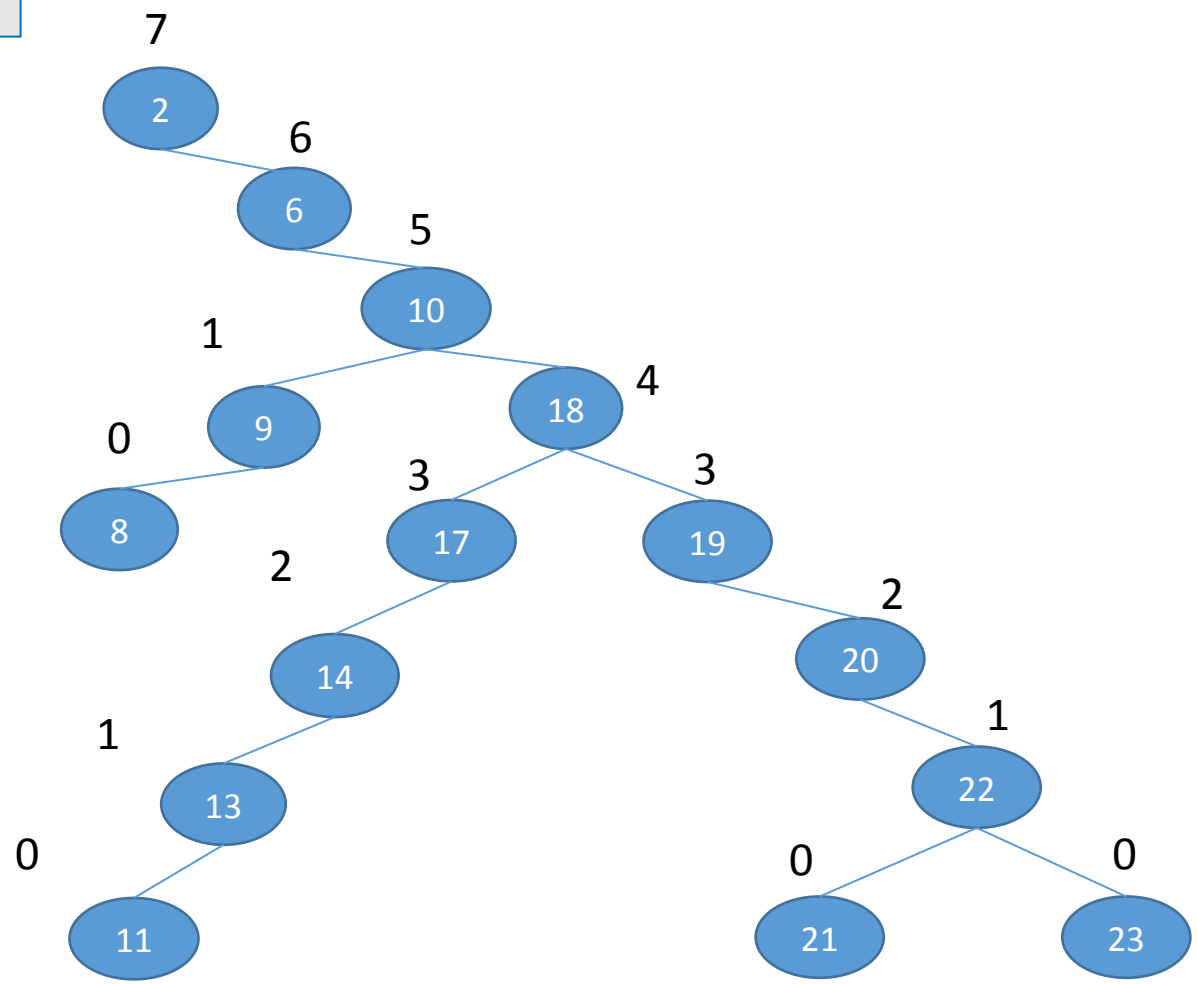
Обратный левый (или правый) **обход**

- если вершина **v** лист, то её высота равна 0;
- если у вершины **v** только одно поддерево, то её высота равна высоте этого поддерева +1;
- если у вершины **v** есть оба поддерева, то её высота равна максимуму из высот поддеревьев, увеличенному на 1.

3) Найти длину наибольшего полупути (корни полупутей наибольшей длины).

Зная метки высот, **можно найти длину наибольшего полупути и его корень**:

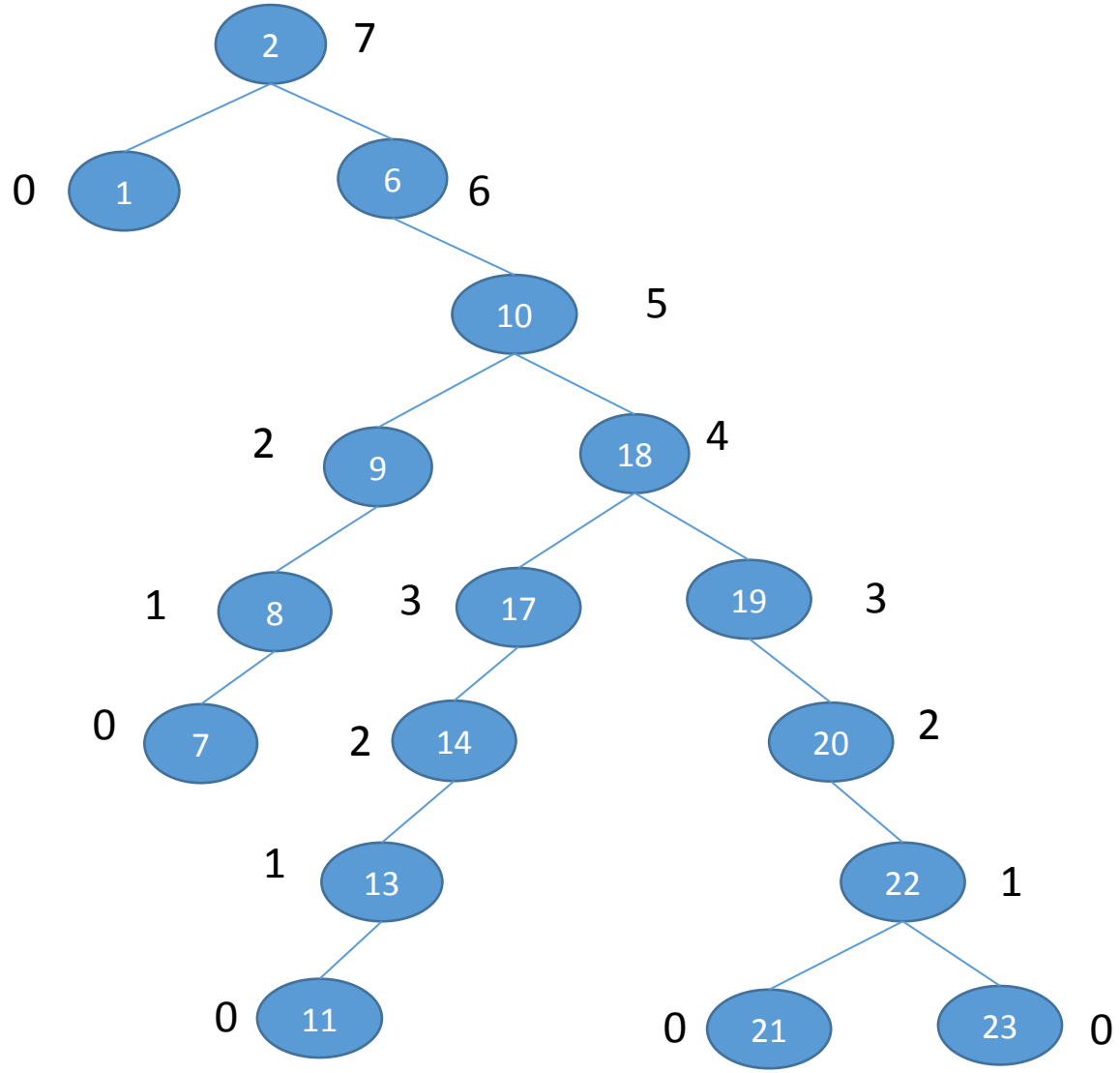
> найдём ту вершину **v**, для которой сумма меток высот её поддеревьев, увеличенная на число 2 или 1 (в зависимости от того, сколько поддеревьев у вершины **v**), является наибольшей.



Вершина 18: 3+3+2=8
является корнем наибольшего полупути.

Длина наибольшего полупути равна 8.

3) Найти длину наибольшего полупути и указать корни полупутей наибольшей длины.



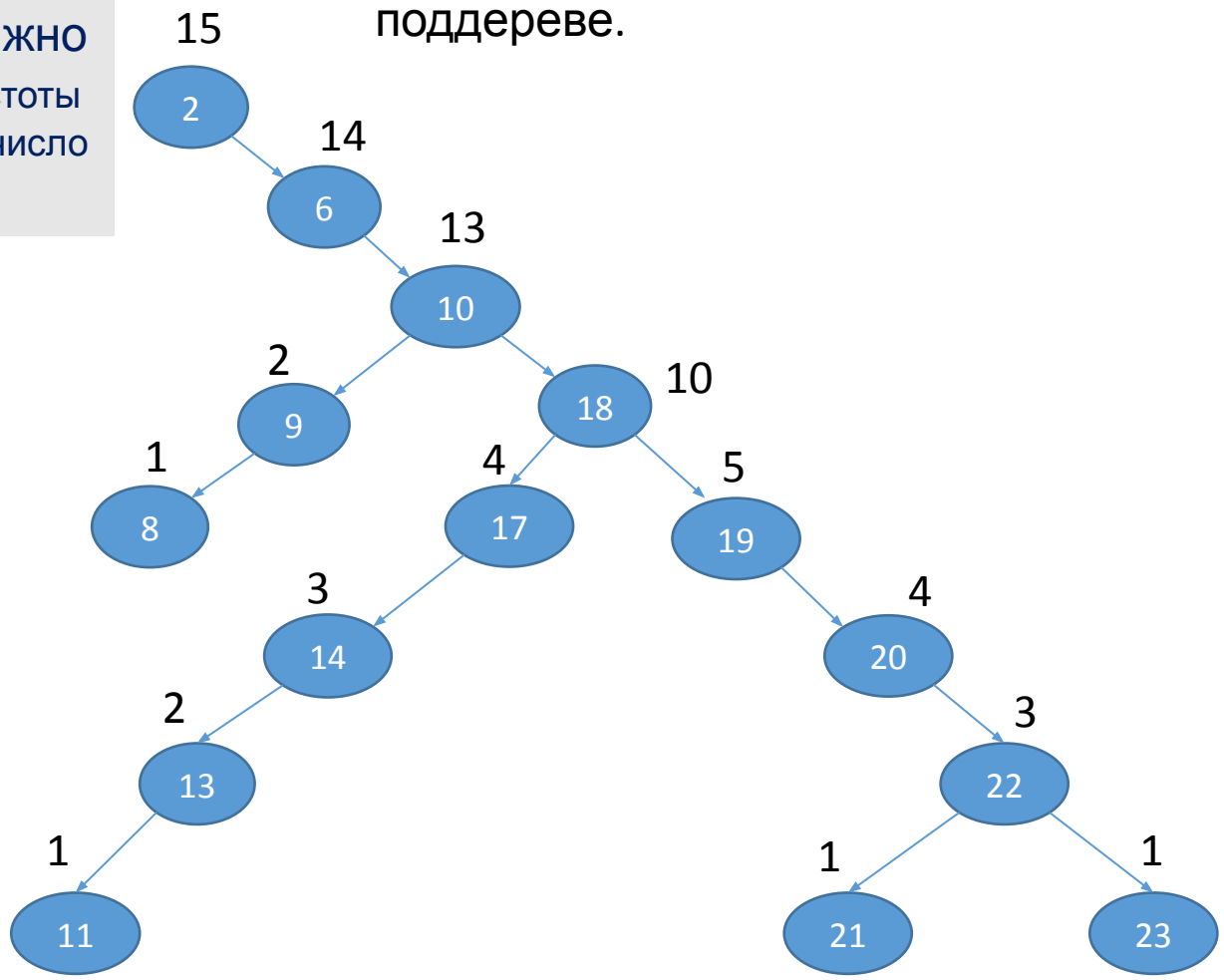
Вершины **2, 10, 18** являются корнями полупутей наибольшей длины **8**.

Вопрос 1.
Сколько полупутей наибольшей длины проходит через вершины?
1
2
10
18
19

Вопрос 2.
Через какие вершины пройдёт наибольшее число полупутей наибольшей длины?

4) Проверить, является ли дерево идеально-сбалансированным по числу вершин: для каждой вершины число вершин в поддеревьях должно отличаться не более, чем на 1 (для простоты вычислений, если у вершины отсутствует поддерево, то число вершин в таком поддереве полагается равным 0).

Нужно знать число вершин в каждом поддереве.



Обратный левый (или правый) **обход**

- если вершина v лист, то её метка равна 1;
- если у вершины v только одно поддерево, то её метка равна метке этого поддерева +1;
- если у вершины v есть оба поддерева, то её метка равна сумме меток поддеревьев, увеличенной на 1.

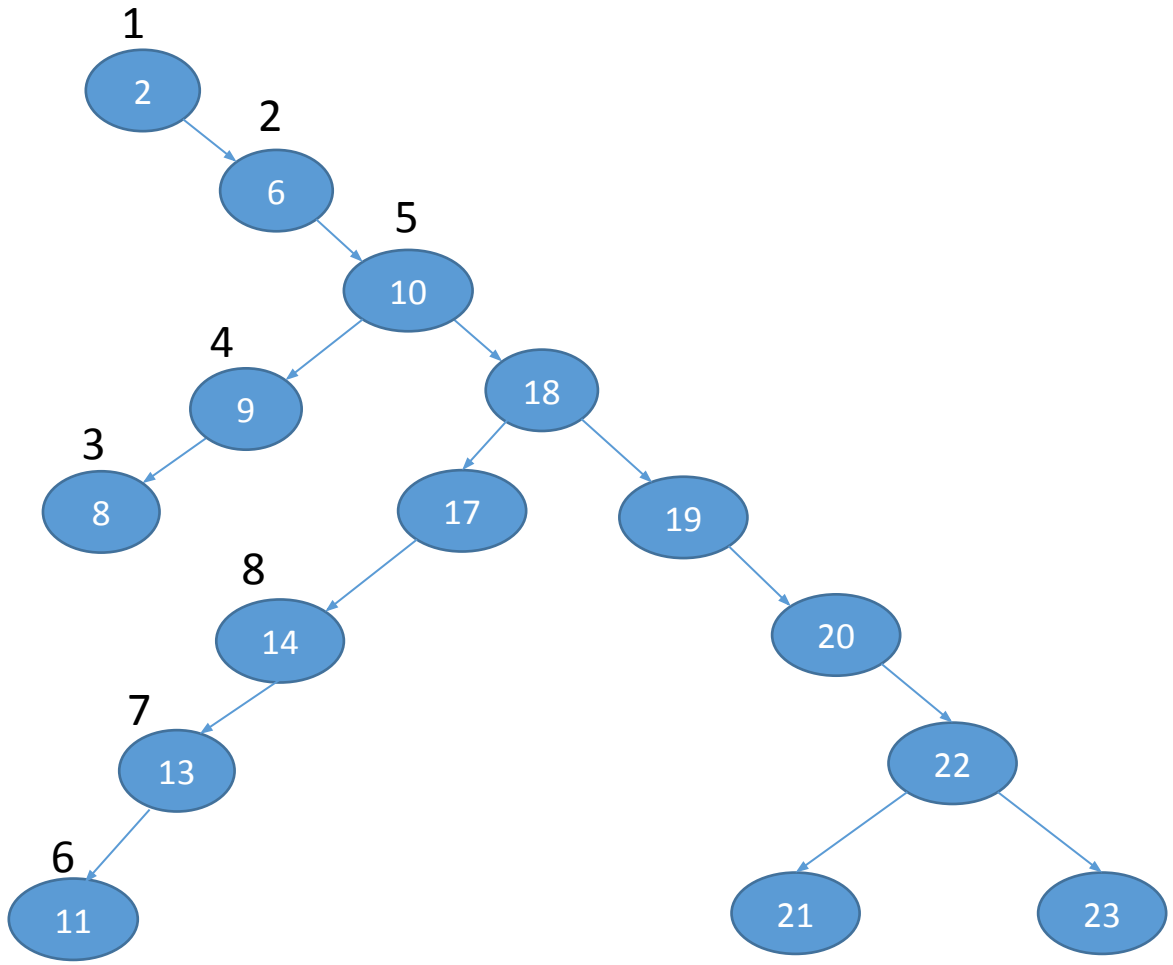
5) Найти среднюю по значению вершину в дереве

(не использовать дополнительную память, зависящую от n, решить задачу за линейное от количества вершин время).

Сначала нужно узнать, чётно или нет число вершин в дереве.

Если число вершин нечётно, то средняя вершина существует и её можно найти, просматривая вершины дерева, например, по неубыванию ключей.

1. Выполнить **любой обход** дерева и подсчитать число вершин (**n=15**).
2. Если n – чётно, то полагаем, что средней не существует.
3. Если n – нечётно, то выполним **внутренний обход**, считая пройденные во время этого обхода вершины. Остановимся, как только счётчик пройденных вершин станет равным [n/2]+1 **(=8)**.



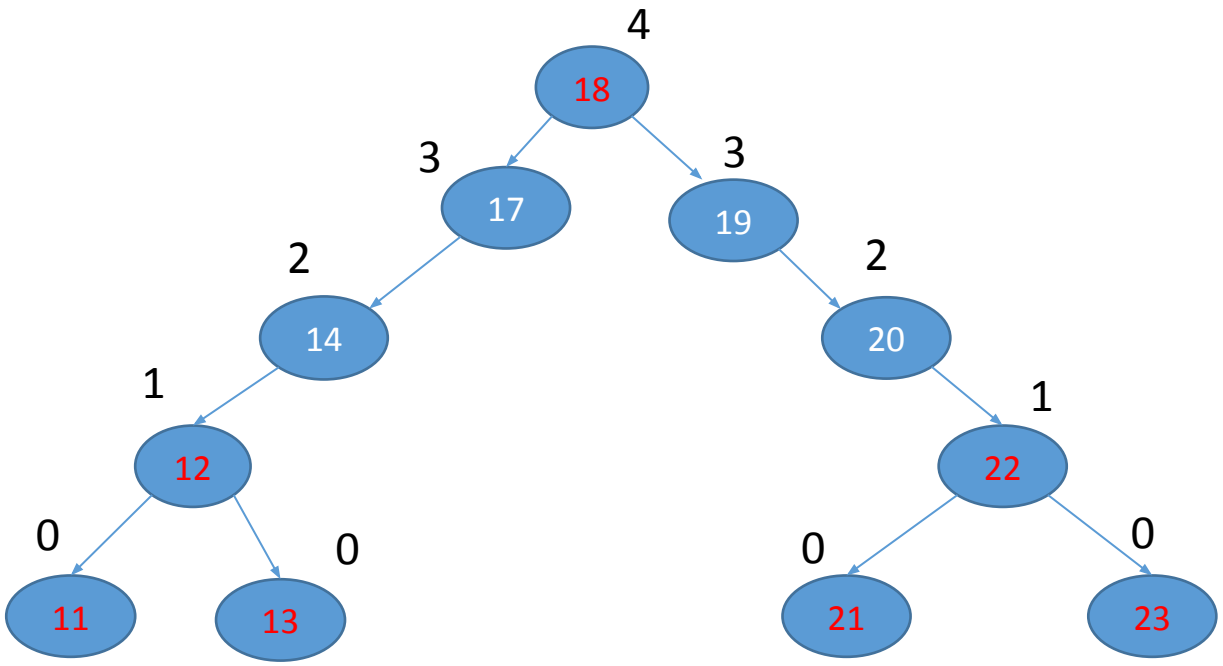
на рис. показана нумерация вершин при обходе.

```
def InOrderTraversal (v):
    if v is not None:
        InOrderTraversal (v.left)
        Action (v)
        InOrderTraversal (v.right)
```

6) Найти среднюю по значению вершину среди вершин, у которых высоты поддеревьев совпадают.

Сначала нужно определить нужные вершину, узнать чётно или нет их количество. Если нечётно, то средняя вершина существует и её можно найти, просматривая нужные вершины, например, по неубыванию ключей.

1. Сначала **обратным** обходом расставить вершинам метки высот. Во время этого же обхода подсчитать количество вершин, у которых метки высот поддеревьев совпали. Пусть у нас **m** таких вершин.
2. Если **m** – чётно, то средней не существует.
3. Если **m** – нечётно, то выполним **внутренний** обход, считая при этом только лишь те вершины, для которых высоты их поддеревьев совпадают. Остановимся, как только счётчик станет равным **[m/2]+1**.



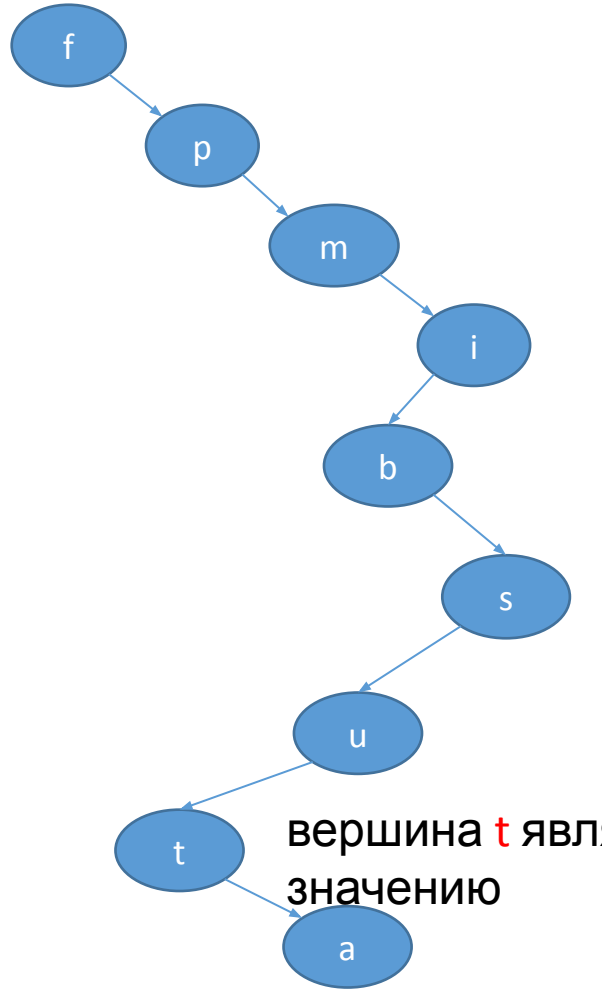
Внутренний (левый) обход:

11, 12, 13, 14, 17, *18*, 19, 20, 21, 22, 23

7) Найти среднюю по значению вершину среди вершин некоторого пути.

Предположим, что задан корень этого пути.

f	f				!!!			
p	f	p			!!!			
m	f	p	m		!!!			
i	f	p	m		!!!			i
b	f	p	m	b	!!!			i
s	f	p	m	b	!!!		s	i
u	f	p	m	b	!!!	u	s	i
t	f	p	m	b	t	u	s	i



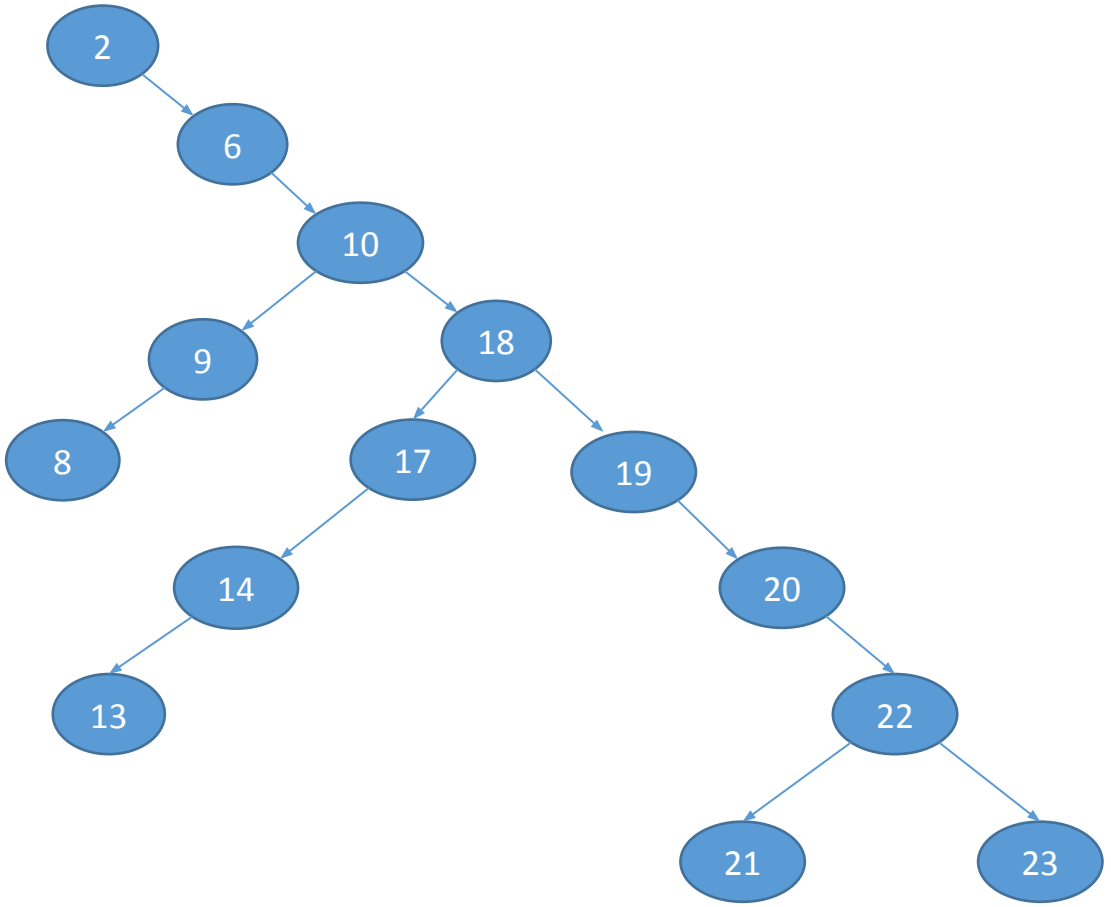
вершина t является средней по значению

Удаление вершины

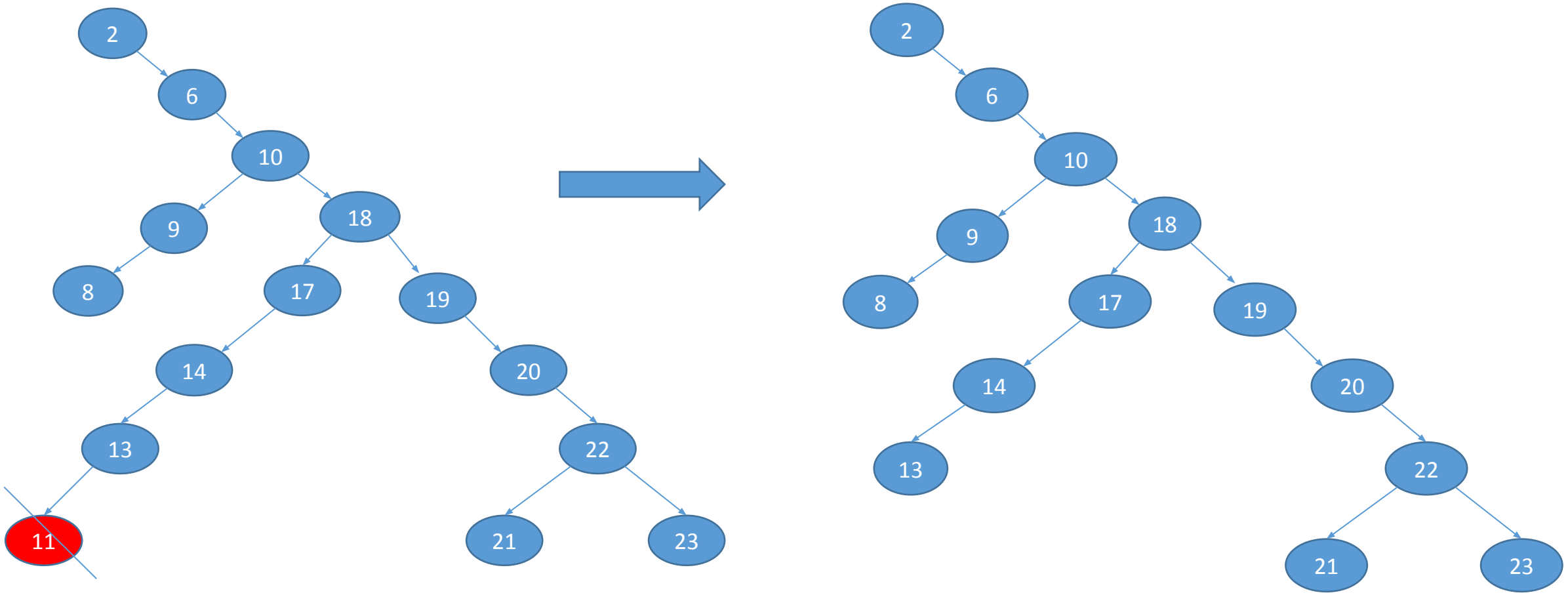
Случай 1. Удаляется лист.

Случай 2. Удаляется вершина, у которой есть только одно поддерево

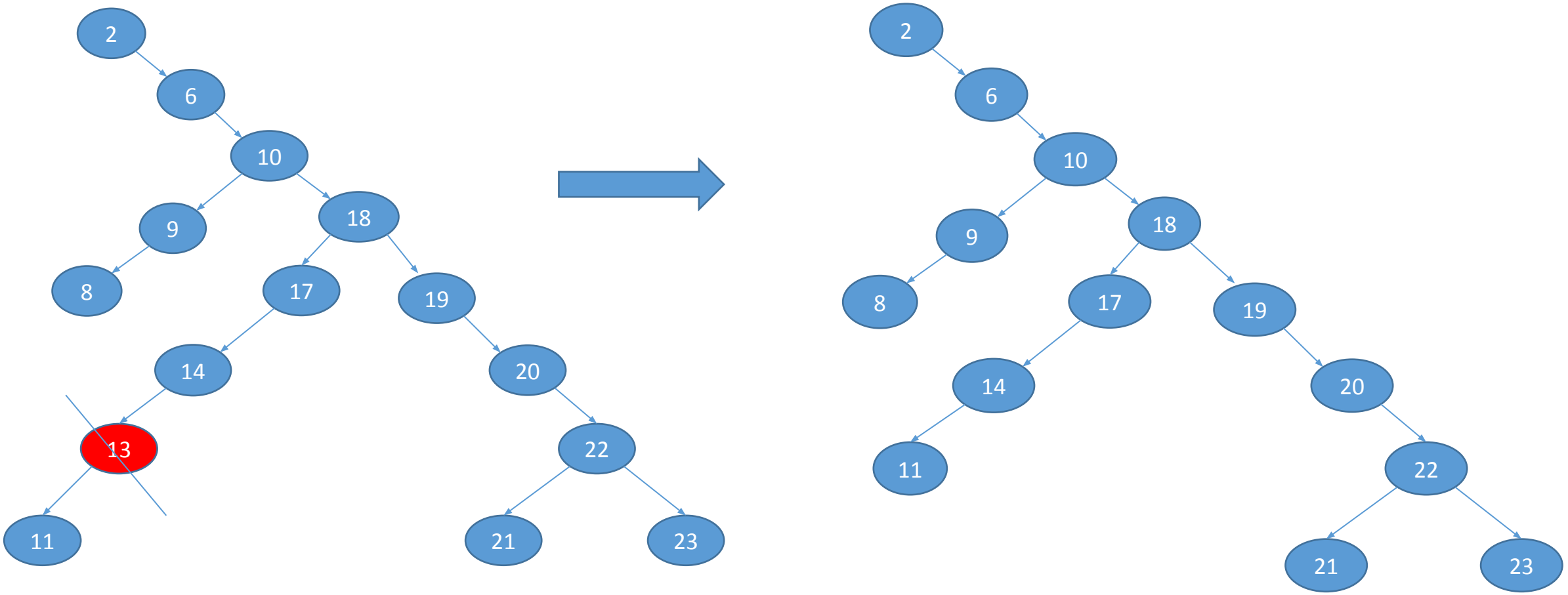
Случай 3. Удаляется вершина, у которой есть оба поддерева (возможно «правое» или «левое» удаление).



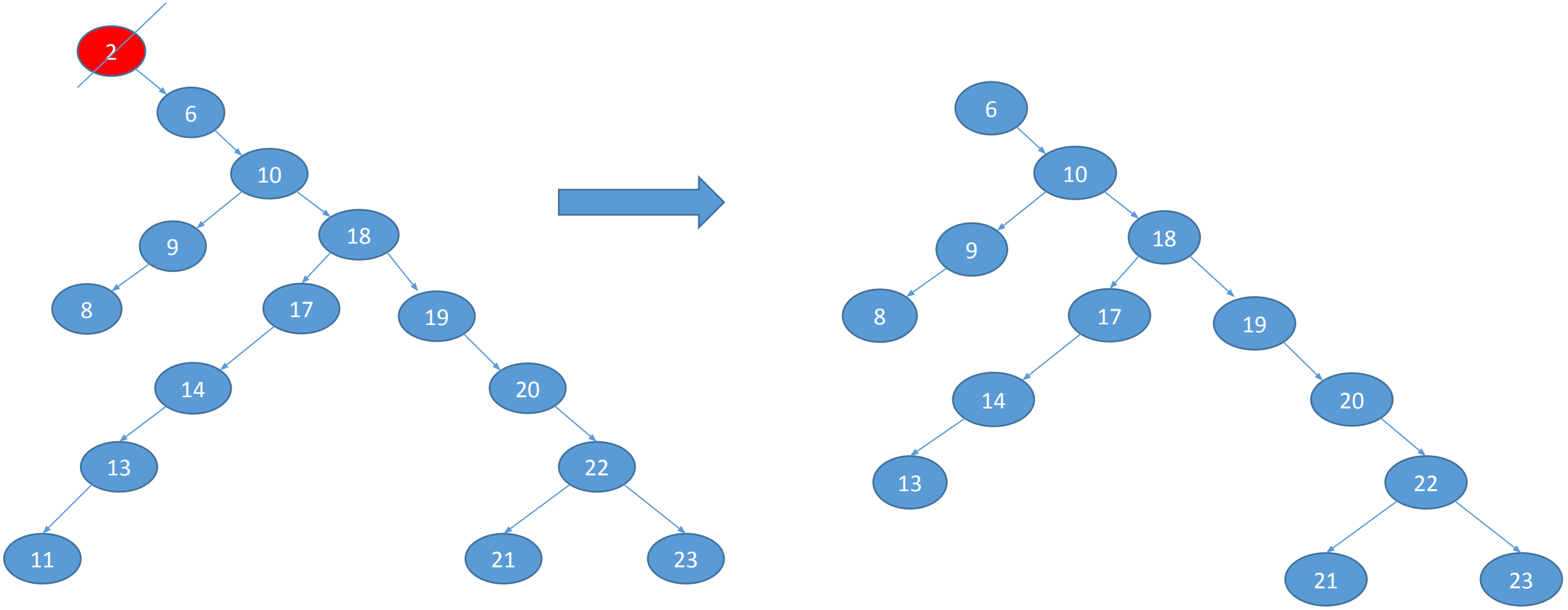
Случай 1. Удаляется лист.



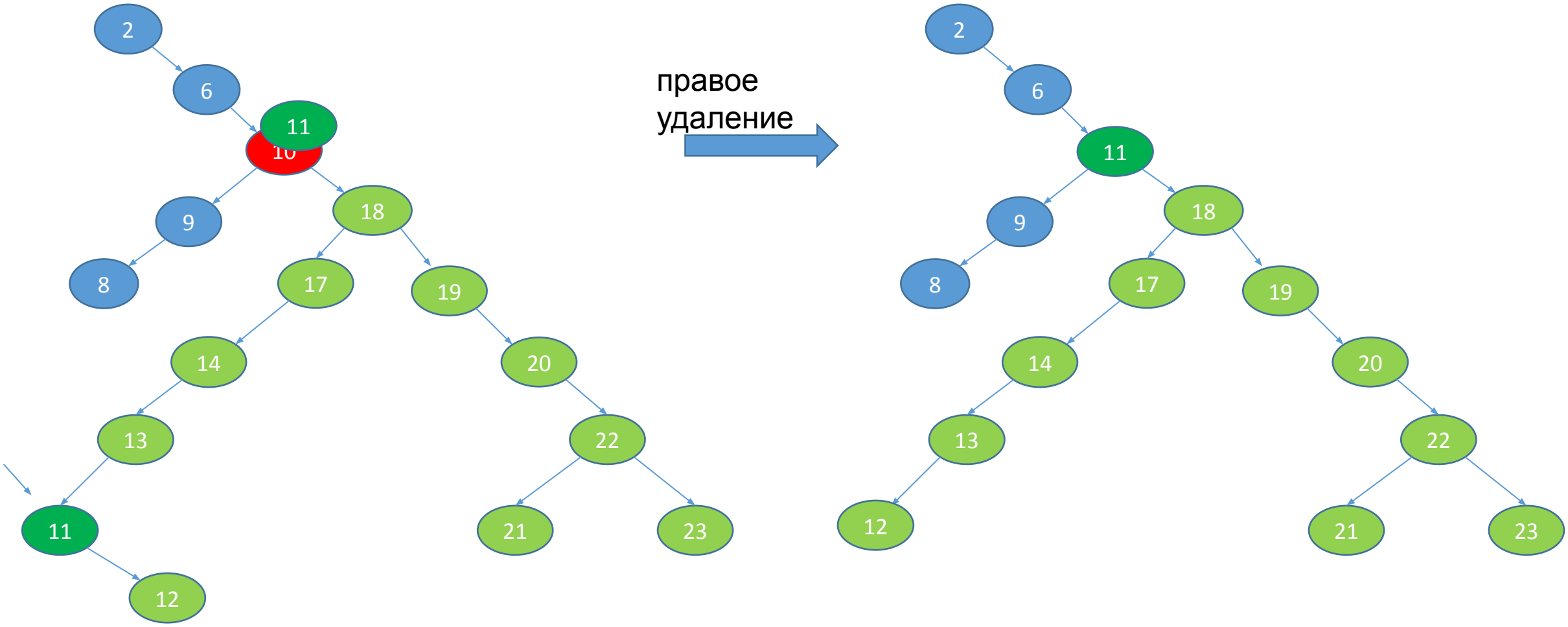
Случай 2. Удаляется вершина, у которой есть только одно поддерево.



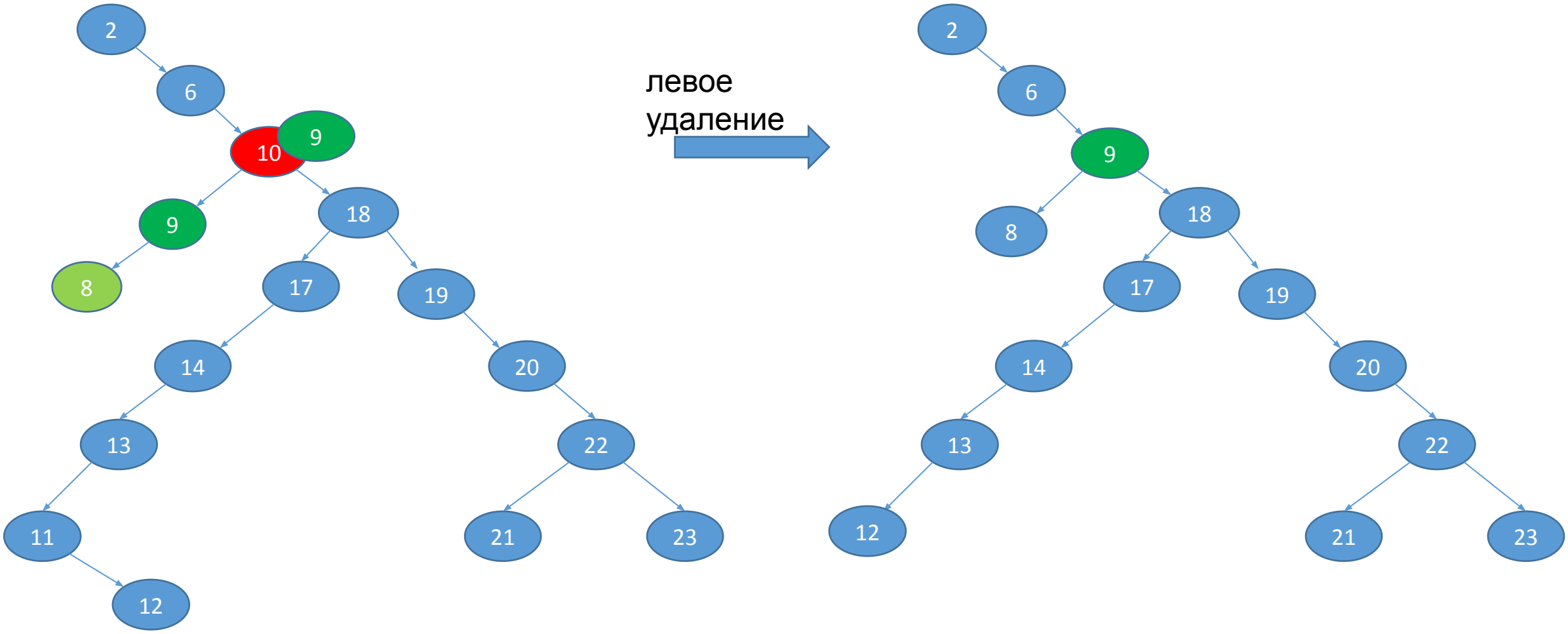
Случай 2. Удаляется вершина, у которой есть только одно поддерево



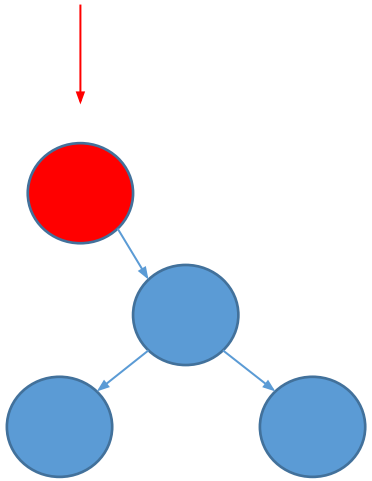
Случай 3. Удаляется вершина, у которой есть оба поддерева («правое» удаление).



Случай 3. Удаляется вершина, у которой есть оба поддерева («левое» удаление).

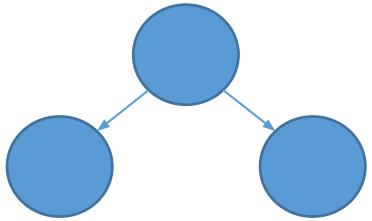


Найти вершину, которая удовлетворяет заданному свойству. Удалить эту вершину (правое удаление).

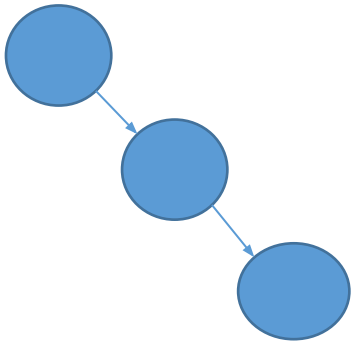


?

или



(рис.1)

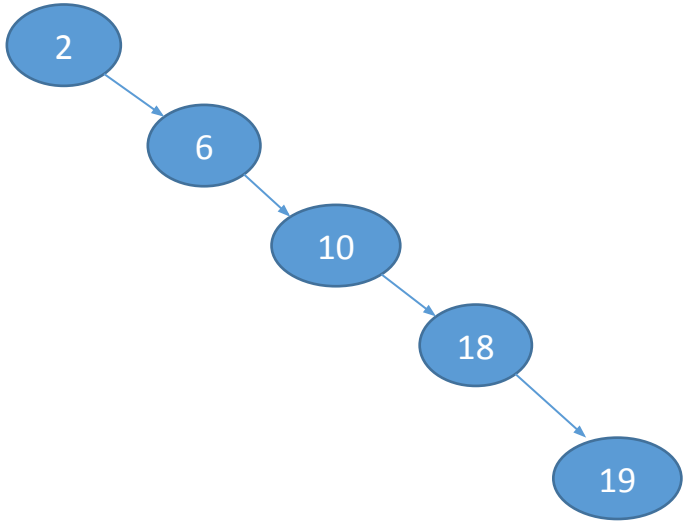


(рис.2)

!!! Если у удаляемой вершины только одно поддерево, то НЕТ ПОНЯТИЯ ПРАВОЕ/ЛЕВОЕ
удаление. Удаление всегда выполняется однозначно.

Ответ: правильно выполнено удаление на рис.
1.

Оценки числа операций в худшем случае



в худшем случае высота дерева
h = n-1

поиск элемента с заданным ключом ***x***	**h**
добавление элемента с заданным ключом ***x***	**h**
построение дерева для последовательности из **n** элементов	**n·h**
удаление элемента с заданным ключом ***x***	**h**
обход дерева из **n** вершин	**n**

В **1962** году советские учёные Г.М. **А**дельсон-**В**ельский и Е.М. **Л**андис предложили структуру данных **сбалансированного поискового дерева** (АВЛ-дерево).

Георгий Максимович **Адельсон-Вельский**

Дата рождения	8 января 1922
Место рождения	Самара, РСФСР
Дата смерти	26 апреля 2014 (92 года)
Место смерти	Гиватаим, Израиль
Страна	СССР → Израиль
Научная сфера	математик
Место работы	Институт теоретической и экспериментальной физики
Альма-матер	МГУ (мехмат)
Учёная степень	кандидат ф.-м. наук

Евгений Михайлович **Ландис**

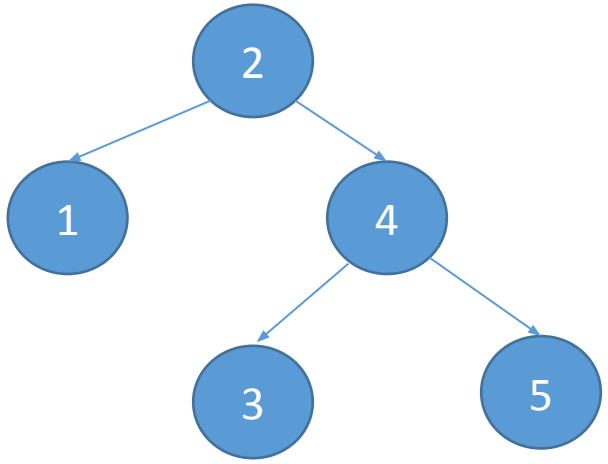
Дата рождения	6 октября 1921
Место рождения	Харьков
Дата смерти	12 декабря 1997 (76 лет)
Место смерти	Москва, Россия
Страна	СССР → Россия
Научная сфера	математика
Место работы	Московский государственный университет
Альма-матер	МГУ (мехмат)
Учёная степень, звание	доктор ф.-м. наук, профессор

В рамках дисциплины мы подробно исследуем эту структуру данных, а пока - краткая информация о ней.

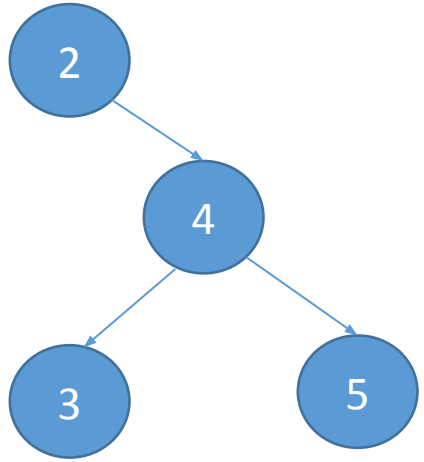
АВЛ-дерево – это бинарное поисковое дерево, которое является сблансированным по высоте.

Свойство сбалансированности по высоте:

для каждой вершины дерева высоты её поддеревьев отличаются не более, чем на 1.



АВЛ-дерево



не является АВЛ-деревом, так как для вершины 2 не выполняется свойство сбалансированности

ТЕОРЕМА. Пусть n – число внутренних вершин АВЛ-дерева, а h – его высота.

Тогда справедливы следующие неравенства:

$$\log(n+1) \le h < 1,4404 \cdot \log(n+2) - 0,328$$

Использование поисковых деревьев на практике

Сортировка деревом

Предположим, что на вход поступаю числа, среди которых нет повторяющихся. Необходимо выдать числа в порядке возрастания.

1. По последовательности чисел сначала построим АВЛ-дерево последовательным добавлением элемента.

$n*\log n$

2. Выполним внутренний левый обход дерева.

n

Какое время работы алгоритма сортировки деревом худшем случае, если на вход поступило n чисел?

$n*\log n$

Абстрактный тип данных: множество (**set**)

Множество (англ. set) —хранит набор попарно различных объектов без определённого порядка.

Интерфейс множества включает три основные операции:

1) Insert(x) — добавить в множество ключ x;
2) Contains(x) — проверить, содержится ли в множестве ключ x;
3) Remove(x) — удалить ключ x из множества.

Для реализации интерфейса множества обычно используются такие структуры данных, как:

- сбалансированные поисковые деревья: например, AVL-деревья, 2-3-деревья, красно-чёрные деревья.
- хеш-таблицы.

В стандартной библиотеке C++ есть контейнер std::set, который реализует множество на основе сбалансированного дерева (обычно красно-чёрного), и контейнер std::unordered_set, построенный на базе хеш-таблицы.

В языке Java определён интерфейс Set, у которого есть несколько реализаций, среди которых классы TreeSet (работает на основе красно-чёрного дерева) и HashSet (на основе хеш-таблицы).

В языке Python есть только встроенный тип set, использующий хеширование, но нет готового класса множества, построенного на сбалансированных деревьях.

Абстрактный тип данных ассоциативный массив (**map**)

Ассоциативный массив (англ. associative array), или отображение (англ. map), или словарь (англ. dictionary), — хранит пары вида (ключ, значение), при этом каждый ключ встречается не более одного раза.

Название «ассоциативный» происходит от того, что значения ассоциируются с ключами.

Интерфейс ассоциативного массива включает операции:

1) Insert(k,v) — добавить пару, состоящую из ключа k и значения v;
2) Find(k) — найти значение, ассоциированное с ключом k, или сообщить, что значения, связанного с заданным ключом, нет;
3) Remove(k) — удалить пару, ключ в которой равен k.

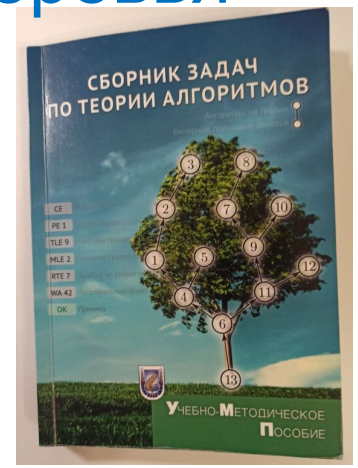
Данный интерфейс реализуется на практике теми же способами, что и интерфейс множества. Реализация технически немного сложнее, чем множества, но использует те же идеи.

Для языка программирования C++ в стандартной библиотеке доступен контейнер std::map, работающий на основе сбалансированного дерева (обычно красно-чёрного), и контейнер std::unordered_map, работающий на основе хеш-таблицы.

В языке Java определён интерфейс Map, который реализуется несколькими классами, в частности классом TreeMap (базируется на красно-чёрном дереве) и HashMap (базируется на хеш-таблице).

В языке Python очень широко используется встроенный тип dict. Этот словарь использует внутри хеширование.

Литература по теме: «Бинарные поисковые деревья»



Сборник задач по теории алгоритмов : учеб.-метод. пособие / В.М. Котов, Ю.Л. Орлович, Е.П. Соболевская, С.А. Соболь – Минск : БГУ, 2017. С. 122-180

https://acm.bsu.by/wiki/Программная_реализация_бинарных_поисковых_деревьев

Общие задачи в iRunner

0.1 построение дерева
0.2 удаление вершин из дерева
0.3 проверка является ли бинарное дерево поисковым

Индивидуальная задача по теме «Деревья поиска» в iRunner

Спасибо за внимание!