

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Романчук Иван Сергеевич

Должность: Ректор

Дата подписания: 30.03.2023 15:46:19

Уникальный программный ключ:

6319edc2b582ffdacea443f01d5779368d0957ac34f5cd074d81181530457479

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования  
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

М.С.ВОРОБЬЕВА, Е.А.ПАВЛОВА

## **СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ**

Методические рекомендации по выполнению лабораторных работ

# ТЕМА 1. АЛГОРИТМЫ: ПОСТРОЕНИЕ И АНАЛИЗ

## Лабораторная работа № 1

**Цель** – сформировать представление о математическом анализе алгоритмов, в том числе тех, которые основаны на рекурсии, изучить способы вычисления рекуррентных отношений.

### Ключевые понятия

Размерность задачи. Временная сложность алгоритмов: время выполнения в худшем случае, в среднем, в лучшем случае. Асимптотическая нотация: верхние оценки временной сложности, точные оценки, нижние оценки. Классификация алгоритмов по временной сложности.

Рекурсивные алгоритмы. Рекуррентные отношения. Способы вычислений рекуррентных отношений: метод подстановки, метод итераций, основная теорема.

### Упражнения

1. Какое утверждение является истинным?

1)  ~~$\sqrt{N} = O(\log N)$~~

2)  $2^{N+1} = O(2^N)$ ;

3)  ~~$3^{2N} = O(2^{3N})$~~

4)  ~~$\left(\frac{3}{2}\right)^N = O\left(\left(\frac{1}{3}\right)^N\right)$~~

Ответ. Запись  $T(N) = O(g(N))$  означает: существует такая константа  $c > 0$  и число  $N_0$ , что  $0 \leq T(N) \leq c \cdot g(N)$  для всех  $N \geq N_0$ .

Проверим утверждение  ~~$\sqrt{N} = O(\log N)$~~ . Предположим, что существуют константы  $c$  и  $N_0$  такие, что для всех  $N \geq N_0$  выполняется неравенство  ~~$\sqrt{N} \leq c \log N$~~ , тогда  $c \geq \frac{\sqrt{N}}{\log N}$  для всех  $N \geq N_0$ . Но  $\frac{\sqrt{N}}{\log N}$  принимает любое, как угодно большое, значение при достаточно большом  $N$ , поэтому не существует такой константы  $c$ , которая могла бы превзойти  $\frac{\sqrt{N}}{\log N}$  для всех  $N$ , то есть  $\sqrt{N}$  не может иметь порядок роста  $O(\log N)$ .

Проверим утверждение  $2^{N+1} = O(2^N)$ .

Предположим, что существуют константы  $c$  и  $N_0$  такие, что для всех  $N \geq N_0$  выполняется неравенство  $2^{N+1} \leq c2^N$ , тогда  $c \geq \frac{2^{N+1}}{2^N}$  для всех  $N \geq N_0$ , то есть при  $c = 2$  и  $N = 1$  неравенство  $2^{N+1} \leq c \cdot 2^N$  выполняется. Следовательно, утверждение  $2^{N+1} = O(2^N)$  является истинным.

Проверим утверждение  ~~$3^{2N+1} = O(3^{2N})$~~ . Предположим, что существуют константы  $c$  и  $N_0$  такие, что для всех  $N \geq N_0$  выполняется неравенство  ~~$3^{2N+1} \leq c3^{2N}$~~ . Разделим обе части неравенства на  $N \log N$ , тогда  ~~$c \geq \frac{3N}{\log N} + 1$~~  для всех  $N \geq N_0$ . Но  $\frac{3N}{\log N} + 1$  принимает любое, как угодно большое, значение при достаточно большом  $N$ , поэтому не существует такой константы  $c$ , которая могла бы превзойти  $\frac{3N}{\log N} + 1$  для всех  $N$ , то есть  ~~$3^{2N+1} = O(3^{2N})$~~  не может иметь порядок роста  ~~$O(3^{2N})$~~ .

Проверим утверждение  ~~$\left(\frac{3}{2}\right)^N = O\left(\left(\frac{1}{3}\right)^N\right)$~~ . Предположим, что существуют константы  $c$  и  $N_0$  такие, что для всех  $N \geq N_0$  выполняется неравенство  ~~$\left(\frac{3}{2}\right)^N \leq c\left(\frac{1}{3}\right)^N$~~ . Разделим обе части неравенства на  $\left(\frac{1}{3}\right)^N$ , тогда  ~~$c \geq \left(\frac{9}{2}\right)^N$~~  для всех  $N \geq N_0$ . Но  $\left(\frac{9}{2}\right)^N$  принимает любое, как угодно большое, значение при достаточно большом  $N$ , поэтому не существует такой константы  $c$ , которая могла бы превзойти  $\left(\frac{9}{2}\right)^N$  для всех  $N$ , то есть  $\left(\frac{3}{2}\right)^N$  не может иметь порядок роста  $O\left(\left(\frac{1}{3}\right)^N\right)$ .

2. Пусть время работы алгоритма  $T(N) = O(N^2)$ . Если 1000 элементов обрабатываются за 2 мсек., то во сколько раз следует ожидать увеличения

времени выполнения при обработке 3000 элементов?

Ответ. Если количество элементов увеличивается в три раза, значит время работы алгоритма  $T(3N) = O((3N)^2)$ , или  $T(3N) = O(9N^2)$ , то есть следует ожидать увеличения времени выполнения в 9 раз.

3. Найти наиболее точную оценку для рекуррентных отношений:

1)  $T(N) = T(N - 1) + 1$  при  $N > 1, T(1) = 1$ ;

2)  $T(N) = 4T(N/2) + N, T(1) = 1$ ;

3)  $T(N) = 4T(N/2) + N^2, T(1) = 1$ ;

4)  $T(N) = 4T(N/2) + N^3, T(1) = 1$ .

Ответ. Для решения уравнения (а) воспользуемся методом итераций, который заключается в постепенном раскрытии рекуррентного отношения до тех пор, пока оно не станет зависеть только от  $N$  и начальных условий.

Раскрывая это отношение по формуле  $T(N) = T(N - 1) + 1$ , имеем:

$$T(N - 1) = T(N - 2) + 1;$$

$$T(N - 2) = T(N - 3) + 1;$$

$$T(N - 3) = T(N - 4) + 1;$$

...

$$T(N - (N - 2)) = T(N - (N - 1)) + 1, \text{ то есть } T(2) = T(1) + 1.$$

Подставляя полученные отношения в исходное уравнение, получаем:

$$T(N) = T(N - 1) + 1 = T(N - 2) + 2 = T(N - 3) + 3 = \dots = T(1) + (N - 1) = 1 + (N - 1) = N.$$

Следовательно,  $T(N) = O(N)$ .

Для решения уравнений (б, в, г) воспользуемся основным методом, который применяется для вычисления рекуррентных отношений вида:

$$T(N) = a T(N/b) + f(N), \tag{1}$$

где  $a \geq 1, b > 1$  — некоторые константы,  $f$  — положительная функция (по крайней мере для больших значений аргументов).

Соотношение (1) возникает в тех алгоритмах, которые делят задачу размера  $N$  на  $a$  подзадач размера  $N/b$ , эти задачи решаются рекурсивно за

время  $T(N/b)$  и результаты объединяются. Затраты времени на разбиение и объединение описываются функцией  $f(N)$ .

Основной метод заключается в применении **основной теоремы**, которая формулируется следующим образом.

Пусть 
$$T(N) = a T(N/b) + f(N),$$

где  $a \geq 1$ ,  $b > 1$  — некоторые константы, а  $f$  — асимптотически положительная функция.

1. если  $f(N) = O(N^{\log_b a - \varepsilon})$  для некоторого  $\varepsilon > 0$ , то  $T(N) = O(N^{\log_b a})$ ;
2. если  $f(N) = \Theta(N^{\log_b a})$ , то  $T(N) = \Theta(N^{\log_b a} \log N)$ ;
3. если  $f(N) = \Omega(N^{\log_b a + \varepsilon})$  для некоторого  $\varepsilon > 0$  и если  $a f(N/b) \leq c f(N)$  для некоторой  $c < 1$  и достаточно больших  $N$ , то  $T(N) = O(f(N))$ .

Суть теоремы заключается в следующем. Сравним  $f(N)$  с  $N^{\log_b a - \varepsilon}$ . Функция, которая растёт быстрее другой, причём с «запасом», равным  $N^\varepsilon$ , и определяет порядок роста  $T(N)$ . Если же функции одного порядка, то появляется дополнительный логарифмический множитель.

В рекуррентном отношении (b)  $a = 4$ ,  $b = 2$ ,  $f(N) = N$ .

Вычислим:  $N \log_4 N$ . Поскольку  $N \in O(N^{\log_4 4 - \varepsilon})$  для  $\varepsilon = 1$ , то выполняется 1 случай основной теоремы, поэтому  $T(N) = \Theta(N^2)$ .

В рекуррентном отношении (c)  $a = 4$ ,  $b = 2$ ,  $f(N) = N^2$ .

Вычислим:  $N^2 \log_4 N$ . Поскольку  $N^2 \in O(N^{\log_4 4})$ , то выполняется 2 случай основной теоремы, поэтому  $T(N) = \Theta(N^2 \log N)$ .

В рекуррентном отношении (d)  $a = 4$ ,  $b = 2$ ,  $f(N) = N^3$ .

Вычислим:  $N^3 \log_4 N$ . Поскольку  $N^3 \in \Omega(N^{\log_4 4 + \varepsilon})$  для любого  $0 < \varepsilon \leq 1$  и  $c f(N/b) \leq c f(N)$  для  $c = 1/2$ , то выполняется 3 случай основной теоремы, поэтому  $T(N) = \Theta(N^3)$ .

4. Используя O- символику, найдите время выполнения (как функции от N) в

наихудшем случае процедуры сортировки «пузырьком», которая упорядочивает массив целых чисел в неубывающем порядке.

Ответ. Рассмотрим процедуру bubble:

```
type Tvector = array[1..N] of iNteger;  
procedure Bubble(var x: Vector);  
VAR i, j, temp: integer;  
begin  
  {1}   for i:= N downto 2 do  
    {2}     for j:= 1 to i-1 do  
      {3}       if x[j] > x[j+1] then  
        begin  
          {4}           temp:= x[j];  
          {5}           x[j]:= x[j+1];  
          {6}           x[j+1]:= temp;  
        end  
      end;  
  end; {bubble}
```

За каждый проход внутреннего цикла (операторы {3} — {6}) «пузырёк» с наибольшим элементом «всплывает» в начало массива. Число элементов  $N$ , подлежащих сортировке, может служить мерой объёма входных данных. Сначала отметим, что все операторы присваивания имеют некоторое постоянное время выполнения, не зависящее от размера входных данных. Таким образом, операторы {4} — {6} имеют время выполнения порядка  $O(1)$ , то есть равнозначно некоторой константе. В соответствии с правилом сумм время выполнения этой группы операторов:  $O(\max(1, 1, 1)) = O(1)$ .

Теперь посчитаем время выполнения условных и циклических операторов. Операторы **if** и **for** вложены друг в друга, поэтому пойдём от внутренних операторов к внешним, последовательно определяя время выполнения условного оператора и каждой итерации цикла. Для оператора **if** проверка логического выражения занимает время порядка  $O(1)$ . Мы не знаем,

будут ли выполняться операторы в теле условного оператора (строки {4} — {6}), но поскольку ищется наихудшее время выполнения, то, естественно, предполагаем, что они выполняются. Таким образом, получаем, что время выполнения группы операторов {3} — {6} имеет порядок  $O(1)$ .

Далее рассмотрим группу {2} — {6} операторов внутреннего цикла. Общее правило вычисления времени выполнения цикла заключается в суммировании времени выполнения каждой итерации цикла. Для операторов {2} — {6} время выполнения на каждой итерации имеет порядок  $O(1)$ . Цикл выполняется  $N-i$  раз, поэтому по правилу произведений общее время выполнения цикла:  $O((N-i) \cdot 1) = O(N-i)$ .

Теперь перейдём к внешнему циклу, который содержит все исполняемые операторы программы. Оператор {1} выполняется  $N-1$  раз, поэтому суммарное время выполнения программы определяется следующим образом:

$$\text{формула суммы} = N(N-1)/2 = N^2/2 - N/2,$$

которое имеет порядок  $O(N^2)$ . Таким образом, процедура сортировки «пузырьком» выполняется за время, пропорциональное квадрату числа элементов, подлежащих упорядочиванию.

### Задания для самостоятельной работы

| № варианта | 1.          | 2.          | 3.          | 4.          | 5.          | 6.          | 7.          | 8.          | 9.          | 10.         | 11.         | 12.         | 13.         | 14.         | 15.         |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| № задания  | a,b,<br>c,p | d,e,<br>f,q | g,h,<br>i,r | j,k,<br>l,s | t,m,<br>n,o | a,d,<br>g,p | b,f,<br>h,q | k,n,<br>e,r | c,f,<br>g,s | i,l,<br>n,t | a,h,<br>o,p | f,i,<br>l,q | b,k,<br>m,r | c,k,<br>o,s | b,i,<br>m,t |

1. Проверить утверждения с помощью определения.

- |                             |                              |
|-----------------------------|------------------------------|
| a) $17 = O(1)$ ;            | g) $3N^3 + 2N^2 = O(N^2)$ ;  |
| b) $N(N - 1)/2 = O(N^2)$ ;  | h) $N \log N + 5 = O(N)$ ;   |
| c) $2^{2N} = O(2^N)$ ;      | i) $2^N + N^3 = O(2^N)$ ;    |
| d) $\log N = O(N)$ ;        | j) $f(N) = O(f(N))$ ;        |
| e) $N \log N = O(N^2)$ ;    | k) $f(N) = O(f(N)^2)$ ;      |
| f) $N \ln N = O(N^{3/2})$ ; | l) $(N + 1) \log N = O(N)$ ; |

m)  ~~$4\sqrt{N} \log N$~~ ;

n)  ~~$2^N = O(2^N)$~~ ;

o)  ~~$N \log N$~~ ;

p)  $N^3 + 2N^2 = \Omega(N^2)$ ;

q)  ~~$\log N = \Omega(N)$~~

r)  $N \log N = \Omega(N^2)$ ;

s)  $3N^2 + 2N - 5 = \Omega(N)$ ;

t)  $N + \log N = \Omega(\log N)$ .

2. Пусть время работы алгоритма  $T(N) = O(f(N))$ . Если X элементов обрабатываются за Y мсек., то во сколько раз следует ожидать увеличения времени выполнения при обработке Z элементов?

| № варианта | f(N)     | X    | Y  | Z     | № варианта | f(N)     | X    | Y   | Z     |
|------------|----------|------|----|-------|------------|----------|------|-----|-------|
| 1.         | $N^2$    | 1000 | 5  | 3000  | 2.         | $N^3$    | 3000 | 50  | 9000  |
| 3.         | N        | 2000 | 10 | 5000  | 4.         | $2^N$    | 1000 | 100 | 2000  |
| 5.         | $N^3$    | 3000 | 10 | 6000  | 6.         | $\log N$ | 5000 | 8   | 20000 |
| 7.         | $2^N$    | 4000 | 4  | 4002  | 8.         | 1        | 1000 | 12  | 3000  |
| 9.         | $\log N$ | 5000 | 2  | 15000 | 10.        | $\log N$ | 2000 | 10  | 6000  |
| 11.        | 1        | 1000 | 4  | 4000  | 12.        | $N^3$    | 1000 | 20  | 2000  |
| 13.        | $N^2$    | 2000 | 11 | 6000  | 14.        | $N^2$    | 2000 | 15  | 10000 |
| 15.        | N        | 3000 | 20 | 9000  | 16.        | $2^N$    | 3000 | 10  | 3003  |

3. Найти наиболее точную оценку для рекуррентных отношений.

| № варианта | 1.          | 2.          | 3.          | 4.          | 5.          | 6.          | 7.          | 8.          | 9.          | 10.         | 11.         | 12.         | 13.         | 14.         | 15.         |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| № задания  | a,b,<br>c,p | d,e,<br>f,q | g,h,<br>i,r | j,k,<br>l,s | p,m,<br>n,o | a,d,<br>g,q | b,f,<br>h,r | k,n,<br>e,s | c,f,<br>g,p | i,l,<br>n,q | a,h,<br>o,r | f,i,<br>l,s | b,k,<br>m,p | c,k,<br>o,q | b,i,<br>m,r |

a)  $T(N) = 3T(N/2) + N, T(1) = 1$ ;

b)  $T(N) = 3T(N/2) + N^2, T(1) = 1$ ;

c)  $T(N) = 8T(N/2) + N^3, T(1) = 1$ ;

d)  $T(N) = 2T(N/2) + N \log N, T(1) = 1$ ;

e)  $T(N) = 9T(N/2) + N^3, T(1) = 1$ ;

f)  $T(1) = 1, T(N) = 2T(N/2) + 1, \text{ при } N > 1$ ;

g)  $T(N) = 3T(N/2) + N^2, T(1) = 1$ ;

h)  $T(N) = 3T(N/2) + N \log N, T(N) — \text{константа при } N \leq 8$ ;

- i)  $T(N) = 16T(N/4) + N^2$ , если  $T(N)$  — константа при  $N \leq 2$ ;
- j)  $T(N) = 9T(N/2) + N^2$ ,  $T(1) = 1$ ;
- k)  $T(N) = 7T(N/2) + N^2$ , если  $T(N)$  — константа при  $N \leq 2$ ;
- l)  $T(N) = 2T(N/4) + N^{1/2}$ , если  $T(N)$  — константа при  $N \leq 2$ ;
- m)  $T(N) = 2T(N/4) + \sqrt{N}$ ,  $T(1) = 1$ ;
- n)  $T(N) = 2T(N/2) + N \log N$ , если  $T(N)$  — константа при  $N \leq 2$ ;
- o)  $T(N) = T(9N/10) + N$ ; если  $T(N)$  - константа при  $N \leq 2$ ;
- p)  $T(N) = 2T(N - 1) + 1$ ,  $T(1) = 2$ ;
- q)  $T(N) = T(N - 1) + N$ ,  $T(1) = 1$ ;
- r)  $T(N) = 2T(N - 1) + N$ ,  $T(1) = 2$ ;
- s)  $T(N) = T(N - 1) + \log N$ ,  $T(1) = 2$ .

4. Используя  $O$ - символику, найдите время выполнения (как функции от  $N$ ) процедуры или функции в наихудшем случае.

| № варианта | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | 11. | 12. | 13. | 14. | 15. |
|------------|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|
| № задания  | a  | b  | c  | d  | a  | b  | c  | d  | a  | b   | c   | d   | b   | c   | d   |

a) **procedure** matmpy (N: integer);

var i, j, k: integer;

**begin**

for i:= 1 to N do

for j:= 1 to N do

**begin**

C[i,j]:= 0;

for k:= 1 to N do

C[i,j]:= C[i,j] + A[i,k] \* B[k,j]

**end**

**end;**

b) **procedure** mystery (N: integer);

var i, j, k: integer;

**begin**

for i:= 1 to N-1 do

for j:= i+1 to N do

for k:= 1 to j do

{группа операторов с временем выполнения  $O(1)$ }

**end;**

c) **procedure** veryodd (N: integer);

```

var i, j, x, y: integer;
begin
  for i:= 1 to N do
    if odd(i) then
      begin
        for j:= 1 to N do x:= x+1;
        for j:= 1 to i do y:= y+1;
      end
    end;
d) function recursive (N: integer): integer;
begin
  if N<= 1 then return (1)
  else
    return (recursive (N-1) + recursive (N-2))
  end;

```

### Тестовые задания

1. Какое утверждение является истинным?

a)  $(N + 5)^3 = O(N^3)$

c)  ~~$5^N = O(N^5)$~~

b)  ~~$\log_2 N = O(\sqrt{N})$~~

d)  ~~$2^N = O(2^N)$~~

2. Какое утверждение не является истинным?

a) если  $f1(N) = O(g(N))$  и  $f2(N) = O(g(N))$ , то  $f1(N) = f2(N)$

b)  $f(N) = \Theta(g(N))$  и  $g(N) = \Theta(h(N))$  влечёт  $f(N) = \Theta(h(N))$

c)  $f(N) = O(g(N))$  и  $g(N) = O(h(N))$  влечёт  $f(N) = O(h(N))$

d)  $f(N) = \Omega(g(N))$  и  $g(N) = \Omega(h(N))$  влечёт  $f(N) = \Omega(h(N))$

3. Пусть время работы алгоритма  $T(N) = O(\log N)$ . Если 1000 элементов обрабатываются за 1 мсек., то при обработке 3000 элементов следует ожидать увеличения времени выполнения

a) на постоянную величину

c) в 3 раза

b) в 9 раз

d) в 6 раз

4. Пусть время работы алгоритма  $T(N) = O(N^3)$ . Если 1000 элементов обрабатываются за 2 мсек., то при обработке 3000 элементов следует ожидать увеличения времени выполнения

a) в 27 раз

b) в 30 раз

c) в 9 раз

d) в 6 раз

5. Нижняя оценка времени работы алгоритма показывает

- a) лучшее время работы
  - b) гарантированное время работы
  - c) среднее время работы
  - d) асимптотически точное время работы
6. Если при удвоении количества элементов время работы алгоритма ( $T(N)$ ) растёт на постоянную величину, то данный алгоритм относится к классу
- a) логарифмических ( $T(N) = O(\log N)$ )
  - b) линейных ( $T(N) = O(N)$ )
  - c) экспоненциальных ( $T(N) = O(2^N)$ )
  - d) с постоянным временем выполнения ( $T(N) = O(1)$ )
7. Если при увеличении количества элементов на единицу время работы алгоритма ( $T(N)$ ) растёт в два раза, то данный алгоритм относится к классу
- a) экспоненциальных ( $T(N) = O(2^N)$ )
  - b) логарифмических ( $T(N) = O(\log N)$ )
  - c) линейных ( $T(N) = O(N)$ )
  - d) квадратичных ( $T(N) = O(N^2)$ )
8. Для рекуррентного отношения  $T(N) = T(N - 1) + 1$ ,  $T(1) = 1$  наиболее точной верхней оценкой является
- a)  $T(N) = O(N)$
  - b)  $T(N) = O(1)$
  - c)  $T(N) = O(\log N)$
  - d)  $T(N) = O(N \log N)$
9. Для рекуррентного отношения  $T(N) = 4T(N/2) + N \log N$ ,  $T(1) = 1$  наиболее точной асимптотической оценкой является
- a)  $T(N) = \Theta(N^2)$
  - b)  $T(N) = \Theta(N^3)$
  - c)  $T(N) = \Theta(N \log N)$
  - d)  $T(N) = \Theta(N^2 \log N)$
10. Для рекуррентного отношения  $T(N) = 3T(N/2) + N^2$ ,  $T(1) = 1$  наиболее точной асимптотической оценкой является
- a)  $T(N) = \Theta(N^2)$
  - b)  $T(N) = \Theta(N^3)$

$$c) T(N) = \Theta(N \log N)$$

$$d) T(N) = \Theta(N^2 \log N)$$

## Лабораторная работа № 2

**Цель** – изучить вопросы применения основных методов построения алгоритмов: метода «разделяй и властвуй» и динамического программирования.

### Ключевые понятия

Основные методы построения рекурсивных алгоритмов. Метод «разделяй и властвуй». Динамическое программирование (нисходящий и восходящий методы), принципы оптимальности Беллмана.

### Задания для самостоятельной работы

1. Сравните время вычисления 35-го числа Фибоначчи при помощи формулы Бине, итерационной формулы, метода «разделяй и властвуй», метода нисходящего динамического программирования, метода восходящего динамического программирования.
2. а). Опишите алгоритм нахождения максимального значения в массиве, используя принцип «разделяй и властвуй».  
б). Опишите алгоритм вычисления функции  $a^N$ ,  $N$  — целое положительное число, используя принцип «разделяй и властвуй».  
в). Напишите реализацию задачи о рюкзаке методом «разделяй и властвуй».  
г). Напишите реализацию задачи о рюкзаке методом восходящего динамического программирования.
3. Решите задачу о Ханойской башне.
4. На вершине лесенки, содержащей  $N$  ступенек, находится мячик, который начинает прыгать по ним вниз, к основанию. Мячик может прыгнуть на следующую ступеньку, на ступеньку через одну или через 2. (То есть, если мячик лежит на 8-ой ступеньке, то он может переместиться на 5-ую, 6-ую или 7-ую.). Напишите программу, которая определяет число всевозможных "маршрутов" мячика с вершины на

землю. Максимальная высоты лесенки — 30 ступеней, минимальная — 1 ступень.

5. На квадратной доске расставлены целые неотрицательные числа. Черепашка, находящаяся в левом верхнем углу, мечтает попасть в правый нижний. При этом она может переползать только в клетку справа или снизу и хочет, чтобы сумма всех чисел, оказавшихся у нее на пути, была бы максимальной. Напишите программу, которая определяет эту сумму.

*Указания.*

- 1). Используйте метод динамического программирования
- 2). Размер доски и числа на доске считывайте из файла.
6. Напишите программу, которая находит кратчайший путь во взвешенном графе (алгоритм Беллмана-Форда).

*Указания.*

- 1). Используйте метод динамического программирования
- 2). Граф считывайте из файла.

### **Тестовые задания**

1. «Мемуаризация» – это название метода
  - a) нисходящего динамического программирования
  - b) восходящего динамического программирования
  - c) «разделяй и властвуй»
  - d) «прямой прогонки»
2. Оценка времени решения дискретной задачи о рюкзаке (где  $M$  – размер рюкзака,  $N$  – число видов предметов) методом ДП равна
  - a)  $T(N) = O(N + M)$
  - b)  $T(N) = O(NM)$
  - c)  $T(N) = O(N \log M)$
  - d)  $T(N) = O(M \log N)$
3. В основу метода ДП положен принцип оптимальности
  - a) Беллмана
  - b) Кормена
  - c) Кнута
  - d) Вирта

4. Задача о Ханойской башне решается за время
- a)  $O(2^N)$       b)  $O(N^2)$       c)  $O(N^3)$       d)  $O(N^2 \log N)$
5. Сколько перемещений необходимо выполнить для перестановки  $N$  дисков в задаче о Ханойской башне?
6. Оценка времени нахождения  $N$ -ного числа Фибоначчи по формуле Бине равна
- a)  $T(N) = O(1)$       c)  $T(N) = O(N \log N)$   
b)  $T(N) = O(N^2)$       d)  $T(N) = O(N)$
7. Наименее эффективным методом нахождения  $N$ -ного числа Фибоначчи является
- a) метод «разделяй и властвуй»  
b) метод восходящего ДП  
c) метод нисходящего ДП  
d) применение формулы Бине
8. Если оптимизационная задача имеет перекрывающиеся подзадачи, то наиболее эффективным будет использование
- a) метода «разделяй и властвуй»  
b) метода восходящего ДП  
c) метода нисходящего ДП  
d) жадного алгоритма
9. Наиболее эффективным методом решения непрерывной задачи о рюкзаке является
- a) метод «разделяй и властвуй»  
b) метод восходящего ДП  
c) метод нисходящего ДП  
d) жадный алгоритм
10. Задача об оптимальной триангуляции выпуклого многоугольника считается классической задачей
- a) метода «разделяй и властвуй»

*b) метода ДП*

*с) жадного алгоритма*

## ТЕМА 2. СТРУКТУРЫ ДАННЫХ

### Лабораторная работа № 3

**Цель** – рассмотреть основные абстрактные типы данных (АТД): линейный список, стек, очередь; проанализировать их реализацию с помощью указателей и массивов.

#### Ключевые понятия

Концепция АТД. Представление АТД в виде структуры данных. Линейные структуры данных: линейный список, стек, очередь, очередь с приоритетами, дек. Основные операции, представление и реализации. Применение структур данных. Метод исключения рекурсии с помощью стека.

#### Задания для самостоятельной работы

1. Смоделируйте ситуацию обслуживания клиентов в банке. Клиенты, случайным образом помещаются в одну из М очередей обслуживания. Случайным образом выбирается одна из очередей, и клиент обслуживается (выбывает из очереди). При каждой операции указывайте номер добавленного клиента, номер обслуженного клиента, и состояние очередей.
2. Реализуйте АТД «Очередь с приоритетами».
  - а). с приоритетным включением;
  - б) с приоритетным исключением.
3. Напишите программу для решения задачи о Ханойской башне без использования рекурсии.

#### Тестовые задания

1. Доступ к элементам стека осуществляется по принципу
  - a) *SIFO*
  - b) *LIFO*
  - c) *LISO*
  - d) *FIFO*
4. Доступ к элементам очереди осуществляется по принципу
  - a) *LISO*
  - b) *LIFO*
  - c) *FILO*
  - d) *FIFO*
5. Дек с ограниченным выходом позволяет:

- a) добавлять элементы только в один конец, а удалять – с обоих концов
- b) добавлять элементы только в левый конец, а удалять – с правого
- c) добавлять элементы в оба конца, а удалять – только с одного
- d) добавлять и удалять элементы только в один конец
6. «Магазин» – так часто называют структуру данных
- a) *Стек* c) *Дек*
- b) *Очередь* d) *Линейный список*
7. Для исключения рекурсии применяются
- a) *дек с ограниченным выходом* c) *дек*
- b) *стек* d) *очередь*
8. При наиболее эффективной реализации очередей с приоритетами операции вставки и удаления выполняются за время
- a)  $O(n/2)$  b)  $O(n)$  c)  $O(\log n)$  d)  $O(1)$

#### Лабораторная работа № 4

**Цель** – рассмотреть АДД дерево, области применения деревьев; проанализировать математические свойства бинарных деревьев.

#### Ключевые понятия

Нелинейные структуры данных: деревья. Основные определения. Ориентированные деревья, упорядоченные деревья, бинарные деревья, m-арные деревья. Основные математические свойства бинарных деревьев. Преобразование упорядоченных деревьев в бинарные. АДД деревья. Основные операции, представление в памяти. Обходы деревьев. Применение деревьев. Деревья Хаффмана.

#### Упражнения

1. Сколько путей длины 3 существует в дереве, показанном на рисунке 1?

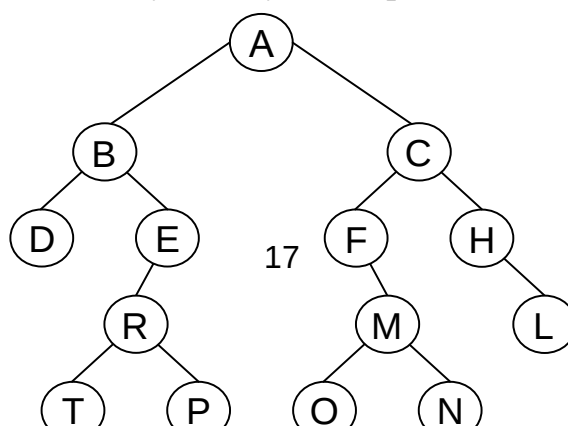


Рис. 1. Пример дерева

Ответ. В дереве существует 7 путей длины 3:

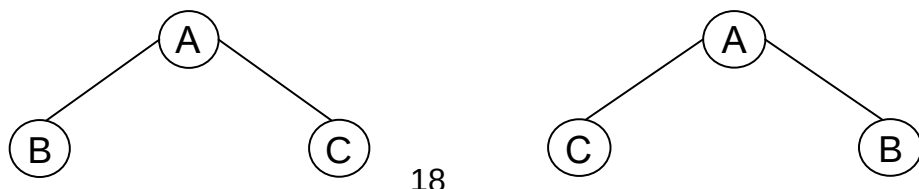
A B E R; B E R T; B E R P; A C F M; A C H L; C F M N; C F M O.

2. В какой последовательности будут перечислены узлы дерева, представленного на рисунке 1, при инфиксном, префиксном, постфиксном, поуровневом обходах?

Ответ. При инфиксном обходе узлы дерева посещаются в соответствии с правилом: «обойти левое поддерево, посетить корень, обойти правое поддерево». Следовательно, порядок узлов будет таким: D B T R P E A F O M N C H L. При префиксном обходе узлы дерева посещаются в соответствии с правилом: «посетить корень, обойти левое поддерево, обойти правое поддерево». Следовательно, порядок узлов будет таким: A B D E R T P C F M O N H L. При постфиксном обходе узлы дерева посещаются в соответствии с правилом: «обойти левое поддерево, обойти правое поддерево, посетить корень». Следовательно, порядок узлов будет таким: D T P R E B O N M F L H C A. При поуровневом обходе узлы дерева посещаются в соответствии с правилом «слева-направо, сверху-вниз». Следовательно, порядок узлов будет таким: A B C D E F H R M L T P O N.

3. Сколько различных ориентированных деревьев можно создать из трёх узлов?

Ответ. Ориентированное дерево – это дерево, в котором неважен относительный порядок поддеревьев, то есть деревья, представленные на рисунке 2, одинаковые.

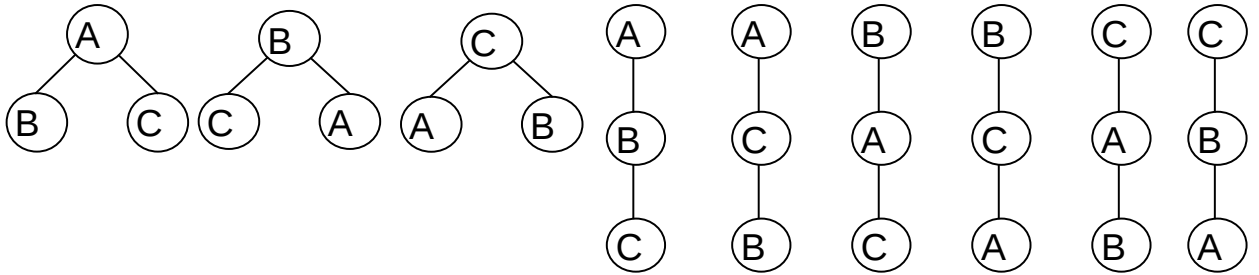


(a)

(б)

Рис. 2. Два одинаковых ориентированных дерева

Из трёх узлов можно построить 9 таких деревьев:



4. Чему равна длина внешнего пути расширенного дерева, представленного на рисунке?

Ответ. Длина внешнего пути дерева – сумма уровней всех внешних узлов расширенного дерева. Данному бинарному дереву будет соответствовать расширенное дерево, представленное на рисунке 3 (внешние узлы обозначены квадратами).

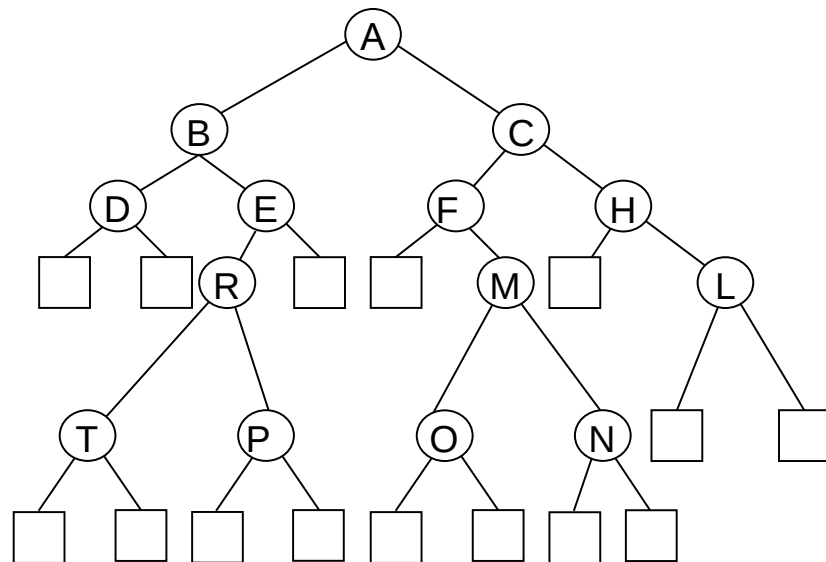


Рис. 3. Расширенное дерево

В этом дереве пять внешних узлов находятся на третьем уровне, два – на четвёртом, восемь – на пятом. Следовательно, длина внешнего пути расширенного дерева равна:  $5 \cdot 3 + 2 \cdot 4 + 8 \cdot 5 = 63$ .

5. Преобразуйте выражение  $((a+b) + c * (d+e) + f) * (g+h)$  в постфиксную

форму.

Ответ. На рисунке 4 показано дерево, построенное для выражения  $((a+b) + c * (d+e) + f) * (g+h)$ .

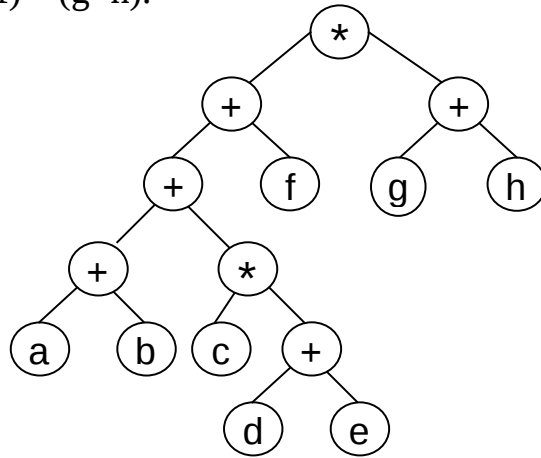
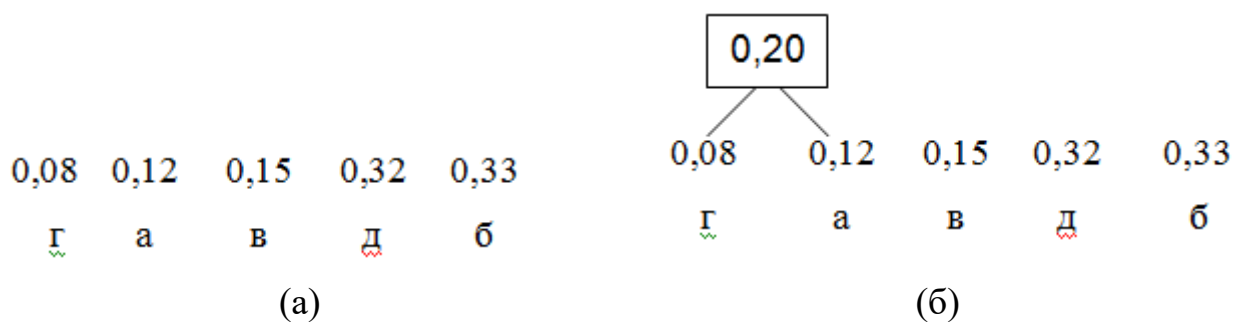


Рис. 4. Представление алгебраического выражения

При постфиксном обходе (левое поддереву – правое поддереву – узел) получаем последовательность:  $a\ b\ +\ c\ d\ e\ +\ *\ +\ f\ +\ g\ h\ +\ *$ .

6. Пусть символы а, б, в, г, д имеют вероятности появления соответственно 0,12; 0,33; 0,15; 0,08; 0,32. Какова средняя длина оптимального кода Хаффмана? Какой объём закодированного сообщения, если в исходном сообщении было 200 символов?

Ответ. Последовательные шаги выполнения алгоритма Хаффмана для символов а, б, в, г, д и их вероятностей представлены на рисунке 5.



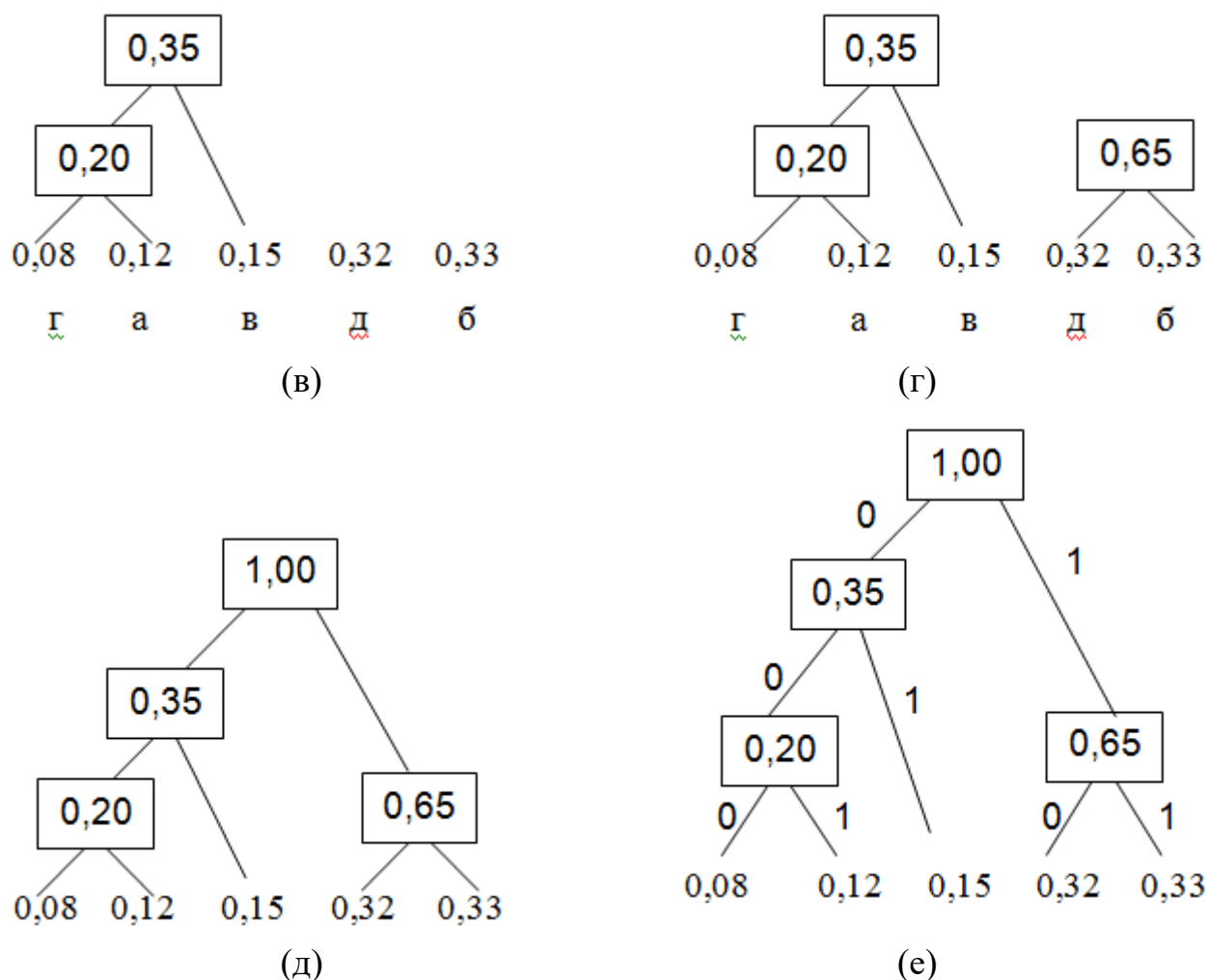


Рис. 5. Этапы построения дерева Хаффмана:

(а) – исходная ситуация; (б) – слияние г с а; (в) – слияние г, а с в; (г) – слияние д с б;  
 (д) – слияние г, а, в, д, б; (е) – дерево, представляющее оптимальный код

Из полученного дерева видно, что символы а, б, в, г, д имеют соответственно коды 001, 11, 01, 000, 10. Средняя длина кода равна

$$(3 + 2 + 2 + 3 + 2)/5 = 12/5 = 2,4$$

Объём закодированного сообщения равен:

$$0,12 \cdot 200 \cdot 3 + 0,33 \cdot 200 \cdot 2 + 0,15 \cdot 200 \cdot 2 + 0,08 \cdot 200 \cdot 3 + 0,32 \cdot 200 \cdot 2 = 440.$$

### Задания для самостоятельной работы

1. С помощью метода математической индукции докажите математических свойства бинарных деревьев.
2. Пусть символы а, б, в, г, д, е встречаются в тексте с частотами, указанными в вашем варианте соответственно. Найдите оптимальный код Хаффмана и нарисуйте соответствующее ему дерево. Какова

средняя длина кода?

| вариант | 1.                                                | 2.                                                | 3.                                                | 4.                                                | 5.                                                | 6.                                                | 7.                                                | 8.                                                | 9.                                                | 10.                                               | 11.                                               | 12.                                               |
|---------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|
| частота | 0,17;<br>0,19;<br>0,02;<br>0,25;<br>0,11;<br>0,26 | 0,05;<br>0,09;<br>0,32;<br>0,18;<br>0,21;<br>0,15 | 0,31;<br>0,14;<br>0,29;<br>0,07;<br>0,04;<br>0,15 | 0,05;<br>0,26;<br>0,06;<br>0,35;<br>0,11;<br>0,17 | 0,03;<br>0,07;<br>0,09;<br>0,28;<br>0,35;<br>0,18 | 0,21;<br>0,25;<br>0,28;<br>0,05;<br>0,06;<br>0,15 | 0,37;<br>0,11;<br>0,13;<br>0,15;<br>0,20;<br>0,04 | 0,27;<br>0,19;<br>0,18;<br>0,06;<br>0,09;<br>0,21 | 0,11;<br>0,13;<br>0,14;<br>0,21;<br>0,28;<br>0,13 | 0,23;<br>0,11;<br>0,16;<br>0,17;<br>0,05;<br>0,28 | 0,23;<br>0,09;<br>0,02;<br>0,15;<br>0,31;<br>0,20 | 0,06;<br>0,04;<br>0,25;<br>0,16;<br>0,41;<br>0,08 |

3. Напишите рекурсивную процедуру построения идеально сбалансированного дерева для произвольных  $N$  узлов, хранящихся в файле. Для полученного дерева: а) определите высоту, б) определите число уровней, с) определите, сколько путей длины три в этом дереве (для каждого пути перечислите узлы, входящие в путь), d) постройте расширенное дерево, е) найдите длину внешнего и внутреннего пути расширенного дерева.

4. Напишите рекурсивную процедуру подсчета листьев в дереве из  $N$  узлов.

5. Напишите процедуры нерекурсивных обходов бинарного дерева:

а) префиксного; б) инфиксного; с) постфиксного; d) поуровневого.

Указание: для поуровневого обхода рекомендуется использовать FIFO-очередь, для остальных обходов — стек.

6. Напишите процедуру вычисления арифметического выражения в дереве, используя постфиксный обход.

7. Пусть в арифметическом выражении в качестве терминалов используются не только цифры, но и латинские буквы 'a', 'b', ..., 'x', играющие роль переменных. Постройте дерево для арифметического выражения и опишите процедуру, которая:

а) (для вариантов 2, 5, 8, 11) упростит дерево, заменяя в нем все поддеревья, соответствующие выражениям  $(f + 0)$ ,  $(0 + f)$ ,  $(f - 0)$ ,  $(f * 1)$ ,  $(1 * f)$ , на поддеревья, соответствующие выражению  $f$ , а поддеревья, соответствующие выражениям  $(f * 0)$ ,  $(0 * f)$ , на

вершину с 0;

- b) (для вариантов 1, 4, 7, 10) преобразует дерево, заменяя в нем все поддеревья, соответствующие выражениям  $((f1 \pm f2) * f3)$  и  $(f1 * (f2 \pm f3))$ , на поддеревья, соответствующие выражениям  $((f1 * f3) \pm (f2 * f3))$  и  $((f1 * f2) \pm (f1 * f3))$ ;
- c) (для вариантов 3, 6, 9, 12) выполните преобразования, обратные преобразованиям из задачи b).

### Тестовые задания

1. Сколько различных бинарных деревьев можно построить из трёх узлов?  
a) 18                      b) 9                      c) 30                      d) 27
2. Сколько структурно различных ориентированных деревьев можно построить из трёх узлов?  
a) 3                      9                      b) 2                      c) 1
3. Если узел A имеет трёх братьев, а узел B является родителем узла A, то степень узла B равна  
a) 4                      b) 3                      c) 5                      d) 2
4. Пусть для выражения  $((a+b) + c * (d+e)) * h$  построено дерево. Какому обходу соответствует последовательность  $a b + c d e + * + h *$ ?  
a) *инфиксному*                      c) *поуровневому*  
b) *префиксному*                      D) *постфиксному*

## ТЕМА 3. АЛГОРИТМЫ ПОИСКА

### Лабораторная работа № 5

**Цель** – проанализировать эффективность алгоритмов поиска, вставки и удаления в линейных и нелинейных таблицах; разобрать структуру данных для внешнего поиска, проанализировать эффективность алгоритмов вставки, удаления и поиска для этой структуры.

#### Ключевые понятия

Постановка задачи, основные понятия. АД таблица. Поиск в линейных таблицах. Анализ эффективности алгоритмов. Поиск в нелинейных таблицах. Бинарные деревья поиска (BST). Основные операции. Анализ эффективности алгоритмов. Сбалансированные (АВЛ) деревья. Критерий сбалансированности. Деревья Фибоначчи. Виды балансировки. Основные операции. Анализ эффективности алгоритмов.

#### Упражнения

1. За какое время в худшем случае выполняется поиск в линейной неупорядоченной таблице размера  $N$ ?

Ответ. Поиск в линейной неупорядоченной таблице размера  $N$  можно произвести только с помощью алгоритма последовательного поиска, временная сложность которого равна  $O(N)$ . Этот алгоритм относится к классу линейных, то есть при увеличении размера линейной неупорядоченной таблицы в два раза следует ожидать двукратного увеличения времени поиска.

2. За какое время в худшем случае выполняется алгоритм бинарного поиска в линейной упорядоченной таблице из  $N$  элементов?

Ответ. При поиске в линейной упорядоченной таблице из  $N$  элементов с помощью алгоритма бинарного поиска количество обрабатываемых записей на каждом шаге сокращается вдвое, то есть время работы алгоритма определяется рекуррентным отношением:

$$T(N) = \begin{cases} 1 & \text{если } N = 1 \\ T(N/2) + 1 & \text{иначе} \end{cases}$$

Это рекуррентное отношение асимптотически точно оценивается:

$T(N) = \Theta(\log N)$ . Это означает, что алгоритм бинарного поиска является асимптотически оптимальным алгоритмом, то есть время работы в худшем и в лучшем случае совпадает.

3. За какое время в худшем случае осуществляется поиск элемента в бинарном дереве поиска из  $N$  узлов?

Ответ. Время поиска в бинарном дереве поиска зависит от его высоты  $h$ , т. е.  $T(N) = O(h)$ . В худшем случае высота в бинарном дереве поиска из  $N$  элементов равна  $N$  (в этом случае говорят, что дерево поиска вырожденное). Следовательно, поиск в таком дереве осуществляется за время  $O(N)$ . В лучшем случае высота в бинарном дереве поиска из  $N$  элементов равна  $\log N$ , и время поиска имеет временную сложность  $O(\log N)$ .

4. Предположим, что в бинарном дереве поиска хранятся числа от 1 до 1000, нужно найти число 363. Какая из следующих последовательностей не может быть последовательностью просматриваемых при этом ключей?

- a) 925, 202, 911, 240, 912, 245, 363
- b) 2, 252, 401, 398, 330, 344, 397, 363
- c) 924, 220, 911, 244, 898, 258, 362, 363
- d) 2, 399, 387, 219, 266, 382, 381, 278, 363

Ответ. Поиск в бинарном дереве осуществляется по следующему алгоритму: сравнить искомый ключ с текущим; если искомый ключ меньше текущего, то продолжить поиск в левом поддереве, иначе – в правом. Последовательность 925, 202, 911, 240, 912, 245, 363 при поиске числа 363 не может быть последовательностью просматриваемых ключей. Порядок просмотра ключей можно представить в виде бинарного дерева поиска

(рис.6). В левом поддереве узла с ключом 911 оказался узел с ключом 912, но в бинарном дереве поиска этого быть не может.

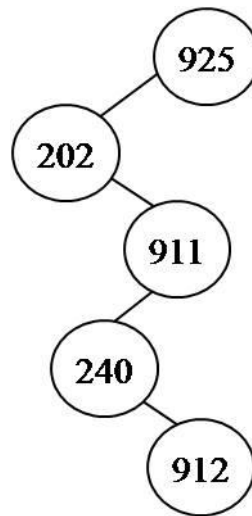
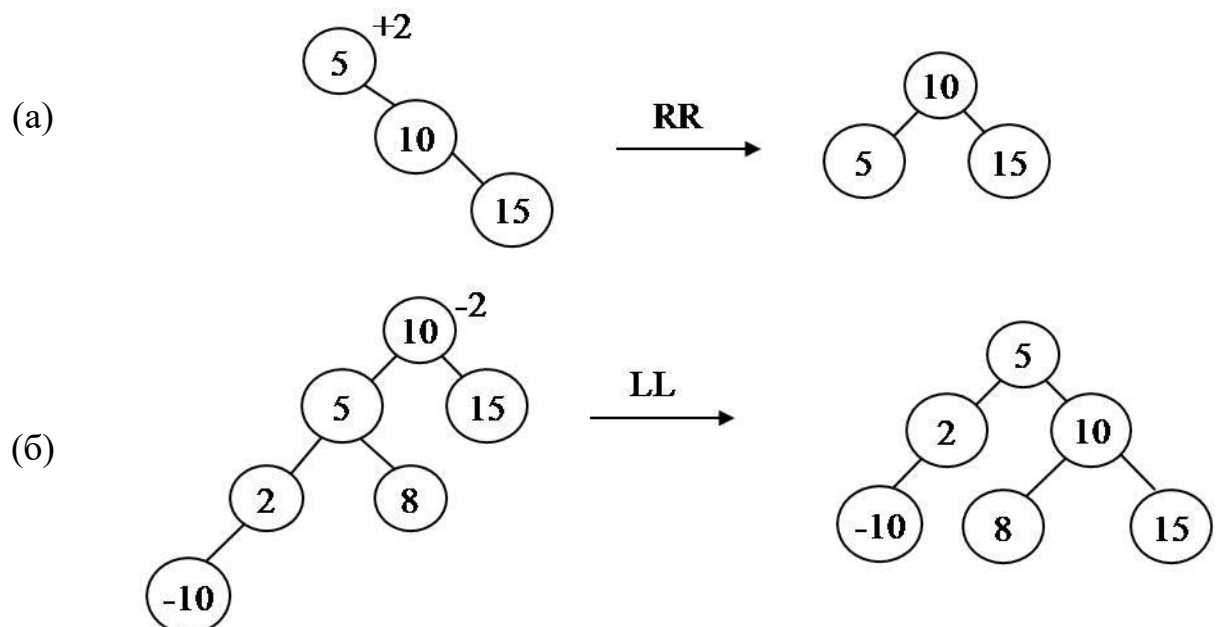


Рис. 6. Порядок просмотра ключей при поиске числа 363

5. В каком порядке будут происходить повороты при построении АВЛ дерева по элементам 5, 10, 15, 2, 8, -10, 4, 25, 12, 13?

Ответ. На рисунке 7 представлены этапы построения АВЛ дерева для ключей 5, 10, 15, 2, 8, -10, 4, 25, 12, 13.



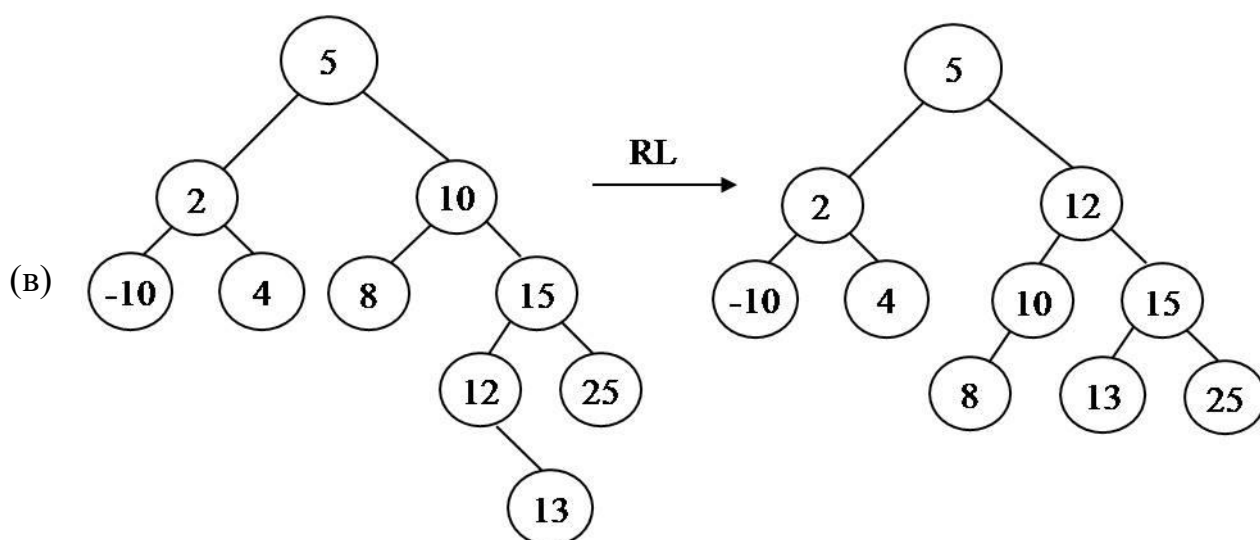


Рис. 7. Этапы построения АВЛ дерева:

(а) – добавление узла с ключом 15 вызывает RR поворот; (б) – добавление узла с ключом -10 вызывает LL поворот; (в) - добавление узла с ключом 13 вызывает LL поворот

При вставке узла с ключом 15 критерий сбалансированности дерева нарушается, значит, необходимо выполнить поворот RR. При добавлении элемента -10 нарушается баланс в узле с ключом 10 – требуется перестройка дерева с помощью поворота LL. Критерий сбалансированности нарушается при вставке узла с ключом 13, в этом случае нужно выполнить поворот RL.

6. Сколько поворотов в худшем случае требует удаление в АВЛ дереве из  $N$  узлов?

Ответ. Высота АВЛ дерева имеет порядок  $\log N$  ( $N$  – количество узлов в дереве). Удаление узла в АВЛ дереве может вызвать поворот в каждом узле вдоль пути поиска, то есть порядка  $\log N$  поворотов. Самым неудачным случаем является удаление в дереве Фибоначчи элемента из самого правого узла, которое потребует максимального количества поворотов.

### Задания для самостоятельной работы

| вариант | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | 11. | 12. |
|---------|----|----|----|----|----|----|----|----|----|-----|-----|-----|
|         | 68 | 95 | 33 | 65 | 59 | 73 | 16 | 11 | 19 | 48  | 23  | 52  |
|         | 5  | 51 | 82 | 12 | 38 | 8  | 11 | 52 | 45 | 38  | 53  | 46  |
|         | 76 | 28 | 66 | 21 | 85 | 11 | 77 | 86 | 12 | 63  | 14  | 55  |
|         | 52 | 84 | 72 | 93 | 25 | 31 | 85 | 89 | 85 | 37  | 87  | 9   |

|  |    |    |    |    |    |    |    |    |    |    |    |    |
|--|----|----|----|----|----|----|----|----|----|----|----|----|
|  | 56 | 74 | 81 | 28 | 60 | 81 | 52 | 21 | 61 | 30 | 4  | 24 |
|  | 80 | 47 | 90 | 88 | 28 | 23 | 53 | 90 | 66 | 14 | 34 | 90 |
|  | 26 | 33 | 28 | 96 | 89 | 62 | 46 | 43 | 59 | 93 | 42 | 14 |
|  | 57 | 18 | 10 | 22 | 4  | 94 | 83 | 26 | 5  | 88 | 52 | 51 |
|  | 18 | 46 | 80 | 66 | 68 | 4  | 87 | 75 | 2  | 95 | 30 | 50 |
|  | 58 | 49 | 82 | 32 | 91 | 33 | 12 | 7  | 89 | 64 | 11 | 42 |
|  | 48 | 25 | 59 | 63 | 72 | 42 | 50 | 56 | 94 | 31 | 39 | 41 |
|  | 77 | 16 | 19 | 69 | 39 | 97 | 30 | 72 | 68 | 73 | 37 | 79 |
|  | 13 | 20 | 2  | 65 | 27 | 22 | 68 | 40 | 17 | 86 | 80 | 28 |
|  | 8  | 58 | 23 | 53 | 76 | 88 | 81 | 48 | 81 | 72 | 66 | 88 |
|  | 20 | 90 | 94 | 78 | 40 | 72 | 25 | 31 | 82 | 25 | 84 | 83 |

1. Постройте бинарное дерево поиска, вставив последовательно элементы с ключами, указанными в вашем варианте.
2. Удалите из полученного дерева последовательно первые 7 элементов с ключами, указанными в вашем варианте. После каждого удаления нужно рисовать полученное дерево, если удаляемый элемент не лист.
3. Постройте AVL-дерево, вставив последовательно элементы из своего варианта в изначально пустое AVL-дерево.
4. Из полученного AVL-дерева удалите последовательно первые 7 элементов с ключами, указанными в вашем варианте. После каждого удаления нужно рисовать полученное дерево.
5. Подберите последовательность элементов такую, чтобы при вставке в изначально пустое AVL-дерево выполнялись бы все четыре поворота. Какова минимальная длина такой последовательности?
6. Постройте дерево Фибоначчи порядка 8. Удалите самый правый элемент.

### Тестовые задания

1. Для того, чтобы напечатать узлы бинарного дерева поиска в порядке возрастания нужно применить
 

|                      |                      |
|----------------------|----------------------|
| а) инфиксный обход   | с) префиксный обход  |
| б) постфиксный обход | д) поуровневый обход |

2. Длина внешнего пути бинарного дерева поиска, построенного для чисел 10, 15, 12, 14, 5, 8 равна
- a) 28                      b) 21                      c) 9                      d) 7
3. В худшем случае вставка в бинарное дерево поиска из  $N$  узлов осуществляется за время
- a)  $O(N)$                       c)  $O(1)$   
b)  $O(\log N)$                       d)  $O(N \log N)$
4. Минимальная длина последовательности элементов, при построении АВЛ дерева по которой выполнялись бы все четыре поворота, равна
- a) 7                      b) 6                      c) 8                      d) 9
5. При построении АВЛ дерева по элементам 100, 38, 61, 55, 120, 110, 150, 118, 160, 13 повороты будут происходить в порядке
- a)  $RL, LR, RL$                       c)  $LL, LR, RR$   
b)  $LR, RL, RR$                       d)  $RR, RL, LR$

### **Лабораторная работа № 6**

**Цель** – проанализировать эффективность алгоритмов поиска, вставки и удаления в таблицах с вычисляемыми входами.

#### **Ключевые понятия**

Поиск в таблицах с вычисляемыми входами. Хеширование. Основные методы вычисления хеш-функций: метод деления, метод умножения, комбинированный метод. Разрешение коллизий. Хеширование с цепочками. Хеширование открытой адресацией.

Основные виды повторного хеширования: линейное исследование, квадратичное исследование, двойное хеширование. Основные операции. Анализ эффективности алгоритмов.

#### **Упражнения**

1. При хешировании с цепочками, за какое время в худшем случае выполняется процедура поиска элемента в хеш-таблице?

Ответ. При хешировании с цепочками в худшем случае, когда все хеш-адреса одинаковы, элементы таблицы попадают в один список длины  $N$ , и операция поиска сводится к последовательному поиску в списке. Следовательно, поиск в худшем случае будет выполняться за время  $O(N)$ .

2. При вычислении хеш-функции по методу деления ( $h(K) = K \bmod M$ , где  $K$  – ключ), каким должно быть число  $M$  для наиболее эффективного распределения ключей по таблице?

Ответ. При вычислении хеш-функции по методу деления эффективность равномерного распределения ключей по таблице зависит от  $M$ . Например, для  $M$ , являющегося степенью двойки, значения хеш-функции представляют собой правые биты двоичного представления ключа. Аналогично для  $M$ , являющегося степенью десятки, значения хеш-функции – правые цифры в десятичном представлении ключа. В этом случае оказывается, что только часть цифр ключа полностью определяет хеш-адрес, что может плохо повлиять на равномерность распределения. Наиболее эффективно хеш-функция работает, когда  $M$  – простое число, т. к. это приводит к меньшему числу коллизий.

### **Задания для самостоятельной работы**

3. Дано множество ключей и хеш-функция. Является ли данная хеш-функция совершенной?

| вариант | множество ключей                          | хеш-функция               |
|---------|-------------------------------------------|---------------------------|
| 1.      | 81, 129, 301, 38, 434, 216, 412, 487, 234 | $h(x) = (x + 18) \div 63$ |
| 2.      | 95, 33, 65, 59, 73, 16, 116, 192, 48      | $h(x) = (x + 15) \div 43$ |
| 3.      | 51, 82, 212, 381, 68, 121, 52, 45, 38     | $h(x) = (x + 25) \div 59$ |
| 4.      | 28, 66, 21, 85, 117, 77, 86, 312, 63      | $h(x) = (x + 21) \div 61$ |
| 5.      | 84, 72, 93, 25, 31, 85, 89, 185, 37       | $h(x) = (x + 13) \div 83$ |
| 6.      | 74, 81, 28, 60, 181, 52, 21, 61, 30       | $h(x) = (x + 17) \div 37$ |
| 7.      | 47, 90, 88, 28, 23, 53, 290, 66, 145      | $h(x) = (x + 19) \div 97$ |

|     |                                        |                           |
|-----|----------------------------------------|---------------------------|
| 8.  | 33, 28, 96, 89, 62, 46, 43, 59, 93     | $h(x) = (x + 23) \div 53$ |
| 9.  | 184, 101, 22, 415, 94, 83, 26, 345, 88 | $h(x) = (x + 16) \div 49$ |
| 10. | 46, 81, 66, 68, 413, 87, 75, 246, 95   | $h(x) = (x + 22) \div 67$ |
| 11. | 49, 82, 32, 91, 33, 127, 371, 89, 64   | $h(x) = (x + 31) \div 73$ |
| 12. | 25, 59, 63, 72, 42, 50, 56, 94, 31     | $h(x) = (x + 15) \div 87$ |

1. Дан набор символьных ключей «Иван», «Пётр», «Варвара», «Александр», «Анна», «Алла». Придумайте совершенную хеш-функцию для 7-элементной таблицы.
2. Предположим, что размер хеш-таблицы равен  $M$ , в качестве хеш-функции используется  $h(x) = x \bmod M$ . Постройте хеш-таблицу при вставке ключей из своего варианта. Используйте:
  - а) хеширование с цепочками;
  - а) хеширование с открытой адресацией (линейное);
  - б) хеширование с открытой адресацией (квадратичное);
  - с) двойное хеширование.

При каком способе хеширования получилось меньше всего коллизий?

| вариант | множество ключей                            | М  | вариант | множество ключей                   | М |
|---------|---------------------------------------------|----|---------|------------------------------------|---|
| 1.      | 1, 8, 27, 64, 125, 216, 343                 | 7  | 2.      | 47, 90, 88, 28, 23, 53, 290        | 7 |
| 3.      | 95, 33, 65, 59, 73, 16, 3, 116, 192, 48, 21 | 11 | 4.      | 33, 28, 96, 89, 62, 46, 43, 59, 93 | 9 |
| 5.      | 51, 82, 212, 381, 68, 121, 52               | 7  | 6.      | 184, 101, 22, 415, 94, 83, 26      | 7 |
| 7.      | 28, 66, 21, 85, 117, 77, 86, 312, 63        | 9  | 8.      | 46, 81, 66, 68, 413, 87, 75, 246   | 8 |
| 9.      | 84, 72, 93, 25, 31, 85, 89, 185             | 8  | 10.     | 49, 82, 32, 91, 33, 127, 371       | 7 |
| 11.     | 74, 81, 28, 60, 181, 52, 21                 | 7  | 12.     | 25, 59, 63, 72, 42, 50, 56         | 7 |

3. Напишите программу, которая с использованием хеширования с цепочками вставляет  $N$  случайных целых чисел в таблицу размером  $N/100$ , а затем определяет длину самого короткого и самого длинного списка ( $N = 5000, 10000, 20000$ ).
4. Напишите программу, которая с использованием хеширования с

линейное открытой адресацией вставляет  $N/2$  случайных целых чисел в таблицу размером  $N$ , а затем определяет среднее количество проб при неудачном поиске в таблице ( $N = 5000, 10000, 20000$ ).

5. Напишите хеш-функцию, реализующую метод свёртки, который заключается в разбиении ключа на несколько частей, которые затем комбинируются так, чтобы получилось наименьшее число. Полученное значение используется в виде значения хеш-функции или уменьшается ещё раз. Например, ключ 523456795 можно представить в виде  $523 + 456 + 795 = 1774$ .

### Тестовые задания

1. При вычислении хеш-функции по методу деления ( $h(K) = K \bmod M$ , где  $K$  – ключ) наиболее эффективное распределение ключей по таблице получается, если
  - a)  $M$  – простое число
  - b)  $M$  – степень двойки
  - c)  $M$  – степень десятки
  - d)  $M$  – нечётное число
2. Наиболее эффективным является
  - a) двойное хеширование
  - b) хеширование линейной открытой адресацией
  - c) квадратичное хеширование
  - d) вычисление хеш-функции методом деления
3. При хешировании с цепочками в худшем случае процедура поиска элемента в хеш-таблице выполняется за время
  - a)  $O(1)$
  - b)  $O(n)$
  - c) пропорциональное коэффициенту заполнения таблицы
  - d)  $O(\log \log n)$
4. При хешировании с цепочками в среднем процедура поиска элемента в

хеш-таблице выполняется за время

- a) пропорциональное коэффициенту заполнения таблицы
- b)  $O(n)$
- c)  $O(1)$
- d)  $O(\log \log n)$

5. При хешировании с цепочками в лучшем случае процедура поиска элемента в хеш-таблице выполняется за время

- a)  $O(1)$
- b)  $O(n)$
- c) пропорциональное коэффициенту заполнения таблицы
- d)  $O(\log \log n)$

6. Количество коллизий при вычислении хеш-функции по методу умножения ( $h(K) = \text{целая часть}(M(K\alpha \bmod 1))$ , где  $\alpha$  – константа от 0 до 1,  $K\alpha \bmod 1$  – дробная часть произведения  $K\alpha$ )) уменьшается

- a) если  $\alpha$  – «золотое сечение»
- b) если  $M$  – степень двойки
- c) если  $\alpha$  – число «пи», делённое на десять
- d) если  $M$  – простое число

7. В предположении равномерного хеширования математическое ожидание числа проб при добавлении нового элемента в таблицу с открытой адресацией не превосходит

- a)  $1/(1 - a)$ , где  $a < 1$  – коэффициент заполнения таблицы
- b)  $\frac{1}{a} \ln \frac{1}{1-a}$ , где  $a < 1$  – коэффициент заполнения таблицы
- c)  $a \ln \frac{1}{1-a}$ , где  $a < 1$  – коэффициент заполнения таблицы
- d)  $(1 - a)$ , где  $a < 1$  – коэффициент заполнения таблицы

8. Процедура вставки в хеш-таблице выполняется за время

- a)  $O(\log n)$
- b)  $O(n)$

- c)  $O(1)$
- d)  $O(\log \log n)$

9. Процедура поиска в хеш-таблице выполняется за время

- a)  $O(\log n)$
- b)  $O(n)$
- c)  $O(\log \log n)$
- d)  $O(1)$

10. Процедура удаления в хеш-таблице выполняется за время

- a)  $O(\log n)$
- b)  $O(1)$
- c)  $O(\log \log n)$
- d)  $O(n)$

### **Лабораторная работа № 7**

**Цель** – проанализировать эффективность алгоритмов поиска, вставки и удаления в нелинейных таблицах, разобрать структуру данных для внешнего поиска, проанализировать эффективность алгоритмов вставки, удаления и поиска для этой структуры.

#### **Ключевые понятия**

Б-деревья. Внешний поиск. Основные операции. Анализ эффективности алгоритмов. Разновидности Б-деревьев. Применение структур данных. Красно-черные деревья. Рандомизированные деревья поиска. Оптимальные деревья поиска. Основные операции. Анализ эффективности алгоритмов.

#### **Задания для самостоятельной работы**

1. Используя рекурсивные и нерекурсивные алгоритмы, реализуйте следующие задачи для дерева поиска. Одинаковые элементы хранятся в одном узле дерева, для их счетчика предусмотрено поле Count.

Type Base = integer;  
Tree = ^ Node;

```
Node = record
    elem: Base;
    left, right: Tree
    Count: integer
end;
```

- a) По текстовому файлу  $f$ , содержащему элементы типа  $\text{Base}$  (среди которых могут быть и одинаковые), построить дерево поиска  $T$ .
  - b) Добавить к дереву поиска  $T$  новый элемент  $E$ , если его не было в  $T$ .
  - c) Проверить, входит ли элемент  $E$  в дерево поиска  $T$ .
  - d) Удалить из дерева поиска  $T$  элемент  $E$ , если он есть в  $T$ .
  - e) Записать в текстовый файл  $f$  элементы дерева поиска  $T$  в порядке их возрастания, используя обход дерева соответствующим методом. Одинаковые элементы записывать в файл в количестве  $\text{Count}$ .
2. Проведите эмпирический анализ поиска в деревьях. Напишите программу для сравнения алгоритма поиска в бинарном дереве поиска, АВЛ-дереве, красно-черном и рандомизированном дереве, каждое из которых построено при помощи вставки  $N$  случайных чисел в первоначально пустое дерево. Учтите возможность появления одинаковых чисел. Вычислите среднее значение количества сравнений в каждом из деревьев при выполнении  $N$  произвольных поисков. Прodelайте эксперименты для различных  $N$  (например,  $N = 5000, 10000, 20000$ ), а затем сравните полученные результаты.

### Тестовые задания

1. Какое утверждение является истинным?
  - a) *дерево Фибоначчи является АВЛ деревом*
  - b) *любое бинарное дерево поиска из  $N$  узлов имеет высоту  $\log_2 n$*
  - c) *АВЛ дерево является идеально сбалансированным*
  - d) *в красно-чёрном дереве максимальная длина пути равна  $n \log n$*
3. В худшем случае поиск элемента в красно-чёрном дереве из  $N$  узлов осуществляется за время

- a)  $O(\log n)$  c)  $O(1)$   
 b)  $O(n)$  d)  $O(n \log n)$
4. В лучшем случае поиск элемента в рандомизированном дереве поиска из  $N$  узлов осуществляется за время
- a)  $O(1)$  c)  $O(n)$   
 b)  $O(\log n)$  d)  $O(n \log n)$
5. Максимальное количество потомков у каждого узла Б-дерева, содержащего не менее 3 и не более 6 элементов, равно:
- a) 7 b) 6 c) 5 d) 4
6. Удаление в АВЛ дереве из  $N$  узлов требует в худшем случае
- a)  $O(\log n)$  поворотов c)  $O(n \log n)$  поворотов  
 b)  $O(1)$  поворотов d)  $O(n)$  поворотов
7. Поиск элемента на странице Б-дерева в среднем выполняется за время
- a)  $O(n)$  c)  $O(1)$   
 b)  $O(\log n)$  d)  $O(n \log n)$
8. В лучшем случае поиск элемента в красно-чёрном дереве из  $N$  узлов осуществляется за время
- a)  $O(\log n)$  b)  $O(n)$  c)  $O(n \log n)$  d)  $O(1)$
9. Страница Б-дерева, содержащая  $m$  элементов, имеет
- a)  $2^m$  потомков c)  $2^m + 1$  потомков  
 b)  $m + 1$  потомков d)  $2^m - 1$  потомков
10. Какое утверждение не является истинным?
- a) поиск быстрее осуществляется в красно-чёрных деревьях, чем в АВЛ дереве  
 b) красно-чёрное дерево – вырожденный случай Б-дерева  
 c) вставка быстрее осуществляется в красно-чёрных деревьях, чем в АВЛ дереве  
 d) место элемента в рандомизированном дереве поиска определяет его приоритет

11. Двоичные деревья поиска из  $N$  узлов и высоты  $h$  позволяют выполнять операцию поиска за время

a)  $O(h)$

c)  $O(h^2)$

b)  $O(n * h)$

d)  $O(\log h)$

## ТЕМА 4. АЛГОРИТМЫ СОРТИРОВОК

### Лабораторная работа № 8

**Цель** – рассмотреть методы внутренней сортировки, проанализировать эффективность алгоритмов сортировки для массивов разной упорядоченности.

#### Ключевые понятия

Постановка задачи сортировки, основные определения. Понятие внутренней и внешней сортировки, устойчивость сортировки, основные характеристики эффективности. Анализ алгоритмов. Сортировка Шелла. Понятие  $h$ -сортировки, зависимость эффективности сортировки от выбора последовательности  $h$ . Алгоритм быстрой сортировки, выбор опорного элемента. Обменная поразрядная сортировка. Пирамидальная сортировка. Эффективная реализация очередей с приоритетами. Сортировка подсчётом распределения.

#### Упражнения

1. Упорядочите элементы 8, 4, 1, 9, 2, 1, 7, 4 по неубыванию следующими сортировками: быстрой, пирамидальной, слиянием, обменной поразрядной.
2. Приведите примеры, показывающие, что обменная поразрядная и пирамидальная сортировки являются неустойчивыми.
3. Напишите процедуру нахождения  $k$  наибольших элементов массива из  $N$  элементов на основе использования пирамиды.
4. Реализуйте очередь с приоритетами, используя пирамиду.
5. Сравните эффективность различных вариантов сортировок слиянием: нисходящую сортировку слиянием, восходящую сортировку слиянием, комбинированную нисходящую сортировку слиянием, комбинированную восходящую сортировку слиянием, естественную сортировку слиянием. Рассмотрите алгоритмы на упорядоченных данных, случайных, упорядоченных в обратном порядке.

6. Определите, сколько дополнительной памяти требует сортировка подсчётом распределения при упорядочивании по одному, двум, трём,  $N$  разрядам за один проход по массиву.

#### **Задания для самостоятельной работы**

1. Сравните время выполнения сортировки Шелла при различных способах выбора последовательности  $h$  для массивов с разным числом элементов. Результаты исследования оформите таблицей.
2. Изучите влияние данных, содержащих кроме ключей еще и сопутствующую информацию, на время сортировки Шелла. Результаты исследования оформите таблицей.
3. Проведите эмпирический анализ быстрой сортировки для рекурсивной, нерекурсивной и комбинированной реализаций. В комбинированной реализации используйте метод простых вставок для сортировки массивов длины, меньшей  $M$ .
4. Экспериментально сравните время выполнения быстрой, поразрядной и пирамидальной сортировки для массивов из 50000, 100000, 200000 элементов. Результаты исследования оформите таблицей.

#### **Тестовые задания**

1. Сортировка Шелла основана на сортировке
  - a) пузырьком
  - b) выбором
  - c) вставками
  - d) бинарными вставками
2. Быстрая сортировка относится к классу
  - a) обменных сортировок
  - b) сортировок посредством выбора
  - c) сортировок посредством распределения
  - d) сортировок посредством вставок
3. Алгоритм быстрой сортировки можно улучшить, если для сортировки

подмассивов небольшой длины применять

- a) сортировку простыми вставками
- b) сортировку бинарными вставками
- c) пузырьковую сортировку
- d) сортировку выбором

4. Неверным является утверждение

- a) сортировка слиянием в лучшем случае выполняется за время  $O(N^2)$
- b) пирамидальная сортировка в худшем случае выполняется за время  $O(N^2)$
- c) обменная поразрядная сортировка в лучшем случае выполняется за время  $O(N)$
- d) быстрая сортировка в худшем случае выполняется за время  $O(N^2)$

5. Не требует дополнительной памяти:

- a) пирамидальная сортировка
- b) быстрая сортировка
- c) сортировка слиянием
- d) цифровая сортировка

6. Асимптотически оптимальным методом не является

- a) быстрая сортировка
- b) пирамидальная сортировка
- c) сортировка слиянием
- d) цифровая (поразрядная по младшей цифре) сортировка

7. Быстрая сортировка относится к классу

- a) обменных сортировок
- b) сортировок посредством вставок
- c) сортировок посредством выбора
- d) сортировок посредством распределения

8. Наилучшим образом быстрая сортировка ведёт себя при выборе в качестве опорного

- a) элемента, являющегося медианой
- b) среднего элемента
- c) минимального элемента
- d) максимального элемента

### Лабораторная работа №9

**Цель** — рассмотреть алгоритм топологической сортировки (преобразование частичных порядков в линейные).

Топологическая сортировка применяется для элементов, на которых определен частичный порядок, то есть упорядочение задано не на всех элементах, а только на некоторых парах.

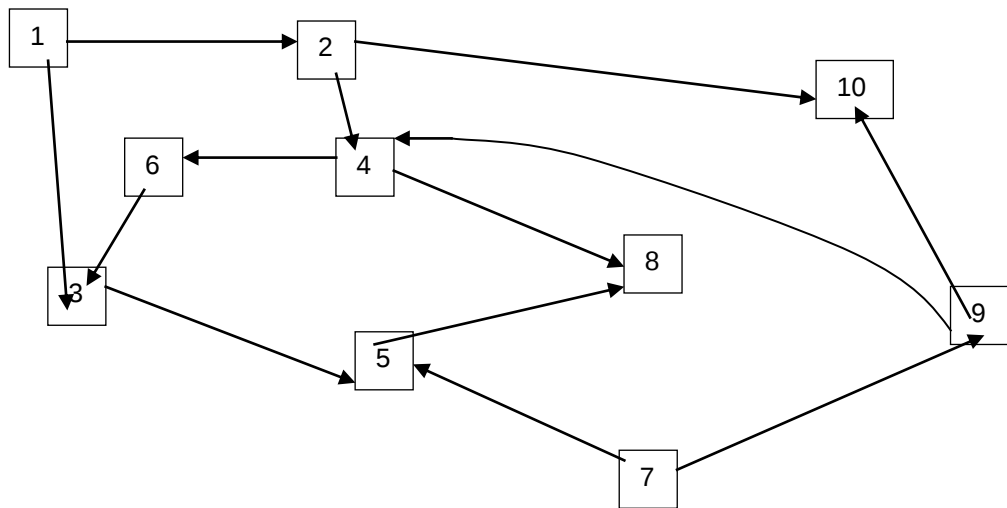
Например:

1. В словаре термин определен через другие термины. Обозначим отношением  $U \sqsubset V$ , если слово  $U$  определено через слово  $V$ .
2. В программе процедура может содержать вызов других процедур. Обозначим отношением  $U \sqsubset V$ , если процедура  $U$  вызывает процедуру  $V$ .

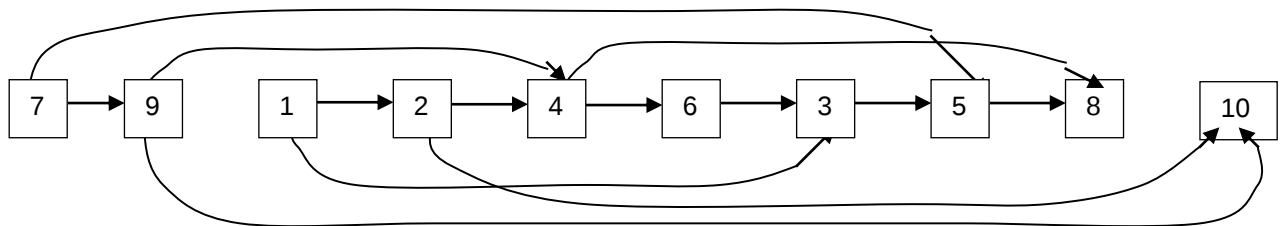
Рассмотрим пример топологической сортировки на множестве из 10 чисел:  $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ . Пусть отношение частичного порядка на множестве  $S$  задано следующими отношениями элементов:

|     |      |     |     |     |      |
|-----|------|-----|-----|-----|------|
| 1<2 | 4<6  | 4<8 | 1<3 | 5<8 | 7<9  |
| 2<4 | 4<6  | 4<8 | 1<3 | 5<8 | 7<9  |
|     | 2<10 | 6<3 | 3<5 | 7<5 | 9<4  |
|     |      |     |     |     | 9<10 |

Удобно проиллюстрировать отношение частичного порядка с помощью графа, где вершины — это элементы множества, а стрелки — отношение порядка. На графе отсутствуют циклы, что является необходимым условием частично упорядоченного множества.



Выберем элемент, которому не предшествует никакой другой – 7. Исклучим его из множества. Выберем следующий элемент, которому не предшествует никакой другой – 9, исключим его из множества. Повторяем до тех пор, пока множество не станет пустым. В результате получим линейное упорядочение вида:



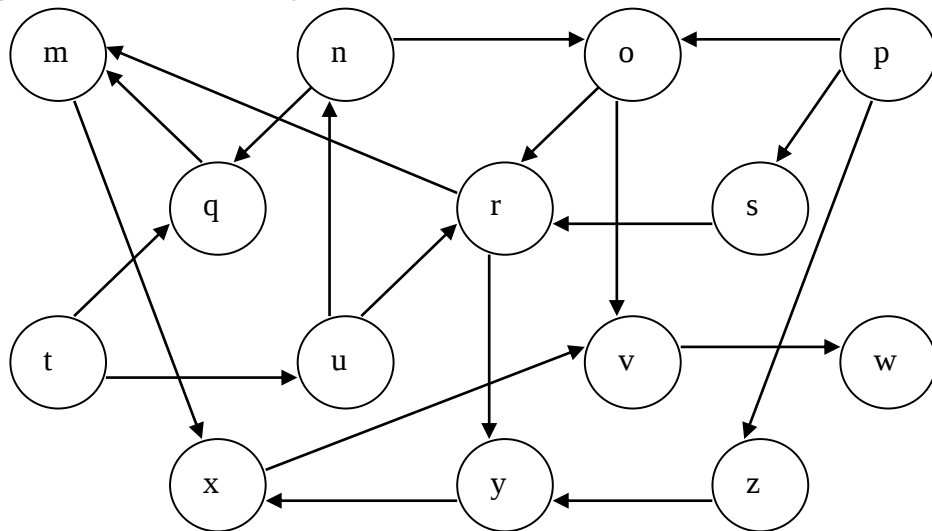
### Задания для самостоятельной работы

1. Выполните топологическую сортировку для множества  $S = \{1, 2, 3, 4, 5, 6, 7\}$  с отношениями частичного порядка:  $1 < 5, 5 < 4, 3 < 1, 3 < 5, 3 < 4, 1 < 7, 5 < 2, 4 < 2, 7 < 2, 2 < 6, 7 < 6$ .
2. Как поведёт себя алгоритм топологической сортировки, если среди частичных порядков будут одинаковые?
3. Реализуйте алгоритм топологической сортировки. *Указание:* входные данные содержаться в файле.
4. Реализуйте алгоритм топологической сортировки так, чтобы в случае, когда множество не является частично упорядоченным, программа выдавала бы последовательность элементов, образующих цикл.

### Тестовые задания

1. Сортировку слиянием можно улучшить, если для серий небольшой длины использовать
  - a) сортировку простыми вставками
  - b) сортировку бинарными вставками
  - c) быструю сортировку
  - d) пирамидальную сортировку
2. Цифровая (поразрядная по младшей цифре) сортировка будет работать правильно, если в качестве вспомогательной сортировки по цифре будет использовать
  1. устойчивую сортировку
  2. неустойчивую сортировку
  3. любую обменную сортировку
  4. любую сортировку посредством вставок
3. Чувствительной к предварительной упорядоченности массива является сортировка
  - e) пирамидальная
  - f) естественным слиянием
  - g) быстрая
  - h) поразрядная по старшей цифре
4. Не требует дополнительной памяти:
  - a) пирамидальная сортировка
  - b) быстрая сортировка
  - c) сортировка слиянием
  - d) цифровая сортировка
5. Асимптотически оптимальным методом не является
  1. быстрая сортировка
  2. пирамидальная сортировка
  3. сортировка слиянием

4. цифровая (поразрядная по младшей цифре) сортировка
6. Пирамидой является массив
- 23 14 17 6 12 7 13 5 1 10
  - 1 5 6 7 10 12 13 14 17 23
  - 1 6 5 7 12 10 14 13 23 17
  - 23 14 17 6 10 7 13 5 1 12
7. Множество  $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$  не является частично упорядоченным множеством, если на нём определены отношения
- $6 < 4$      $4 < 9$      $9 < 2$      $2 < 10$      $2 < 7$      $2 < 8$      $8 < 6$
  - $2 < 3$      $2 < 4$      $2 < 5$      $2 < 6$      $2 < 7$      $2 < 8$      $2 < 9$
  - $3 < 1$      $3 < 2$      $9 < 2$      $9 < 4$      $6 < 5$      $6 < 1$      $9 < 1$
  - $10 < 4$      $2 < 10$      $6 < 2$      $7 < 1$      $3 < 2$      $3 < 5$      $5 < 1$
8. Топологическая сортировка расположит вершины графа, представленного на рисунке, в следующем порядке
- $ptusnorqmzyxvw$
  - $pmoqxyvnuswzt$
  - $rmotqxyvnuzsp$
  - $oprtsnmuvwxyzy$



## **Лабораторная работа 10**

**Цель** — Рассмотреть методы эффективной внешней сортировки; проанализировать эффективность алгоритмов сортировки.

### **Ключевые понятия**

Сортировка слиянием. Понятие двухпутевого слияния. Нисходящая сортировка слиянием. Вопросы устойчивости. Восходящая сортировка слиянием. Сортировка естественным слиянием. Анализ алгоритмов. Реализация алгоритмов на списках.

Сбалансированное многопутевое слияние. Многофазное слияние. Анализ алгоритмов.

### **Упражнения**

1. Упорядочите элементы по неубыванию, используя алгоритм двухпутевого слияния.
2. Проведите эмпирический анализ нисходящей и восходящей сортировок слиянием.
3. Экспериментально сравните время выполнения для массивов из 50000, 100000, 200000 элементов сбалансированным многопутевым и многофазным слиянием. Результаты исследования оформите таблицей.
4. Сравните эффективность различных вариантов сортировок слиянием: нисходящую сортировку слиянием, восходящую сортировку слиянием, комбинированную нисходящую сортировку слиянием, комбинированную восходящую сортировку слиянием, естественную сортировку слиянием.
5. Рассмотрите алгоритмы на упорядоченных данных, случайных, упорядоченных в обратном порядке.

## СПИСОК ЛИТЕРАТУРЫ

### Основная литература

1. Вирт, Никлаус Алгоритмы и структуры данных / Никлаус Вирт ; перевод Ф. В. Ткачева. — 2-е изд. — Саратов: Профобразование, 2019. — 272 с. — ISBN 978-5-4488-0101-3. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <http://www.iprbookshop.ru/88753.html> (дата обращения: 15.05.2020).

### Дополнительная литература

1. Дроздов, С. Н. Структуры и алгоритмы обработки данных: Учебное пособие / Дроздов С.Н. - Таганрог: Южный федеральный университет, 2016. - 228 с.: ISBN 978-5-9275-2242-2. - Текст: электронный. - URL: <https://znanium.com/catalog/product/991928> (дата обращения: 15.05.2020).
2. 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. — ISBN 978-5-4486-0192-7. — Режим доступа: URL: <http://www.iprbookshop.ru/71591.html> (дата обращения: 15.05.2020).
3. Колдаев, В. Д. Структуры и алгоритмы обработки данных [Электронный ресурс] : учеб. пособие / В.Д. Колдаев. — Электрон. текстовые дан. - М.: ИЦ РИОР: НИЦ ИНФРА-М, 2014. — Режим доступа: <http://znanium.com/catalog.php?bookinfo=418290> (дата обращения: 15.05.2020).

### Интернет-ресурсы

1. Электронно-библиотечная система «Университетская библиотека он-лайн». - URL: <http://biblioclub.ru>
2. Электронно-библиотечная система издательства «Инфра». - URL: <http://znanium.com>.
3. eLIBRARY – Научная электронная библиотека (Москва). - URL: <http://elibrary.ru>
4. Курс «Структуры и алгоритмы компьютерной обработки данных». Портал доступа к электронным образовательным ресурсам ТюмГУ. - URL: <http://eLearning.utmn.ru> (вход по корпоративному паролю).

5. Современные профессиональные базы данных и информационные справочные системы:

Межвузовская электронная библиотека (МЭБ). - URL: <https://icdlib.nspu.ru/>

Национальная электронная библиотека. - URL: <https://rusneb.ru/>

Institute of Electrical and Electronics Engineers, Inc (IEEE). - URL: <https://ieeexplore.ieee.org/Xplore/home.jsp?reload=true>

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю)

- Лицензионное ПО:
  - Платформа для электронного обучения Microsoft Teams
  - Программное обеспечение Microsoft Imagine Academy (ранее Dreamspark): MS Visual Studio, MS SQL Server, ОС семейства MS Windows, MS Visio, MS Project
  - Программное обеспечение Microsoft Office 365
- Свободно распространяемое ПО, в том числе отечественного производства:
  - Программная платформа Moodle  
<https://docs.moodle.org/dev/License>