МИНОБРНАУКИ РОССИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА»

Институт информатики, математики и электроники

Дисциплина: «Надежность и качество ПО»

Отчёт по лабораторным работам № 1-5

# «Определение свойств случайного графа структуры ПО. Оценка числа вариантов на отладку ПО по модели структуры ПО»

Вариант № 5Б

Выполнил:

студент группы 6411-100503D

Ковалев К.А

Проверил:

Мостовой Я.А.

Самара, 2019

**Цель работы:** рассмотреть структуру ПО, определить параметр структуры, позволяющий оценить качество структуры с точки зрения трудоёмкости отладки - числа вариантов, необходимых для отладки ПО. Создать обобщенную статистическую модель структуры ПО в виде случайного дерева, разработать соответствующую программу и показать статистическую устойчивость структурного параметра, определяющего число вариантов отладки ПО.

**Работа 1.** Анализ предметной области, синтез модели структуры программы (схема программы).

Структуру работы ПО можно представить в виде графа-дерева. В каждую вершину дерева входит только одно ребро (за исключением корневого). Число ребер такого графа на единицу меньше числа вершин. Эта особенность - однозначная связь между числом узлов и числом ребер отличает деревья от других графов. Число маршрутов в дереве равно числу висячих узлов. Число путей исполнения при отладке ПО должно быть равно числу висячих узлов.

Рассмотрим дерево, в котором всего узлов P, а висячих узлов B,$m_i$ – число ребер, входящих и выходящих из i-го узла.

$$\sum_{i=1}^{P} m_i = 2(P-1)$$

Для “регулярного” дерева, для которого m=const (кроме корневого и висячих узлов)

$$\alpha = \frac{P}{B} = \frac{m-1}{m-2} - \frac{1}{(m-2)B}$$

Для больших деревьев B достаточно велико, следовательно:

$$\alpha \approx \frac{m-1}{m-2}$$

В реальном ПО $m_i \neq const$ и $\alpha$скорее всего величина случайная, но возможны средние оценки, которые могут определить трудозатраты при отладке. Случайный граф, в котором число ребер, исходящих из каждого узла случайно, более адекватная модель ПО.

При росте числа узлов в графе параметр $\alpha$ стремится к постоянной величине и является характеристикой структуры графа, важной для отладки ПО. Экспериментально можно показать, что и для случайного графа - дерева

параметр $\alpha$ для большого числа узлов стремится к постоянной величине, имеющей небольшой случайный разброс.

**Работа 2.** Построить случайный граф ПО, для которого число выходящих из вершины ребер определяется датчиком случайных чисел. Определить значение структурного параметра α= число всех узлов/число висячих узлов. Привести гистограмму для полученных при построении графа значений (m-1). Убедится в правильности построения гистограммы.

m=5, N = 200, правило остановки построения графа Б.

Результат работы программы:

α = 1,66
Кол-во вершин = 306
Кол-во висячих вершин = 184
Высота дерева = 7

**Таблица всех вершин**

| | |
|---|---|
| 1 уровень | [(1, 0)] |
| 2 уровень | [(2-1), (3-1), (4-1), (5-1)] |
| 3 уровень | [(6-2), (7-2), (8-2), (9-2), (10-3), (11-3), (12-3), (13-3), (14-4)] |
| 4 уровень | [(15-6), (16-7), (17-7), (18-8), (19-8), (20-8), (21-10), (22-10), (23-10), (24-10), (25-11), (26-11), (27-12), (28-13), (29-13), (30-13), (31-13), (32-14)] |
| 5 уровень | [(33-16), (34-17), (35-17), (36-17), (37-18), (38-18), (39-18), (40-19), (41-19), (42-21), (43-21), (44-21), (45-22), (46-22), (47-22), (48-22), (49-23), (50-23), (51-23), (52-23), (53-24), (54-24), (55-25), (56-25), (57-26), (58-27), (59-28), (60-28), (61-28), (62-28), (63-30), (64-30), (65-30), (66-30), (67-32), (68-32), (69-32), (70-32)] |
| 6 уровень | [(71-34), (72-35), (73-35), (74-35), (75-35), (76-36), (77-36), (78-36), (79-37), (80-40), (81-40), (82-40), (83-40), (84-41), (85-41), (86-42), (87-42), (88-42), (89-42), (90-43), (91-43), (92-45), (93-46), (94-46), (95-47), (96-47), (97-48), (98-48), (99-48), (100-48), (101-49), (102-49), (103-49), (104-50), (105-50), (106-51), (107-51), (108-51), (109-52), (110-53), (111-54), (112-55), (113-56), (114-56), (115-56), (116-57), (117-57), (118-57), (119-58), (120-58), (121-58), (122-59), (123-60), (124-60), (125-60), (126-60), (127-61), (128-61), (129-61), (130-62), (131-62), (132-63), (133-63), (134-63), (135-63), (136-64), (137-65), (138-65), (139-65), (140-67), (141-67), (142-67), (143-67), (144-69), (145-70), (146-70)] |
| 7 уровень | [(147-72), (148-73), (149-74), (150-76), (151-76), (152-76), (153-76), (154-77), (155-78), (156-78), (157-78), (158-79), (159-79), (160-80), (161-81), (162-82), (163-82), (164-82), (165-82), (166-83), (167-83), (168-83), (169- |

| | |
|---|---|
| | 83), (170-84), (171-84), (172-85), (173-85), (174-86), (175-86), (176-87), (177-87), (178-88), (179-89), (180-89), (181-90), (182-90), (183-90), (184-91), (185-91), (186-92), (187-92), (188-92), (189-92), (190-93), (191-93), (192-93), (193-93), (194-94), (195-94), (196-95), (197-95), (198-95), (199-96), (200-96), (201-97), (202-97), (203-97), (204-98), (205-98), (206-98), (207-98), (208-100), (209-100), (210-100), (211-102), (212-102), (213-104), (214-105), (215-105), (216-106), (217-107), (218-108), (219-108), (220-108), (221-109), (222-109), (223-109), (224-110), (225-110), (226-111), (227-112), (228-112), (229-112), (230-112), (231-113), (232-113), (233-113), (234-113), (235-114), (236-114), (237-114), (238-114), (239-115), (240-115), (241-115), (242-115), (243-116), (244-116), (245-116), (246-116), (247-117), (248-117), (249-117), (250-118), (251-118), (252-118), (253-119), (254-119), (255-120), (256-120), (257-121), (258-121), (259-124), (260-124), (261-124), (262-124), (263-127), (264-127), (265-127), (266-127), (267-128), (268-128), (269-129), (270-129), (271-129), (272-130), (273-130), (274-131), (275-132), (276-132), (277-133), (278-133), (279-133), (280-133), (281-134), (282-134), (283-134), (284-134), (285-135), (286-135), (287-135), (288-135), (289-137), (290-137), (291-137), (292-137), (293-138), (294-138), (295-138), (296-138), (297-139), (298-139), (299-140), (300-140), (301-140), (302-142), (303-143), (304-145), (305-146), (306-146)] |

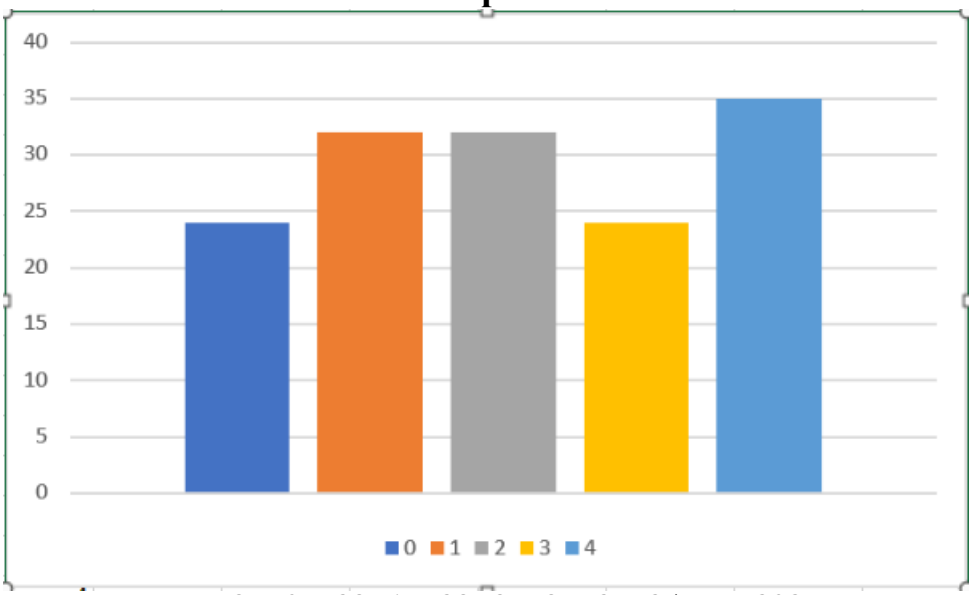
## Таблица висячих вершин

| |
|---|
| [(5-1), (9-2), (15-6), (20-8), (29-13), (31-13), (33-16), (38-18), (39-18), (44-21), (66-30), (68-32), (71-34), (75-35), (99-48), (101-49), (103-49), (122-59), (123-60), (125-60), (126-60), (136-64), (141-67), (144-69), (147-72), (148-73), (149-74), (150-76), (151-76), (152-76), (153-76), (154-77), (155-78), (156-78), (157-78), (158-79), (159-79), (160-80), (161-81), (162-82), (163-82), (164-82), (165-82), (166-83), (167-83), (168-83), (169-83), (170-84), (171-84), (172-85), (173-85), (174-86), (175-86), (176-87), (177-87), (178-88), (179-89), (180-89), (181-90), (182-90), (183-90), (184-91), (185-91), (186-92), (187-92), (188-92), (189-92), (190-93), (191-93), (192-93), (193-93), (194-94), (195-94), (196-95), (197-95), (198-95), (199-96), (200-96), (201-97), (202-97), (203-97), (204-98), (205-98), (206-98), (207-98), (208-100), (209-100), (210-100), (211-102), (212-102), (213-104), (214-105), (215-105), (216-106), (217-107), (218-108), (219-108), (220-108), (221-109), (222-109), (223-109), (224-110), (225-110), (226-111), (227-112), (228-112), (229-112), (230-112), (231-113), (232-113), (233-113), (234-113), (235-114), (236-114), (237-114), (238-114), (239-115), (240-115), (241-115), (242-115), (243-116), (244-116), (245-116), (246-116), (247-117), (248-117), (249-117), (250-118), (251-118), (252-118), (253-119), (254-119), (255-120), (256-120), (257-121), (258-121), (259-124), (260-124), (261-124), (262-124), (263-127), (264-127), (265-127), (266-127), (267-128), (268-128), (269-129), |

(270-129), (271-129), (272-130), (273-130), (274-131), (275-132), (276-132), (277-133), (278-133), (279-133), (280-133), (281-134), (282-134), (283-134), (284-134), (285-135), (286-135), (287-135), (288-135), (289-137), (290-137), (291-137), (292-137), (293-138), (294-138), (295-138), (296-138), (297-139), (298-139), (299-140), (300-140), (301-140), (302-142), (303-143), (304-145), (305-146), (306-146)]

Данные для гистограммы:
[0] =24, [1] = 32, [2] = 32, [3] = 24, [4] = 35;

**Гистограмма**




$$M[\alpha] = \frac{24 \cdot 0 + 32 \cdot 1 + 32 \cdot 2 + 24 \cdot 3 + 35 \cdot 4}{24 + 32 + 32 + 24 + 35} = \frac{308}{147} = 2.09$$

$$M_{\text{теор}}[m] = \frac{5 - 1}{2} = 2$$

Практическое мат. ожидание величины m-1 : 2,09
Теоретическое мат. ожидание величины m-1: 2

*Утверждение: Математическое ожидание попадает в допустимый интервал теоретического значения. Если совпадения нет, пользователю выдается ошибка несовпадения с интервалом:*

```
    if((Mat>2,2) ||(Mat<1,8)
     System.out.println("Мат ожидание выходит за допустимые пределы");
```

**Работа 3.** Этим же алгоритмом построить детерминированный граф ПО при фиксированном значении числа выходящих из узлов ребер (mфикс -1) = (m-1). Это можно сделать, заглушая обращение к датчику случайных чисел, и для каждого узла использовать заданную константу при определении числа исходящих ребер. Определить число висячих узлов и параметр$\alpha$. Убедится в правильности расчета структурного параметра α. В тексте программы и в отчете привести «утверждение».

Теоретический расчет:

$\alpha = \frac{m-1}{m-2} = \frac{4}{3} = 1.33$

Результат работы программы:

Альфа = 1.33

Число вершин = 341

Число висячих вершин = 256

Высота дерева = 5

***Таблица всех вершин***

| | |
|---|---|
| 1 уровень | [(1, 0)] |
| 2 уровень | [(2-1), (3-1), (4-1), (5-1)] |
| 3 уровень | [(6-2), (7-2), (8-2), (9-2), (10-3), (11-3), (12-3), (13-3), (14-4), (15-4), (16-4), (17-4), (18-5), (19-5), (20-5), (21-5)] |
| 4 уровень | [(22-6), (23-6), (24-6), (25-6), (26-7), (27-7), (28-7), (29-7), (30-8), (31-8), (32-8), (33-8), (34-9), (35-9), (36-9), (37-9), (38-10), (39-10), (40-10), (41-10), (42-11), (43-11), (44-11), (45-11), (46-12), (47-12), (48-12), (49-12), (50-13), (51-13), (52-13), (53-13), (54-14), (55-14), (56-14), (57-14), (58-15), (59-15), (60-15), (61-15), (62-16), (63-16), (64-16), (65-16), (66-17), (67-17), (68-17), (69-17), (70-18), (71-18), (72-18), (73-18), (74-19), (75-19), (76-19), (77-19), (78-20), (79-20), (80-20), (81-20), (82-21), (83-21), (84-21), (85-21)] |
| 5 уровень | [(86-22), (87-22), (88-22), (89-22), (90-23), (91-23), (92-23), (93-23), (94-24), (95-24), (96-24), (97-24), (98-25), (99-25), (100-25), (101-25), (102-26), (103-26), (104-26), (105-26), (106-27), (107-27), (108-27), (109-27), (110-28), (111-28), (112-28), (113-28), (114-29), (115-29), (116-29), (117-29), (118-30), (119-30), (120-30), (121-30), (122-31), (123-31), (124-31), (125-31), (126-32), (127-32), (128-32), (129-32), (130-33), (131-33), (132-33), (133-33), (134-34), (135-34), (136-34), (137-34), (138-35), (139-35), (140-35), (141-35), (142-36), (143-36), (144-36), (145-36), (146-37), (147-37), (148-37), (149-37), (150-38), (151-38), (152-38), (153-38), (154-39), (155-39), (156-39), (157-39), (158-40), (159-40), (160-40), (161-40), (162-41), (163-41), (164-41), (165-41), (166-42), (167-42), (168-42), (169-42), (170-43), (171-43), (172-43), (173-43), (174-44), (175-44), (176-44), (177-44), (178-45), (179-45), (180-45), (181-45), (182-46), (183-46), (184-46), (185-46), (186-47), (187-47), (188-47), (189-47), (190-48), (191- |

| | |
|---|---|
| | 48), (192-48), (193-48), (194-49), (195-49), (196-49), (197-49), (198-50), (199-50), (200-50), (201-50), (202-51), (203-51), (204-51), (205-51), (206-52), (207-52), (208-52), (209-52), (210-53), (211-53), (212-53), (213-53), (214-54), (215-54), (216-54), (217-54), (218-55), (219-55), (220-55), (221-55), (222-56), (223-56), (224-56), (225-56), (226-57), (227-57), (228-57), (229-57), (230-58), (231-58), (232-58), (233-58), (234-59), (235-59), (236-59), (237-59), (238-60), (239-60), (240-60), (241-60), (242-61), (243-61), (244-61), (245-61), (246-62), (247-62), (248-62), (249-62), (250-63), (251-63), (252-63), (253-63), (254-64), (255-64), (256-64), (257-64), (258-65), (259-65), (260-65), (261-65), (262-66), (263-66), (264-66), (265-66), (266-67), (267-67), (268-67), (269-67), (270-68), (271-68), (272-68), (273-68), (274-69), (275-69), (276-69), (277-69), (278-70), (279-70), (280-70), (281-70), (282-71), (283-71), (284-71), (285-71), (286-72), (287-72), (288-72), (289-72), (290-73), (291-73), (292-73), (293-73), (294-74), (295-74), (296-74), (297-74), (298-75), (299-75), (300-75), (301-75), (302-76), (303-76), (304-76), (305-76), (306-77), (307-77), (308-77), (309-77), (310-78), (311-78), (312-78), (313-78), (314-79), (315-79), (316-79), (317-79), (318-80), (319-80), (320-80), (321-80), (322-81), (323-81), (324-81), (325-81), (326-82), (327-82), (328-82), (329-82), (330-83), (331-83), (332-83), (333-83), (334-84), (335-84), (336-84), (337-84), (338-85), (339-85), (340-85), (341-85)] |

## Таблица висячих вершин

| |
|---|
| [(86-22), (87-22), (88-22), (89-22), (90-23), (91-23), (92-23), (93-23), (94-24), (95-24), (96-24), (97-24), (98-25), (99-25), (100-25), (101-25), (102-26), (103-26), (104-26), (105-26), (106-27), (107-27), (108-27), (109-27), (110-28), (111-28), (112-28), (113-28), (114-29), (115-29), (116-29), (117-29), (118-30), (119-30), (120-30), (121-30), (122-31), (123-31), (124-31), (125-31), (126-32), (127-32), (128-32), (129-32), (130-33), (131-33), (132-33), (133-33), (134-34), (135-34), (136-34), (137-34), (138-35), (139-35), (140-35), (141-35), (142-36), (143-36), (144-36), (145-36), (146-37), (147-37), (148-37), (149-37), (150-38), (151-38), (152-38), (153-38), (154-39), (155-39), (156-39), (157-39), (158-40), (159-40), (160-40), (161-40), (162-41), (163-41), (164-41), (165-41), (166-42), (167-42), (168-42), (169-42), (170-43), (171-43), (172-43), (173-43), (174-44), (175-44), (176-44), (177-44), (178-45), (179-45), (180-45), (181-45), (182-46), (183-46), (184-46), (185-46), (186-47), (187-47), (188-47), (189-47), (190-48), (191-48), (192-48), (193-48), (194-49), (195-49), (196-49), (197-49), (198-50), (199-50), (200-50), (201-50), (202-51), (203-51), (204-51), (205-51), (206-52), (207-52), (208-52), (209-52), (210-53), (211-53), (212-53), (213-53), (214-54), (215-54), (216-54), (217-54), (218-55), (219-55), (220-55), (221-55), (222-56), (223-56), (224-56), (225-56), (226-57), (227-57), (228-57), (229-57), (230-58), (231-58), (232-58), (233-58), (234-59), (235-59), (236-59), (237-59), (238-60), (239-60), (240-60), (241-60), (242-61), (243-61), (244-61), (245-61), (246-62), (247-62), (248-62), (249-62), (250-63), (251-63), (252-63), (253-63), (254-64), (255-64), (256-64), (257-64), (258-65), (259-65), (260-65), (261-65), (262-66), (263-66), (264-66), (265-66), (266-67), (267-67), (268-67), (269-67), (270-68), (271-68), (272-68), (273-68), (274-69), (275-69), (276-69), (277-69), (278-70), (279-70), (280-70), (281-70), (282-71), (283-71), (284-71), (285-71), (286-72), (287-72), (288-72), (289-72), (290-73), (291-73), (292-73), (293-73), (294-74), (295-74), (296-74), (297-74), (298-75), (299-75), (300-75), (301-75), (302-76), (303-76), (304-76), (305-76), (306- |

77), (307-77), (308-77), (309-77), (310-78), (311-78), (312-78), (313-78), (314-79), (315-79), (316-79), (317-79), (318-80), (319-80), (320-80), (321-80), (322-81), (323-81), (324-81), (325-81), (326-82), (327-82), (328-82), (329-82), (330-83), (331-83), (332-83), (333-83), (334-84), (335-84), (336-84), (337-84), (338-85), (339-85), (340-85), (341-85)]

***Вывод:*** Практическое значение α=1,33 совпало с теоретическим значением α=1,33.

***Утверждение.*** Если значение α случайного дерева в программе меньше чем 1,3, то в программе есть ошибка.

```
if (alpha <= 1,33) System.out.println("Ошибка");
```

**Работа 4.** Подсчитать среднее число висячих вершин у совокупности R случайных графов ПО, определить среднее значение и дисперсию структурного параметра α = отношению общего числа вершин к числу висячих вершин.
R=100, N=200, m=5

| № | Уровни | Всего вершин | Висячие вершины | α |
|---|---|---|---|---|
| 1 | 8 | 302 | 189 | 1.6 |
| 2 | 7 | 366 | 223 | 1.64 |
| 3 | 8 | 262 | 159 | 1.65 |
| 4 | 8 | 389 | 237 | 1.64 |
| 5 | 7 | 216 | 131 | 1.65 |
| 6 | 9 | 369 | 224 | 1.65 |
| 7 | 8 | 237 | 141 | 1.68 |
| 8 | 8 | 239 | 142 | 1.68 |
| 9 | 8 | 321 | 188 | 1.71 |
| 10 | 7 | 320 | 193 | 1.66 |
| 11 | 7 | 313 | 191 | 1.64 |
| 12 | 9 | 239 | 143 | 1.67 |

| | | | | |
|---|---|---|---|---|
| 13 | 8 | 293 | 179 | 1.64 |
| 14 | 8 | 351 | 205 | 1.71 |
| 15 | 14 | 227 | 135 | 1.68 |
| 16 | 10 | 323 | 199 | 1.62 |
| 17 | 7 | 294 | 177 | 1.66 |
| 18 | 7 | 267 | 159 | 1.68 |
| 19 | 7 | 302 | 189 | 1.6 |
| 20 | 9 | 405 | 246 | 1.65 |
| 21 | 8 | 352 | 219 | 1.61 |
| 22 | 8 | 215 | 126 | 1.71 |
| 23 | 7 | 265 | 154 | 1.72 |
| 24 | 7 | 346 | 214 | 1.62 |
| 25 | 8 | 282 | 163 | 1.73 |
| 26 | 10 | 233 | 141 | 1.65 |
| 27 | 10 | 366 | 216 | 1.69 |
| 28 | 8 | 230 | 139 | 1.65 |
| 29 | 10 | 249 | 156 | 1.6 |
| 30 | 10 | 234 | 144 | 1.63 |
| 31 | 9 | 224 | 142 | 1.58 |
| 32 | 6 | 271 | 171 | 1.58 |
| 33 | 9 | 293 | 173 | 1.69 |
| 34 | 7 | 251 | 149 | 1.68 |
| 35 | 8 | 331 | 200 | 1.66 |
| 36 | 7 | 214 | 126 | 1.7 |
| 37 | 6 | 219 | 134 | 1.63 |

| | | | | |
|---|---|---|---|---|
| 38 | 8 | 220 | 139 | 1.58 |
| 39 | 10 | 247 | 145 | 1.7 |
| 40 | 8 | 382 | 228 | 1.68 |
| 41 | 8 | 300 | 182 | 1.65 |
| 42 | 9 | 209 | 128 | 1.63 |
| 43 | 7 | 339 | 204 | 1.66 |
| 44 | 8 | 256 | 156 | 1.64 |
| 45 | 7 | 304 | 184 | 1.65 |
| 46 | 8 | 281 | 160 | 1.76 |
| 47 | 8 | 226 | 144 | 1.57 |
| 48 | 8 | 385 | 234 | 1.65 |
| 49 | 8 | 312 | 189 | 1.65 |
| 50 | 7 | 286 | 173 | 1.65 |
| 51 | 8 | 334 | 199 | 1.68 |
| 52 | 9 | 321 | 189 | 1.7 |
| 53 | 7 | 238 | 135 | 1.76 |
| 54 | 10 | 284 | 171 | 1.66 |
| 55 | 8 | 230 | 139 | 1.65 |
| 56 | 7 | 334 | 205 | 1.63 |
| 57 | 8 | 269 | 167 | 1.61 |
| 58 | 7 | 210 | 130 | 1.62 |
| 59 | 7 | 240 | 150 | 1.6 |
| 60 | 9 | 384 | 235 | 1.63 |
| 61 | 7 | 385 | 237 | 1.62 |
| 62 | 8 | 209 | 124 | 1.69 |

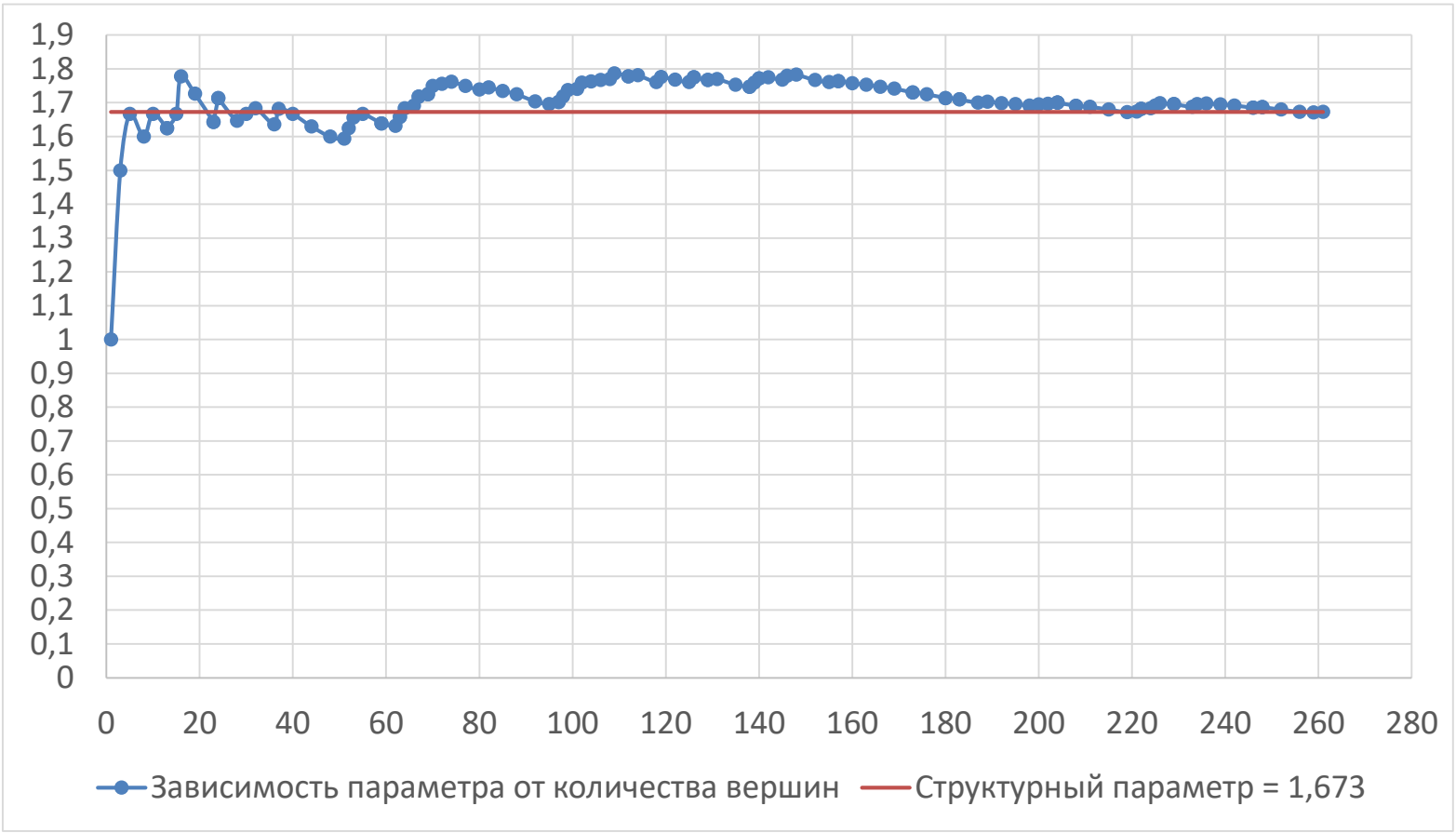
| | | | | |
|---|---|---|---|---|
| 63 | 9 | 303 | 182 | 1.66 |
| 64 | 9 | 277 | 165 | 1.68 |
| 65 | 7 | 388 | 236 | 1.64 |
| 66 | 10 | 306 | 187 | 1.64 |
| 67 | 8 | 337 | 200 | 1.69 |
| 68 | 7 | 242 | 145 | 1.67 |
| 69 | 8 | 348 | 215 | 1.62 |
| 70 | 10 | 244 | 144 | 1.69 |
| 71 | 8 | 238 | 141 | 1.69 |
| 72 | 8 | 218 | 133 | 1.64 |
| 73 | 7 | 208 | 131 | 1.59 |
| 74 | 9 | 428 | 262 | 1.63 |
| 75 | 10 | 240 | 133 | 1.8 |
| 76 | 8 | 239 | 136 | 1.76 |
| 77 | 7 | 266 | 161 | 1.65 |
| 78 | 7 | 234 | 140 | 1.67 |
| 79 | 8 | 255 | 146 | 1.75 |
| 80 | 8 | 271 | 155 | 1.75 |
| 81 | 8 | 380 | 234 | 1.62 |
| 82 | 8 | 297 | 174 | 1.71 |
| 83 | 10 | 365 | 220 | 1.66 |
| 84 | 9 | 236 | 141 | 1.67 |
| 85 | 7 | 210 | 127 | 1.65 |
| 86 | 7 | 208 | 128 | 1.63 |
| 87 | 7 | 270 | 165 | 1.64 |

| | | | | |
|---|---|---|---|---|
| 88 | 7 | 238 | 151 | 1.58 |
| 89 | 8 | 311 | 177 | 1.76 |
| 90 | 11 | 321 | 200 | 1.61 |
| 91 | 7 | 268 | 164 | 1.63 |
| 92 | 7 | 212 | 134 | 1.58 |
| 93 | 8 | 258 | 158 | 1.63 |
| 94 | 7 | 372 | 216 | 1.72 |
| 95 | 7 | 375 | 223 | 1.68 |
| 96 | 8 | 229 | 132 | 1.73 |
| 97 | 8 | 344 | 206 | 1.67 |
| 98 | 7 | 206 | 122 | 1.69 |
| 99 | 7 | 234 | 132 | 1.77 |
| 100 | 7 | 370 | 225 | 1.64 |

$M[\alpha] = 1.660619$ , $D[\alpha] = 0.002240$

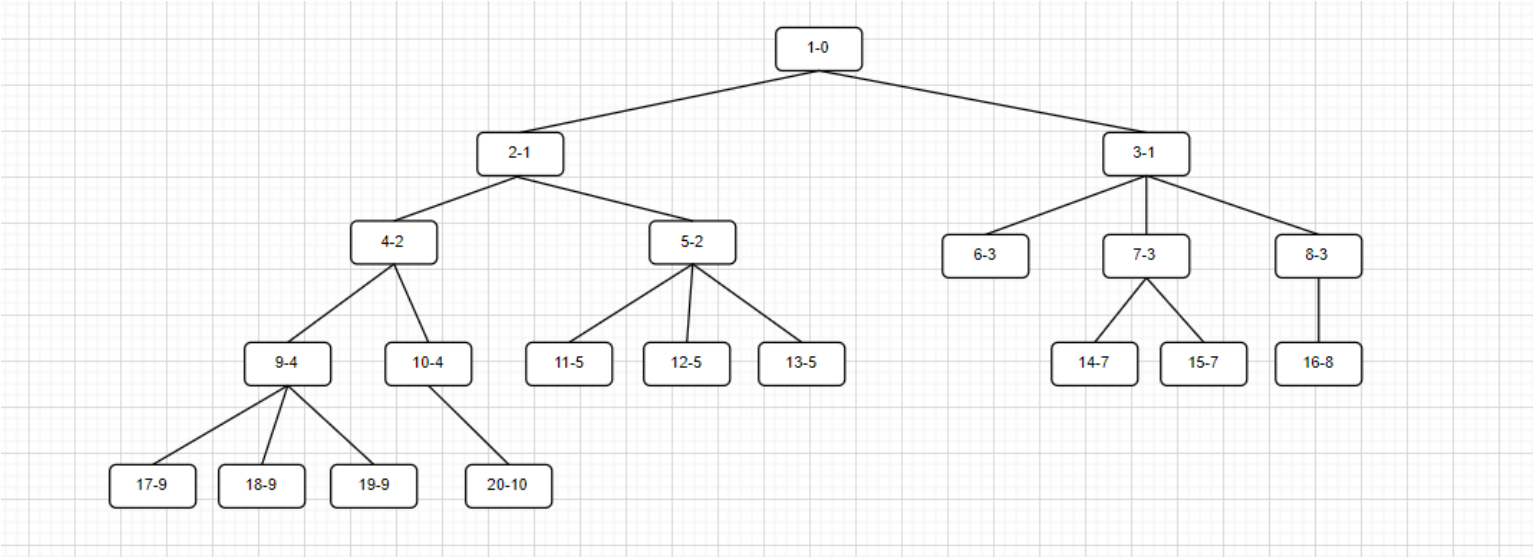
**Работа 5.** Исследовать сходимость значения структурного параметра $\alpha$ к устойчивому значению с ростом числа узлов графа P, построив для этого функцию изменения структурного параметра $\alpha$ с ростом числа узлов P. Для данного графа построить графический фрагмент из первых двадцати узлов.

**График зависимости структурного параметра $\alpha$ от числа вершин**




**Графическое представление первых 20 вершин графа**

| | |
|---|---|
| 1 уровень | [(1, 0)] |
| 2 уровень | [(2,1),(3,1)] |
| 3 уровень | [(4-2), (5-2), (6-3), (7-3), (8-3)] |
| 4 уровень | [(9-4), (10-4), (11-5), (12-5), (13-5), (14-7), (15-7), (16-8)] |
| 5 уровень | [(17-9), (18-9), (19-9), (20-10),…] |




## Листинг программы

```
package TreeStructure;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;

/**
 * Вариант #5
 * m = 5 - возможное количество потомков у узла
 * N = 200 - общее количество узлов
 * R = 100 - количество графов для построения
 * Правило остагновки Б) - число узлов >= заданному числу N && последний уровень иерархии графа должен быть до конца
 * заполнен  висячими узлами
 */

class Tree {

   private TreeNode root;
   public int[] gI = {0,0,0,0,0};
   private int vertexCnt;
   private LinkedList<TreeNode> terminalVertexes = new LinkedList<>();
   private HashMap<Integer, LinkedList<TreeNode>> hierarchy = new HashMap<>();
   private final Random generator = new Random();

   public TreeNode getRoot() {
      return root;
   }

   public int getVertexCnt() {
      return vertexCnt;
   }

   public HashMap<Integer, LinkedList<TreeNode>> getHierarchy(){
      return hierarchy;
   }
```

```
public LinkedList<TreeNode> getTerminalVertexes() {
   return terminalVertexes;
}

private TreeNode rootInitialize(int maxChildCnt, boolean isRandom) {
   LinkedList<TreeNode> rootLevel = new LinkedList<>();
   TreeNode root = new TreeNode(1, 0, 1);
   int childCnt = 0;
   rootLevel.add(root);
   hierarchy.put(1, rootLevel);
   if(isRandom) {
      while (childCnt == 0) {
         childCnt = generator.nextInt(maxChildCnt);
         gI[childCnt]++;
      }
   }else childCnt = maxChildCnt;
   root.childList = new LinkedList<>();
   for (int i = 2; i <= childCnt + 1; i++) {
      root.childList.add(new TreeNode(i, 1, 2));
   }
   hierarchy.put(2, root.childList);
   return root;
}

Tree(int maxVertexCnt, int maxChildCnt, boolean isRandom) {
   File file = new File("graphics.txt");
   try (FileWriter writer = new FileWriter(file)) {
      this.root = rootInitialize(maxChildCnt, isRandom);
      Deque<TreeNode> nodeQueue = new LinkedList<>();
      int curNodeIndex = this.root.childList.size() + 1;
      int extremeHierarchyLevel = 0;
      boolean isFirstOverFlowed = false;
      int parentIndex = root.index;
      for (TreeNode childNode : root.childList) {
         if (childNode != null) {
            nodeQueue.addLast(childNode);
         }
      }
      int index = 1;
      while (!nodeQueue.isEmpty()) {
         writer.write(Integer.toString(index));
         writer.write("\t");
         writer.write(Integer.toString(curNodeIndex));
         writer.write("\t");

writer.write(Double.toString(Math.round((double)curNodeIndex/((double)nodeQueue.size() +
terminalVertexes.size() )* 1000.0) / 1000.0));
         writer.write("\n");
         index++;
         TreeNode processedNode = nodeQueue.removeFirst();
         int childCnt;
         if (isRandom) {
            childCnt = generator.nextInt(maxChildCnt);
            gI[childCnt]++;
         } else childCnt = maxChildCnt;
```

```
                    if (childCnt == 0) {
                        terminalVertexes.add(processedNode);
                        parentIndex++;
                        continue;
                    }
                    if (curNodeIndex + 1 > maxVertexCnt && processedNode.hierarchyLevel !=
extremeHierarchyLevel) {
                        terminalVertexes.add(processedNode);
                        for (TreeNode node : nodeQueue) {
                            terminalVertexes.add(node);
                        }
                        break;
                    }
                    processedNode.childList = new LinkedList<>();
                    parentIndex++;
                    for (int i = 0; i < childCnt; i++) {
                        curNodeIndex++;
                        int childHierarchyLevel = processedNode.hierarchyLevel + 1;
                        TreeNode childNode = new TreeNode(curNodeIndex, parentIndex,
childHierarchyLevel);
                        if (!hierarchy.containsKey(childHierarchyLevel)) {
                            hierarchy.put(childHierarchyLevel, new LinkedList<TreeNode>());
                        }
                        hierarchy.get(childHierarchyLevel).add(childNode);
                        processedNode.childList.add(childNode);
                    }
                    if (curNodeIndex >= maxVertexCnt && !isFirstOverFlowed) {
                        extremeHierarchyLevel = processedNode.hierarchyLevel;
                        isFirstOverFlowed = true;
                    }
                    for (TreeNode childNode : processedNode.childList) nodeQueue.addLast(childNode);
                }
                this.vertexCnt = curNodeIndex;
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        static class TreeNode {
            LinkedList<TreeNode> childList;
            int index;
            int parentIndex;
            int hierarchyLevel;

            TreeNode(int index, int parentIndex, int hierarchyLevel) {
                this.index = index;
                this.parentIndex = parentIndex;
                this.hierarchyLevel = hierarchyLevel;
            }

            @Override
            public String toString() {
                return "(" + index + "-" + parentIndex /*+ "-" + (childList != null ? childList.size() : 0 )*/ +
")";
            }

        }
```

```
    public String noChild(){
        Deque<TreeNode> nodeQueue = new LinkedList<>();
        nodeQueue.addLast(root);
        StringBuilder builder = new StringBuilder();
        builder.append("[");
        while (!nodeQueue.isEmpty()) {
            TreeNode processedNode = nodeQueue.removeFirst();
            if((processedNode.childList != null ? processedNode.childList.size() : 0) == 0) {
                builder.append(processedNode.index);
                builder.append(", ");
            }
            if (processedNode.childList == null) continue;
            for (TreeNode childNode : processedNode.childList) nodeQueue.addLast(childNode);
        }
        builder.append("]");
        return builder.toString();
    }

    public double getAlphaValue(){
        return (double) vertexCnt / (double) terminalVertexes.size();
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        for (int i = 1; i <= hierarchy.size(); i++) {
            builder.append(hierarchy.get(i).toString());
            builder.append("\n");
        }
        return builder.toString();
    }
}
```