Министерство науки и высшего образования РФ

ФГБОУ ВО «Тверской государственный университет»

направление «Фундаментальная информатика и информационные

технологии»

**РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА**

по дисциплине «Теоретические основы информатики»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | Автор:  Иванов Иван Иванович | |  |  | |  |

Тверь 2018

# Объект исследования

Для выполнения расчетно-графической работы было выбрано литературное произведение Адамса Дугласа «Детективное агентство Дирка Джентли».

Ссылка на исходный текст: <http://lib.ru/ADAMS/gently_1.txt>

# Информационные характеристики

Частотная характеристика текста:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Символ** | **Количество** | **Вероятность** | **%** |
| 1 | (пробел) | 82902 | 0,1539184 | 15,39% |
| 2 | О | 51012 | 0,0947105 | 9,47% |
| 3 | ЕЁ | 36961 | 0,0686229 | 6,86% |
| 4 | А | 33252 | 0,0617367 | 6,17% |
| 5 | Н | 29630 | 0,0550120 | 5,50% |
| 6 | И | 27535 | 0,0511223 | 5,11% |
| 7 | Т | 26706 | 0,0495832 | 4,96% |
| 8 | С | 24132 | 0,0448042 | 4,48% |
| 9 | Л | 21121 | 0,0392139 | 3,92% |
| 10 | Р | 20422 | 0,0379161 | 3,79% |
| 11 | В | 19026 | 0,0353243 | 3,53% |
| 12 | К | 14968 | 0,0277901 | 2,78% |
| 13 | М | 13877 | 0,0257645 | 2,58% |
| 14 | Д | 13833 | 0,0256828 | 2,57% |
| 15 | П | 12742 | 0,0236572 | 2,37% |
| 16 | У | 12393 | 0,0230092 | 2,30% |
| 17 | Я | 8728 | 0,0162047 | 1,62% |
| 18 | ЬЪ | 8378 | 0,0155549 | 1,56% |
| 19 | , | 8186 | 0,0151984 | 1,52% |
| 20 | Ы | 7855 | 0,0145838 | 1,46% |
| 21 | З | 7506 | 0,0139359 | 1,39% |
| 22 | Г | 7418 | 0,0137725 | 1,38% |
| 23 | Ч | 7418 | 0,0137725 | 1,38% |
| 24 | Б | 6895 | 0,0128015 | 1,28% |
| 25 | - | 6228 | 0,0115631 | 1,16% |
| 26 | Й | 4756 | 0,0088301 | 0,88% |
| 27 | Ж | 4276 | 0,0079390 | 0,79% |
| 28 | Ш | 3753 | 0,0069679 | 0,70% |
| 29 | - | 3322 | 0,0061677 | 0,62% |
| 30 | Х | 3229 | 0,0059951 | 0,60% |
| 31 | Ю | 2618 | 0,0048607 | 0,49% |
| 32 | Э | 2007 | 0,0037263 | 0,37% |
| 33 | Ц | 1396 | 0,0025919 | 0,26% |
| 34 | Щ | 1353 | 0,0025120 | 0,25% |
| 35 | Ф | 1222 | 0,0022688 | 0,23% |
| 36 | ? | 655 | 0,0012161 | 0,12% |
| 37 | “ | 426 | 0,0007909 | 0,08% |
| 38 | : | 155 | 0,0002878 | 0,03% |
| 39 | ! | 141 | 0,0002618 | 0,03% |
| 40 | 1 | 30 | 0,0000557 | 0,01% |
| 41 | 2 | 27 | 0,0000501 | 0,01% |
| 42 | 3 | 26 | 0,0000483 | 0,00% |
| 43 | 0 | 16 | 0,0000297 | 0,00% |
| 44 | 7 | 15 | 0,0000278 | 0,00% |
| 45 | 6 | 13 | 0,0000241 | 0,00% |
| 46 | 4 | 12 | 0,0000223 | 0,00% |
| 47 | 8 | 11 | 0,0000204 | 0,00% |
| 48 | ; | 10 | 0,0000186 | 0,00% |
| 49 | 5 | 10 | 0,0000186 | 0,00% |
| 50 | 9 | 7 | 0,0000130 | 0,00% |

Гистограмма распределения самых частых символов (не менее 2%):

Основные показатели:

|  |  |
| --- | --- |
| Мощность алфавита | 50 символов |
| Размер текста | 538 610 символов |
| Размер текста в битах (N) | 4 308 880 бит |
| Энтропия по Хартли | 5.643856 |
| Количество информации по Хартли | 3 039 837 бит |
| Энтропия по Шеннону () | 4.516293 |
| Количество информации по Шеннону | 2 432 521 бит |
| Минимальная длина кода | 4.516293 |

# Оптимальный равномерный код

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Символ** | **Код** | **№** | **Символ** | **Код** | **№** | **Символ** | **Код** |
| 1 | (пробел) | 000000 | 18 | ЬЪ | 010001 | 35 | Ф | 100010 |
| 2 | О | 000001 | 19 | , | 010010 | 36 | ? | 100011 |
| 3 | ЕЁ | 000010 | 20 | Ы | 010011 | 37 | “ | 100100 |
| 4 | А | 000011 | 21 | З | 010100 | 38 | : | 100101 |
| 5 | Н | 000100 | 22 | Г | 010101 | 39 | ! | 100110 |
| 6 | И | 000101 | 23 | Ч | 010110 | 40 | 1 | 100111 |
| 7 | Т | 000110 | 24 | Б | 010111 | 41 | 2 | 101000 |
| 8 | С | 000111 | 25 | - | 011000 | 42 | 3 | 101001 |
| 9 | Л | 001000 | 26 | Й | 011001 | 43 | 0 | 101010 |
| 10 | Р | 001001 | 27 | Ж | 011010 | 44 | 7 | 101011 |
| 11 | В | 001010 | 28 | Ш | 011011 | 45 | 6 | 101100 |
| 12 | К | 001011 | 29 | - | 011100 | 46 | 4 | 101101 |
| 13 | М | 001100 | 30 | Х | 011101 | 47 | 8 | 101110 |
| 14 | Д | 001101 | 31 | Ю | 011110 | 48 | ; | 101111 |
| 15 | П | 001110 | 32 | Э | 011111 | 49 | 5 | 110000 |
| 16 | У | 001111 | 33 | Ц | 100000 | 50 | 9 | 110001 |
| 17 | Я | 010000 | 34 | Щ | 100001 |  |  |  |

Основные показатели:

|  |  |
| --- | --- |
| Длина оптимального равномерного кода () | 6 бит |
| Размер закодированного текста () | 3 231 660 бита |
| Относительная избыточность кода | 0,3285 |
| Коэффициент сжатия | 75% |

# Оптимальный неравномерный префиксный код

Для построения оптимального префиксного кода был выбран алгоритм Хаффмана.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Символ** | **Код** | **№** | **Символ** | **Код** | **№** | **Символ** | **Код** |
| 1 | (пробел) | 110 | 18 | ЬЪ | 100010 | 35 | Ф | 101110110 |
| 2 | О | 000 | 19 | , | 100001 | 36 | ? | 1011111110 |
| 3 | ЕЁ | 1010 | 20 | Ы | 100000 | 37 | “ | 10111111110 |
| 4 | А | 1001 | 21 | З | 011101 | 38 | : | 1011111111111 |
| 5 | Н | 0110 | 22 | Ч | 010111 | 39 | ! | 1011111111110 |
| 6 | И | 0100 | 23 | Г | 011100 | 40 | 1 | 10111111111000 |
| 7 | Т | 0010 | 24 | Б | 001101 | 41 | 2 | 101111111110110 |
| 8 | С | 11110 | 25 | Й | 1011110 | 42 | 3 | 101111111110101 |
| 9 | Л | 11101 | 26 | Ж | 1011100 | 43 | 0 | 101111111110010 |
| 10 | Р | 11100 | 27 | Ш | 0101101 | 44 | 7 | 1011111111101111 |
| 11 | В | 10110 | 28 | - | 0011001 | 45 | 6 | 1011111111101110 |
| 12 | К | 01111 | 29 | - | 0101100 | 46 | 4 | 1011111111101001 |
| 13 | М | 01010 | 30 | Х | 0011000 | 47 | 8 | 1011111111101000 |
| 14 | Д | 00111 | 31 | Ю | 10111110 | 48 | ; | 1011111111100111 |
| 15 | П | 111111 | 32 | Э | 10111010 | 49 | 5 | 10111111111001101 |
| 16 | У | 111110 | 33 | Ц | 101111110 | 50 | 9 | 10111111111001100 |
| 17 | Я | 100011 | 34 | Щ | 101110111 |  |  |  |

Основные показатели:

|  |  |
| --- | --- |
| Средняя длина оптимального префиксного кода | 4,5479623 бита |
| Размер закодированного текста () | 2 449 578 бита |
| Относительная избыточность кода | 0,007 |
| Коэффициент сжатия | 56,8% |

# Сжатие словарным методом

|  |  |
| --- | --- |
| Размер файла до сжатия | 538 610 байт |
| Размер файла после сжатия методом LZW (zip-архив) | 156 786 байт |
| Коэффициент сжатия | 29.1% |

# Программный код

**Программа для расчета основных информационных характеристик**

from math import \*

from random import \*

popsize = 50

a=0.05

Pc=0.8

Pm=0.2

maxM=1

NumIterations=2000

seed()

def shouldDo(probability):

if randint(0,100)/100 <= probability: return True

return False

def generateVector():

x = randint(-100,100)/100

y = sqrt(1-x\*x)

return (x,y)

def func3(x,y):

if x <= 0 or y <= 0: return False

if (8\*x-2\*y-max(3/2\*x,0.5\*y)\*0.7 <= 16.45 and 8\*x-2\*y+max(3/2\*x,0.5\*y)\*0.7>=11.55): return True

else: return False

def func4(x,y):

if x <= 0 or y <= 0: return False

if (x-5\*y-max(x,2\*y)\*0.6 <= 1.5 and x-5\*y+max(x,2\*y)\*0.6 >= -1.5): return True

else: return False

def func7(x,y):

if x == 0 or y == 0: return False

if (((11.55 <= 8\*x-2\*y <= 16.45) or (8\*x-2\*y-max(3/2\*x,0.5\*y)\*0.7<=14<=8\*x-2\*y+max(3/2\*x,0.5\*y)\*0.7))): return True

else: return False

def func8(x,y):

if x == 0 or y == 0: return False

if (((-1.5<=x-5\*y<=1.5) or (x-5\*y-max(x,2\*y)\*0.6<=0<=x-5\*y+max(x,2\*y)\*0.6))): return True

else: return False

def val(x,y):

return x+5\*y+0.5\*max(x,3/2\*y)

def eval(i):

return a\*pow((1-a),i-1)

def init():

res = []

for i in range(popsize):

x = randint(0,1000)/100

y = randint(0,1000)/100

while (not func7(x,y) or not func8(x,y)):

x = randint(0,1000)/100

y = randint(0,1000)/100

res.append((x,y))

return res

def crossover(population):

res = []

for i in range(len(population)):

if shouldDo(Pc):

res.append(i)

if len(res)%2 != 0: res.append(0)

shuffle(res)

for i in range(int(len(res)/2)):

c = randint(0,100)/100

X1 = c \* population[res[2\*i]][0] + (1-c)\*population[res[2\*i+1]][0]

Y1 = c \* population[res[2\*i]][1] + (1-c)\*population[res[2\*i+1]][1]

X2 = (1-c) \* population[res[2\*i]][0] + c\*population[res[2\*i+1]][0]

Y2 = (1-c) \* population[res[2\*i]][1] + c\*population[res[2\*i+1]][1]

while (not func7(X1,Y1) or not func8(X1,Y1)):

c = randint(0,100)/100

X1 = c \* population[res[2\*i]][0] + (1-c)\*population[res[2\*i+1]][0]

Y1 = c \* population[res[2\*i]][1] + (1-c)\*population[res[2\*i+1]][1]

while (not func7(X2,Y2) or not func8(X2,Y2)):

c = randint(0,100)/100

X2 = (1-c) \* population[res[2\*i]][0] + c\*population[res[2\*i+1]][0]

Y2 = (1-c) \* population[res[2\*i]][1] + c\*population[res[2\*i+1]][1]

population[res[2\*i]]=(X1,Y1)

population[res[2\*i+1]]=(X2,Y2)

def mutation(population):

res = []

for i in range(len(population)):

if shouldDo(Pm):

res.append(i)

for i in res:

M = randint(0,100)/100 \* maxM

d = generateVector()

X1 = population[i][0] + M\*d[0]

Y1 = population[i][1] + M\*d[1]

while (not func7(X1,Y1) or not func8(X1,Y1)):

M = randint(0,100)/100 \* M

X1 = population[i][0] + M\*d[0]

Y1 = population[i][1] + M\*d[1]

population[i]=(X1,Y1)

def selection(population):

population.sort(key=lambda v:val(v[0],v[1]),reverse=True)

for i in range(int(len(population)/2)):

population[len(population)-i-1]=population[i]

def genetic():

population = init()

population.sort(key=lambda v:val(v[0],v[1]),reverse=True)

best=population[0]

for i in range(NumIterations):

if(val(best[0],best[1]) < val(population[0][0],population[0][1])): best=population[0]

print(str(best[0]) + ", " + str(best[1]) + ": " + str(val(best[0],best[1])))

crossover(population)

mutation(population)

selection(population)

genetic()