

**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**Научно-исследовательская работа**

**“Быстрые алгоритмы подсчета и поиска двоичных разрядов”**

Студент Журавлев Н.В.

Группа РК6-42Б

Тип задания Научно-исследовательская работа

Студент

**Журавлев Н.В.**

*подпись, дата фамилия, и.о.*

Преподаватель

**Родионов С.В.**

*подпись, дата фамилия, и.о.*

*Москва, 2021*

**Оглавление**

**Оглавление2**

**Техническое задание3**

**Введение4**

**Программная реализация4**

**Тесты5**

**Листинг программы6**

**Результаты исследования18**

**Техническое задание**

Быстрые алгоритмы подсчета и поиска двоичных разрядов

**Введение**

Данная тема состоит из 2 подтем: поиска старшего бита и подсчёт единичных битов. В первой подтеме изложенные алгоритмы помогут ускорить работу с битами, а другая больше похожа на развлечение, чем на применение в реальной жизни, хотя есть задача range minimum query (RMQ) в которой это применимо.

Рассматриваемые алгоритмы могут быть применены для чисел из 8, 16, 32, 64 бит. Тестирование выполнена на языке C на процессоре Intel(R) Core(TM) i3 CPU M380 @3GHz на 64-х битовой Ubuntu 20.04.

**Программная реализация**

Файл main.c – содержит вызов и измерения для алгоритмов. Библиотека bit.c содержит следующие функции:

int CountOnes5(unsigned x) – алгоритм Разделяй и властвуй

int CountOnes4(unsigned x) – алгоритм Комбинированный метод

int CountOnes3(unsigned x) – алгоритм Параллельное суммирование

int CountOnes2(unsigned x) – алгоритм Умножение и остаток от деления

int CountOnes1(unsigned x) – алгоритм Дуальный

int CountOnes0(unsigned x) – алгоритм Наивный метод

int CountNull0(unsigned x) – алгоритм Бинарный поиск

int CountNull1(unsigned x) – алгоритм Двухсторонний подсчёт

int CountNull2(unsigned x) – алгоритм Харли

int CountNull3(unsigned x) – алгоритм Вариация Горявски

int ntz0(unsigned x) – алгоритм Бинарный поиск

int ntz1(unsigned x) – алгоритм Поиск по бинарному дереву

int ntz2(unsigned x) – алгоритм Наивный метод

int ntz3(unsigned x) – алгоритм Годо

int ntz4(unsigned x) – алгоритм Сила

int ntz5(unsigned x) – алгоритм Райзера

int high\_bit\_line\_0(int x) – алгоритм Наивный метод

int high\_bit\_line\_1(int x) – алгоритм Метод с округлениями

int high\_bit\_line\_2(int x) – алгоритм Степени двойки

int high\_bit\_line\_3(int x) – алгоритм Бинарный поиск

int high\_bit\_line\_4(int x) - алгоритм Заполнение

**Тесты**

Для тестирования программы были использованы следующие числа:

Подсчёт количества единиц:

|  |  |  |
| --- | --- | --- |
|  | u8 | u16 |
| Лучший случай | 0 | 0 |
| Средний случай | 200 | 30000 |
| Худший случай | 255 | 65535 |

Подсчёт ведущих нулей:

|  |  |
| --- | --- |
| Лучший случай | 255 |
| Средний случай | 20 |
| Худший случай | 0 |

Подсчёт завершающих нулей:

|  |  |
| --- | --- |
| Лучший случай | 255 |
| Средний случай | 200 |
| Худший случай | 0 |

Нахождение ведущей единиц:

|  |  |
| --- | --- |
| Лучший случай | 1 |
| Средний случай | 200 |
| Худший случай | 1024 |

**Листинг программы**

**main.c**

#include <stdio.h>

#include "bit.c"

#include <time.h>

#define SRC\_ONE 200

#define SRC\_NULL 20

#define SRC\_END\_NULL 200

#define SRC\_START\_ONE 200

int main() {

printf("Подсчёт количества единиц:\n");

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountOnes0(SRC\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Наивный метод: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountOnes1(SRC\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Дуальный алгоритм: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountOnes2\_8(SRC\_ONE); // Менять для количества бит

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Умножение и остаток от деления: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountOnes3\_8(SRC\_ONE); // Менять для количества бит

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Параллельное суммирование: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountOnes4(SRC\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Комбинированный метод: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountOnes5(SRC\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Разделяй и властвуй: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

printf("\nПодсчёт ведущих нулей:\n");

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountNull0(SRC\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Бинарный поиск: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountNull1(SRC\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Двухсторонний подсчёт: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountNull2(SRC\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Алгоритм Харли: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

CountNull3(SRC\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Вариация Горявски: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

printf("\nПодсчёт завершающих нулевых битов:\n");

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

ntz0(SRC\_END\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Бинарный поиск: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

ntz1(SRC\_END\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Поиск по бинарному дереву: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

ntz2(SRC\_END\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Наивный метод: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

ntz3(SRC\_END\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Алгоритм Годо: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

ntz4(SRC\_END\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Алгоритм Сила: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

ntz5(SRC\_END\_NULL);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Алгоритм Райзера: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

printf("\nПоиск старшего единичного бита:\n");

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

high\_bit\_line\_0(SRC\_START\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Наивный метод: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

high\_bit\_line\_1(SRC\_START\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Метод с округлениями: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

high\_bit\_line\_2(SRC\_START\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Степени двойки: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

high\_bit\_line\_3(SRC\_START\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Бинарный поиск: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

{

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

high\_bit\_line\_4(SRC\_START\_ONE);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Заполнение: %ld наносекунд\n", end.tv\_nsec - start.tv\_nsec);

}

return 0;

}

**bit.c**

#ifndef BIT\_C\_

#define BIT\_C\_

#include <math.h>

typedef unsigned char u8;

typedef unsigned short int u16;

typedef unsigned int u32;

typedef unsigned long long u64;

// Подсчёт количества единиц

// Наивный метод

u8 CountOnes0 (u8 n) {

u8 res = 0;

while (n) {

res += n&1;

n >>= 1;

}

return res;

}

// Дуальный алгоритм

u8 CountOnes1 (u8 n) {

u8 res = 0;

while (n) {

res ++;

n &= n-1;

}

return res;

}

// Умножение и остаток от деления

u8 CountOnes2\_8 (u8 n) {

if (n == 0) return 0;

if (n == 0xFF) return 8;

n = (0x010101\*n & 0x249249) % 7;

if (n == 0) return 7;

return n;

}

u8 CountOnes2\_16 (u16 n) {

return CountOnes2\_8((n & 0xFF)) + CountOnes2\_8((n >> 8));

}

// Параллельное суммирование

u8 CountOnes3\_8 (u8 n) {

n = ((n>>1) & 0x55) + (n & 0x55);

n = ((n>>2) & 0x33) + (n & 0x33);

n = ((n>>4) & 0x0F) + (n & 0x0F);

return n;

}

u8 CountOnes3\_16 (u16 n) {

n = ((n>>1) & 0x5555) + (n & 0x5555);

n = ((n>>2) & 0x3333) + (n & 0x3333);

n = ((n>>4) & 0x0F0F) + (n & 0x0F0F);

n = ((n>>8) & 0x00FF) + (n & 0x00FF);

return n;

}

// Комбинированный метод

u8 CountOnes4 (u16 n) {

n -= (n>>1) & 0x5555;

n = ((n>>2) & 0x3333) + (n & 0x3333);

n = ((((n>>4) + n) & 0x0F0F) \* 0x0101) >> 8;

return n;

}

// Разделяй и властвуй

int CountOnes5(unsigned x) {

x = (x & 0x55555555) + ((x >> 1) & 0x55555555);

x = (x & 0x33333333) + ((x >> 2) & 0x33333333);

x = (x & 0x0F0F0F0F) + ((x >> 4) & 0x0F0F0F0F);

x = (x & 0x00FF00FF) + ((x >> 8) & 0x00FF00FF);

x = (x & 0x0000FFFF) + ((x >>16) & 0x0000FFFF);

return x;

}

// Подсчёт ведущих нулей

// Бинарный поиск

int CountNull0(unsigned x) {

int n;

if (x == 0) return(32);

n = 0;

if (x <= 0x0000FFFF) {n = n +16; x = x <<16;}

if (x <= 0x00FFFFFF) {n = n + 8; x = x << 8;}

if (x <= 0x0FFFFFFF) {n = n + 4; x = x << 4;}

if (x <= 0x3FFFFFFF) {n = n + 2; x = x << 2;}

if (x <= 0x7FFFFFFF) {n = n + 1;}

return n;

}

// Двухсторонний подсчёт

int CountNull1(int x) {

int y, n;

n = 0;

y = x;

L: if (x < 0) return n;

if (y == 0) return 32 - n;

n = n + 1;

x = x << 1;

y = y >> 1;

goto L;

}

// Алгоритм Харли

#define u 0

int CountNull2(unsigned x) {

static char table[64] =

{32,31, u,16, u,30, 3, u, 15, u, u, u,29,10, 2, u,

u, u,12,14,21, u,19, u, u,28, u,25, u, 9, 1, u,

17, u, 4, u, u, u,11, u, 13,22,20, u,26, u, u,18,

5, u, u,23, u,27, u, 6, u,24, 7, u, 8, u, 0, u};

x = x | (x >> 1);

x = x | (x >> 2);

x = x | (x >> 4);

x = x | (x >> 8);

x = x | (x >>16);

x = x\*0x06EB14F9;

return table[x >> 26];

}

// Вариация Горявски алгоритма Харли

int CountNull3(unsigned x) {

static char table[64] =

{32,20,19, u, u,18, u, 7, 10,17, u, u,14, u, 6, u,

u, 9, u,16, u, u, 1,26, u,13, u, u,24, 5, u, u,

u,21, u, 8,11, u,15, u, u, u, u, 2,27, 0,25, u,

22, u,12, u, u, 3,28, u, 23, u, 4,29, u, u,30,31};

x = x | (x >> 1);

x = x | (x >> 2);

x = x | (x >> 4);

x = x | (x >> 8);

x = x & ~(x >> 16);

x = x\*0xFD7049FF;

return table[x >> 26];

}

// Подсчёт завершающих нулевых битов

// Бинарный поиск

int ntz0(unsigned x) {

int n;

if (x == 0) return(32);

n = 1;

if ((x & 0x0000FFFF) == 0) {n = n +16; x = x >>16;}

if ((x & 0x000000FF) == 0) {n = n + 8; x = x >> 8;}

if ((x & 0x0000000F) == 0) {n = n + 4; x = x >> 4;}

if ((x & 0x00000003) == 0) {n = n + 2; x = x >> 2;}

return n - (x & 1);

}

// Поиск по бинарному дереву

int ntz1(char x) {

if (x & 15) {

if (x & 3) {

if (x & 1) return 0;

else return 1;

}

else if (x & 4) return 2;

else return 3;

}

else if (x & 0x30) {

if (x & 0x10) return 4;

else return 5;

}

else if (x & 0x40) return 6;

else if (x) return 7;

else return 8;

}

// Наивный метод

int ntz2(unsigned x) {

int n;

x = ~x & (x - 1);

n = 0;

while(x != 0) {

n = n + 1;

x = x >> 1;

}

return n;

}

// Алгоритм Годо

int ntz3(unsigned x) {

unsigned y, bz, b4, b3, b2, b1, b0;

y = x & -x;

bz = y ? 0 : 1;

b4 = (y & 0x0000FFFF) ? 0 : 16;

b3 = (y & 0x00FF00FF) ? 0 : 8;

b2 = (y & 0x0F0F0F0F) ? 0 : 4;

b1 = (y & 0x33333333) ? 0 : 2;

b0 = (y & 0x55555555) ? 0 : 1;

return bz + b4 + b3 + b2 + b1 + b0;

}

// Алгоритм Сила

int ntz4(unsigned x) {

static char table[64] =

{32, 0, 1,12, 2, 6, u,13, 3, u, 7, u, u, u, u,14,

10, 4, u, u, 8, u, u,25, u, u, u, u, u,21,27,15,

31,11, 5, u, u, u, u, u, 9, u, u,24, u, u,20,26,

30, u, u, u, u,23, u,19, 29, u,22,18,28,17,16, u};

x = (x & -x)\*0x0450FBAF;

return table[x >> 26];

}

// Алгоритм Райзера

int ntz5(unsigned x) {

static char table[37] = {32, 0, 1, 26, 2, 23, 27,

u, 3, 16, 24, 30, 28, 11, u, 13, 4,

7, 17, u, 25, 22, 31, 15, 29, 10, 12,

6, u, 21, 14, 9, 5, 20, 8, 19, 18};

x = (x & -x)%37;

return table[x];

}

// Поиск старшего единичного бита

// Наивный метод

int high\_bit\_line\_0(unsigned int n) {

int res = 0;

while (n != 1) {

n >>= 1;

res++;

}

return res;

}

// Метод с округлениями

const double EPS = 1e-11;

inline double log2(double n) {

return log(n) / log(2.0);

}

int high\_bit\_line\_1(unsigned int n) {

return (int) (log2((double) n) + EPS);

}

// Степени двойки

int high\_bit\_line\_2(int x) {

int t = 1 << 30;

while (x < t) t >>= 1;

return t;

}

// Бинарный поиск

int high\_bit\_line\_3(int n) {

int size = sizeof(n) \* 4;

int res = 0;

while (n != 1) {

int l = n >> size;

if (l) {

n = l;

res += size;

} else {

n ^= l << size;

}

size >>= 1;

}

return res;

}

// Заполнение

int high\_bit\_line\_4(int x) {

x |= x >> 1;

x |= x >> 2;

x |= x >> 4;

x |= x >> 8;

x |= x >> 16;

return x - (x >> 1);

}

#endif // BIT\_C\_

**Результаты исследования**

Всего тестировалось для 4 задач. Выполнялось тестирование 3 раза и в результат писалось среднее время.

Подсчёт количества единиц:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | u8 | | |
| Лучший случай | Средний  случай | Худший случай |
| Наивный | (300+544+241)=361.6 | (203+285+189)=225.3 | (292+239+277)=269.3 |
| Дуальный | (79+112+120)=103.6 | (87+103+83)=91 | (113+117+90)=106.6 |
| Умножение и остаток от деления | (52+78+92)=74 | (106+101+78)=95 | (55+66+129)=83.3 |
| Параллельное суммирование | (52+178+98)=109.3 | (84+57+45)=62 | (46+60+51)=52.3 |
| Комбинаторный метод | - | (76+103+170)=116.3 | (344+68+76)=162.6 |
| Разделяй и властвуй | (63+82+213)=119.3 | (55+61+50)=55.3 | (165+52+49)=88.6 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | u16 | | |
| Лучший случай | Средний  случай | Худший случай |
| Наивный | (244+144+236)=208 | (293+580+216)=363 | (222+380+417)=339.6 |
| Дуальный | (76+54+52)=60.6 | (139+198+106)=147.6 | (98+110+114)=107.3 |
| Умножение и остаток от деления | (206+182+180)=189.3 | (119+421+565)=368.3 | (89+252+91)=144 |
| Параллельное суммирование | (191+85+60)=112 | (79+165+82)=108.6 | (84+203+90)=125.6 |
| Комбинаторный метод | (95+53+68)=72 | (83+128+54)=88.3 | (56+206+77)=113 |
| Разделяй и властвуй | (72+50+49)=57 | (51+129+49)=76.3 | (63+216+171)=150 |

Подсчёт ведущих нулей:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Лучший случай | Средний  Случай | Худший случай |
| Бинарный поиск | (98+126+142)=122 | (86+96+80)=87.3 | (172+73+68)=104.3 |
| Двухсторонний подсчёт | (148+144+185)=159 | (107+123+102)=110.6 | (70+76+49)=65 |
| Алгоритм Харли | (230+118+168)=186.6 | (220+195+102)=172.3 | (139+114+83)=112 |
| Вариация Горявски | (65+86+212)=121 | (84+167+86)=112.3 | (356+190+80)=208.6 |

Подсчёт завершающих нулей:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Лучший случай | Средний  Случай | Худший случай |
| Бинарный поиск | (110+160+217)=162.3 | (81+90+93)=88 | (171+58+46)=91.6 |
| Поиск по бинарному дереву | (82+192+205)=159.6 | (70+73+164)=102.3 | (192+90+78)=120 |
| Наивный метод | (63+92+124)=93 | (75+81+79)=78.3 | (164+167+151)=160.6 |
| Алгоритм Годо | (87+197+220)=168 | (85+77+158)=106.6 | (98+133+101)=110.6 |
| Алгоритм Сила | (177+78+81)=112 | (57+51+47)=51.6 | (67+60+67)=64.6 |
| Алгоритм Райзера | (261+75+116)=150.6 | (58+177+60)=98.3 | (62+166+73)=100.3 |

Нахождение ведущей единицы:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Лучший случай | Средний  Случай | Худший случай |
| Наивный метод | (90+75+91)=85.3 | (90+94+84)=89.3 | (109+104+114)=109 |
| Метод с округлениями | (1098+1030+1098)=1075.3 | (6267+6590+5019)=5958.6 | (4771+5953+3927)=4883.6 |
| Степени двойки | (309+379+377)=355 | (266+285+327)=292.6 | (258+293+249)=266.3 |
| Бинарный поиск | (80+77+84)=80.3 | (127+136+122)=128.3 | (130+166+138)=144.6 |
| Заполнение | (95+62+89)=82 | (67+64+66)=65.6 | (67+59+67)=66.3 |