Анализ алгоритмов
Рекурсия

u.to/72D_Gg

Лекция 6, 12 марта, 2021

Лектор:

Дмитрий Северов, кафедра информатики 608 КПМ

dseverov@mail.mipt.ru

Обратная связь: **u.to/7Wn7Gg**

Основы анализа алгоритмов

Алгоритм - точное предписание, задающее процесс преобразования исходных данных в результаты.

- Свойства:
  - *Результативность*
  - *Конечность*
  - *Однозначность*
  - *Массовость*
  - *Детерминированность*
  - *Эффективность*

Обозначения

- M – набор машинных инструкций, составляющих программу;
- S – память для стека и хранения промежуточных результатов;
- d(n) – набор входных данных;
- D(n) = {d(n)} – множество всевозможных наборов входных данных;
- T = {t} – время выполнения программы;

Допущения

- каждая машинная команда выполняется не более чем за фиксированное время;
- рассматриваем правильные и финитные алгоритмы, т.е. алгоритмы, дающие единственное решение общей проблемы

В силу сделанных допущений

- Для любого начального набора данных алгоритм выполняет не более, чем конечное количество «элементарных» операций абстрактной машины.

- Под трудоемкостью F_d алгоритма для данного набора начальных данных – *d(n),* будем понимать количество «элементарных» операций (≡время их выполнения), совершаемых алгоритмом для решения конкретной проблемы, в данной формальной системе.

Цена выполнения программы

$$t = c_1 * d(n) + c_2 * M + c_3 * S,$$

здесь $c_i(d(n))$ – «веса» ресурсов

$T^{\wedge} = \max_{d \in D} T$ – наихудший случай

$T^{v} = \min_{d \in D} T$ – наилучший случай

$\check{T} = \Sigma\, t / |T|$ – средний случай

Классификация алгоритмов по виду функции трудоёмкости

- Параметрически-зависимые (порядко-независимые)

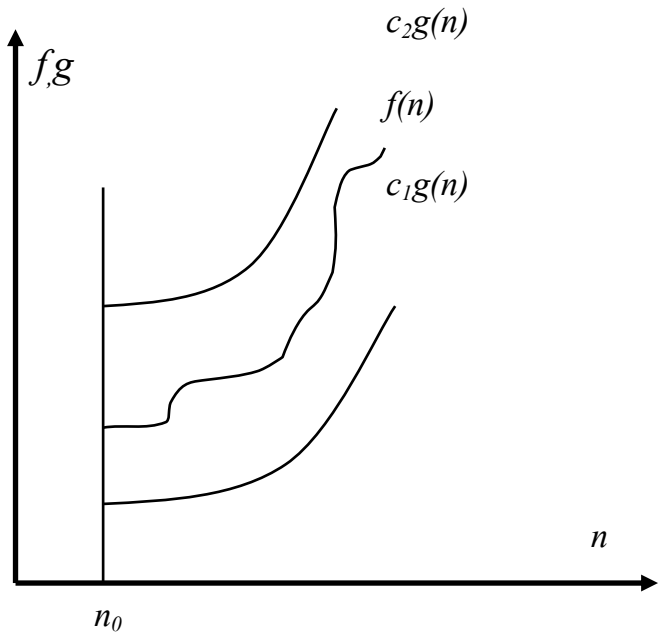
  $F_d = f(n)$; $T^\wedge = T^v = \check{T}$

- Порядко-зависимые

  $F_d = f(d,n)$; $T^v(d,n) \leq \check{T}(n) \leq T^\wedge(d,n)$

Асимптотические оценки

- <u>Θ (тета)</u>



$f(n)$ и $g(n)$: $\exists c_1, c_2, n_0 : \forall n > n_0$

$c_2 * g(n) \le f(n) \le c_1 * g(n) \rightarrow$

$f(n) = \Theta(\ g(n)\ )$

Пример: $F(x) = 4x^2 + \sin(x) = \Theta(x^2)$

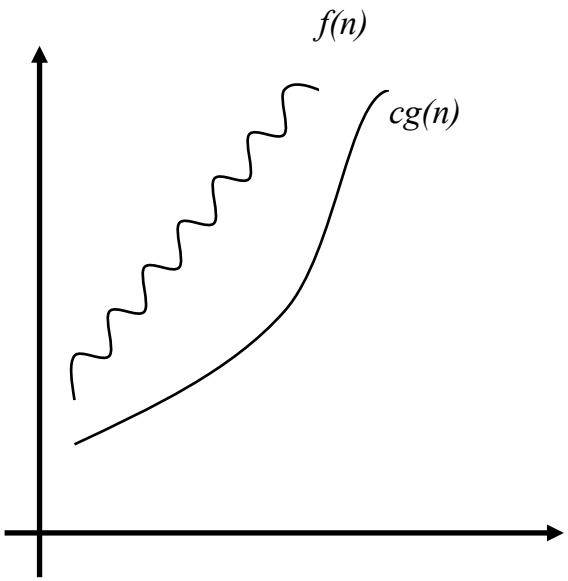
Асимптотические оценки

- <u>Ω (омега)</u>

$f(n)$ и $g(n)$: $\exists c, n_0 : \forall n > n_0$

$c * g(n) \leq f(n) \rightarrow$

$f(n) = \Omega(\ g(n)\ )$



Пример: $f(x) = x * \ln(x) + 1/x = \Omega\ (x * \ln(x))$

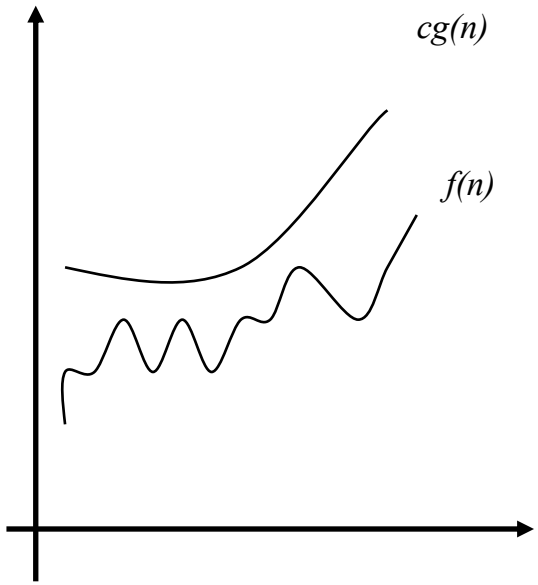
Асимптотические оценки

- <u>O (O большое)</u>

$f(n)$ и $g(n)$: $\exists c, n_0 : \forall n > n_0$

$f(n) \leq c * g(n) \rightarrow$

$f(n) = O(g(n))$



Пример: $f(x) = 4x^2 + x = O(x^2)$

Сравнение времен выполнения алгоритмов

	$n=10$	$n=10^3$	$n= 10^6$
$Log_2 n$	0,2 сек	0,6 сек	1,2 сек
n	0,6 сек	1 мин	16,6 час
n^2	6 сек	16,6 час	1902 года
2^n	1 мин	10^{295} лет	10^{300000} лет

Задача Иосифа

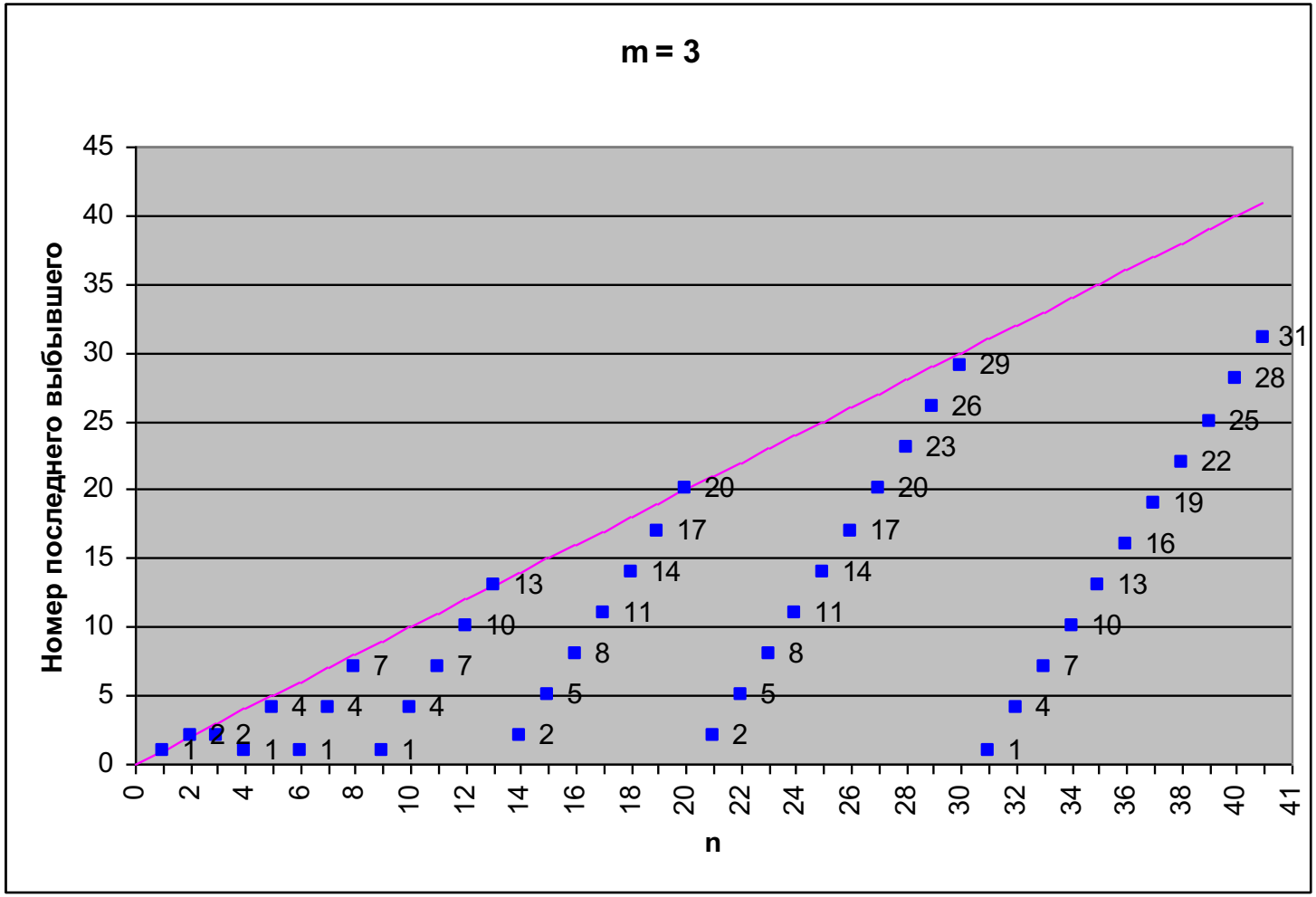
Пусть *n* человек, стоящие по кругу, считаются (начиная с первого) считалкой из *m* слов. Человек, на котором считалка заканчивается, - выбывает, круг смыкается, а счет продолжается с человека, следующего за выбывшим.

Напишите программу, выводящую номера трех человек, выбывших последними, в порядке их выбывания.

Результаты для *m*=3

n	последний	предпоследний	Пред предпоследний
1	1		
2	2	1	
3	2	1	3
4	1	4	2
5	4	2	5
6	1	5	2
7	4	1	5
8	7	4	8
9	1	7	2
10	4	10	5
11	7	2	8
12	10	5	11

Задача Иосифа



$$F(1)=1;\ F(n)=(\ F(n-1)\ +\ m\ )_{\mathrm{mod}\ n}$$

РЕШЕНИЕ 1:
АНАЛИЗ ЗНАЧЕНИЙ ЭЛЕМЕНТОВ МАССИВА

```
#include <iostream>
using namespace std;

unsigned long long start,end;
unsigned long long access_counter() { asm("rdtsc"); }

int main() {
int N,M,i,a,k=0,n;
    cin >> N >> M; M--;
    start = access_counter();
int *x = new int[N];
    for(i=0;i<N;i++) x[i] = i+1; n=N;
    while(n) {  a=M%n--; while(a) { if(x[k++]) a--; if(k==N) k=0; }
                while(!x[k]) { k++; if(k==N) k=0; }
                if(n<3) cout << x[k] << ' '; x[k]=0;
    }
    cout << endl;
    end = access_counter();
    cout << end-start << endl;

    return 0;
}
```

РЕШЕНИЕ 2:
УДАЛЕНИЕ ИЗ МАССИВА ВЫБЫВШИХ ЭЛЕМЕНТОВ

```
#include <iostream>
using namespace std;

unsigned long long start,end;
unsigned long long access_counter() { asm("rdtsc");

int main() {
int N,M,i,a=0;
    cin >> N >> M; M--;
    start = access_counter();
int *x = new int[N];
    for(i=0;i<N;i++) x[i] = i+1;
    while(N) { a=(a+M)%N--; if(N<3) cout << x[a] <<
                for(i=a;i<N;i++) x[i]=x[i+1];
    }
    cout << endl;
    end = access_counter();
    cout << end-start << endl;

    return 0;
}
```

Число сравнений: ~ $m * n * \ln n$

РЕШЕНИЕ 3:
ХРАНЕНИЕ В МАССИВЕ ИНДЕКСОВ СОСЕДНИХ ЭЛЕМЕНТОВ

```
#include <iostream>
using namespace std;

unsigned long long start,end;
unsigned long long access_counter() { asm("rdtsc"); }

int main() {
int N,M,i,n,k;

    cin >> N >> M; M--;
    start = access_counter();
int *x = new int[N];
    for(i=0;i<N-1;i++) x[i]=i+1; x[n=N-1]=0;
    while(N) { k=M%N--; while(k--) n=x[n];
            if(N<3) cout << x[n]+1 << ' '; x[n]=x[x[n]];
    }

    cout << endl;
    end = access_counter();
    cout << end-start << endl;

    return 0;
}
```

D:\A1\Lect2015\Exmp\6\ios_3.cpp

Число сдвигов: $\sim n * n / 4$

РЕШЕНИЕ 4: ДИНАМИЧЕСКИ СОЗДАВАЕМЫЕ СТРУКТУРЫ

```
#include <iostream>
using namespace std;

unsigned long long start,end;
unsigned long long access_counter() { asm("rdtsc"); }

struct item { int n; item *next;
    item(int k, item *nt = NULL) { n=k; next=nt;}
};

int main() {
int N,M,i,n,k;

    cin >> N >> M; M--;
    start = access_counter();
item *p, *t = p = new item(1);
    for(i=2;i<=N;i++) p = ( p->next = new item(i) ); p->next = t;
    while(N) { k=M%N--; while(k--) p = p->next;
                if(N<3) cout << p->next->n << ' '; p->next=p->next->next;
    }

    cout << endl;
    end = access_counter();
    cout << end-start << endl;

    return 0;
}
```

ios_1.cpp | ios_2.cpp | ios_3.cpp | ios_4.cpp

D:\A1\Lect2015\Exmp\6\ios_1.exe

```
10000 20000
8864 8493 9161
788314128
```

D:\A1\Lect2015\Exmp\6\ios_2.exe

```
10000 20000
8864 8493 9161
258553470
```

D:\A1\Lect2015\Exmp\6\ios_4.exe

```
10000 20000
8864 8493 9161
249595766
```

D:\A1\Lect2015\Exmp\6\ios_3.exe

```
10000 20000
8864 8493 9161
231565397
```

РЕШЕНИЕ НА ОСНОВЕ ПОСТРОЕНИЯ РЕКУРРЕНТНОГО СООТНОШЕНИЯ

```
#include <iostream>
using namespace std;

unsigned long long start,end;
unsigned long long access_counter() { asm("rdtsc"); }

int f(int n, int m) { if(n>1) { int p=(f(n-1,m)+m)%n; return p?p:n; }
    else return 1; }
int f1(int n, int m) { if(n>2) { int p=(f1(n-1,m)+m)%n; return p?p:n; }
    else return m%2?1:2; }
int f2(int n, int m) { if(n>3) { int p=(f2(n-1,m)+m)%n; return p?p:n; }
    else return m%3?m%3:3; }

int main() {
int N,M;
    cin >> N >> M;
    start = access_counter();
    cout << f2(N,M) << ' ';
    cout << f1(N,M) << ' ';
    cout << f(N,M) << endl;

    cout << endl;
    end = access_counter();
    cout << end-start << endl;

    return 0;
}
```

D:\A1\Lect2015\Exmp\6\ios_5.exe

```
10000 20000
8864 8493 9161

3068044

--------------------------------
Process exited after 12.6 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Рекурсия

- Способ конечного описания бесконечных объектов.
- Мощный метод решения задач рекуррентной природы.
- Использование стека для организации неявного хранения наборов данных.

- Накладные расходы, связанные с обращением функции к себе.
- Экспоненциальный рост трудоемкости в случае определения одного значения по нескольким другим.
- Сложность поиска ошибок и верификации программ.

$$n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots \cdot n$$

От простого к сложному

$x_1 = 1$ и $x_n = n \cdot x_{n-1}$

От сложного к простому

$n! = n \cdot (n-1)!$ и $1! = 1$

```
int x = 1;
for(int i=2; i<=n; i++) x *= i;
```

Итерации

```
int f(n) = n>1?n*f(n-1):1;
```

Рекурсия

Формальное представление

- Пусть:

P – рекурсивная процедура;

S_i – базовые операции, не содержащие P;

$\mathcal{P}$ – композиция операторов;

- Тогда:

$$P = \mathcal{P}[S_i, P]$$

- Средство создания рекурсивной процедуры - описание функций, которые определяют в процедуре имена, по которым процедура сама себя вызывает.

Классификация и данные

Прямо рекурсивная

$$P = \mathcal{P}[S_i, P]$$

Косвенно рекурсивная

$$P = \mathcal{P}[S_i, Q]$$

$$Q = \mathcal{Q}[S_i, P]$$

Переменные, определённые внутри рекурсивной функции, создаются заново при каждом вызове такой функции.

Идентификаторы всегда ссылаются на переменные, созданные последними.

Важно !

- Проблема окончания работы.

$$P = \mathcal{P}[\ S_i,\ \underline{\textbf{if}}\ !B\ \underline{\textbf{then}}\ P]$$

Наиболее надежный способ окончания работы – связать с P параметр *n* и рекурсивно вызывать P с параметром *n-1*:

$$P(n) = \mathcal{P}[S_i,\ \underline{\textbf{if}}\ n>0\ \underline{\textbf{then}}\ P(n-1)]$$

- Глубина рекурсии.

Следует убедиться, что она не только конечна, но и достаточно мала.

Примеры использования рекурсии при решении задач

ПОИСК ПУТИ В ЛАБИРИНТЕ

NT.CPP | My_str.h

```
#include <string.h>
#include <malloc.h>

class My_string { char* s; int l;
public:
My_string(const char* p1=NULL, const char* p2=NULL) { char* b;
    if(p1 == NULL ) { s = NULL; l = 0; return; }
    else if(p2 == NULL) { p2 = p1; while(*p2++); p2--; }
    l = p2 - p1; b = s = new char[l+1];
    while(p1<p2) *b++ = *p1++; *b='\0'; }

void operator=(const char* x) { delete s; s = new char[(l=strlen(x))+1]; strcpy(s,x); }

void operator=(const My_string& x) { delete s; s = new char[(l=x.l)+1]; strcpy(s,x.s); }

My_string operator+(const My_string& a) {My_string loc;
    loc.l=l+a.l; loc.s = new char [loc.l+1]; strcpy(loc.s,s);
    strcat(loc.s,a.s); return loc; }

char& operator[](int i) { return s[i]; }

bool operator==(const char* x) { for(int i=0;i<=l;i++)
 if(s[i]!=x[i]) return false; return true; }

bool operator==(const My_string& x) { if(l!=x.l) return false;
    for(int i=0;i<l;i++) if(s[i]!=x.s[i]) return false; return true; }
```

своя строка

создать свою от начала до конца

присвоить стандартную

присвоить свою

добавить свою

выбрать символ в своей

равна ли стандартной

равна ли своей

NT.CPP | My_str.h

```
bool operator<(const My_string& x) {
    for(int i=0;i<=l;i++) if(s[i]==x.s[i]) continue; else return s[i]<x.s[i];
    return false; }

bool operator>(const My_string& x) {
    for(int i=0;i<=l;i++) if(s[i]==x.s[i]) continue; else return s[i]>x.s[i];
    return false; }

int size() { return l; }

int find(const My_string& x) {
    for(int i=0; i<l-x.l; i++) if(!strncmp(s+i,x.s,x.l)) return i;
    return -1; }

~My_string() { delete s; l = 0; }

friend ostream& operator<<(ostream& a, const My_string& x) { return a << x.s; }

friend istream& operator>> (istream& b, My_string& a) {
const int n=20; char c,s[n];
    delete a.s; *(a.s = new char)='\0'; a.l=0;
    if(!b.get(c) || c=='\n') return b;
    do { b.putback(c); b.get(s,n); a.s=(char*)realloc(a.s,(a.l+=strlen(s))+1);
        strcat(a.s,s); b.get(c); } while(c!='\n');
    return b; }
};
```

меньше ли своей

больше ли своей

измерить свою

найти свою в своей

удалить свою

вывести свою

ввести свою

LABIRINT.CPP | My_str.h

```
#include <iostream>
#include <fstream>

using namespace std;

#include "My_str.h"

int N,M,F=0,X,Y;
My_string* A;
int** B;

void array(istream& in,int n=0) { My_string a;
    if(in >> a) array(in,n+1);
    else { A = new My_string[N=n]; return; }
    A[n] = a;
}

void print_way(int n,int m,int l) {
    cout << n << ' ' << m << endl; A[n][m]='*';
    if(!--l) return;
    if(n && B[n-1][m]==l) n--;
    else if(n+1<N && B[n+1][m]==l) n++;
        else if(m && B[n][m-1]==l) m--; else m++;
    print_way(n,m,l);
}
```

для карты лабиринта

для расстояний от начала

ввести массив своих

отметить обратный путь

отметить точку пути

сместиться к началу

отметить оставшийся путь

LABIRINT.CPP | My_str.h

```
void next(int n,int m,int l) {
    if( n<0 || m<0 || n>=N || m>=M) return;
    if( A[n][m]=='.' && (!B[n][m] || l<B[n][m]) ) { B[n][m]=l; next(n-1,m,l+1);
        next(n,m+1,l+1); next(n+1,m,l+1); next(n,m-1,l+1); }
    else if( A[n][m]=='E' && (!B[n][m] || l<B[n][m]) ) { B[n][m]=l; F++; X=n; Y=m; }
    return;
}

int main(int argn, char** argv) {
int n,m;

   ifstream in(argv[1]);
   if(!in) { cerr << "Cann't open file"; return 1;}

   array(in); for(n=0;n<N;n++) cout << A[n] << endl; M=A[0].size();
   B = new int*[N];
   for(n=0;n<N;n++) { B[n]=new int[M]; for(m=0;m<M;m++) B[n][m]=0; }

   next(0,0,1);
   if(!F) cout << "No way\n"; else print_way(X,Y,B[X][Y]);

   for(n=0;n<N;n++) cout << A[n] << endl;

   return 0;
}
```

дальше

некуда

ещё дальше

дошли

начать

вывести легенду

вывести карту

D:\A1\Lect2013\Exmp\8\LABIRINT....

```
.......................#
...................#...#.
..................#...#..
.................#...#...
................#...#....
...............#...#.....
..............#...#......
.............#...#.......
............#...#........
...........#...#.........
..........#...#..........
.........#...#...........
........#...#............
.......#...#.............
......#...#..............
.....#...#...............
....#...#................
...#...#.................
..#...#..................
.#...#...................E
#........................
18 23
19 23
19 22
19 21
19 20
19 19
19 18
19 17
19 16
19 15
19 14
19 13
```

D:\A1\Lect2013\Exmp\8\LABIRINT....

```
0 11
0 10
0 9
0 8
0 7
0 6
0 5
0 4
0 3
0 2
0 1
0 0
***********************...#
...................#*..#.
..................#**.#..
.................#**.#...
................#**.#....
...............#**.#.....
..............#**.#......
.............#**.#.......
............#**.#........
...........#**.#.........
..........#**.#..........
.........#**.#...........
........#**.#............
.......#**.#.............
......#**.#..............
.....#**.#...............
....#**.#................
...#**.#.................
..#.*.#..................
.#..*#...................*
#...***********************
```

D:\A1\Lect2013\Exmp\8\LABIRINT.exe

```
.....................#......#
...................#...#...#...#.
..................#...#...#...#..
.................#...#...#...#...
................#...#...#...#....
...............#...#...#...#.....
..............#...#...#...#......
.............#...#...#...#.......
............#...#...#...#........
...........#...#...#...#.........
..........#...#...#...#..........
.........#...#...#...#...........
........#...#...#...#............
.......#...#...#...#.............
......#...#...#...#..............
.....#...#...#...#...............
....#...#...#...#................
...#...#...#...#.................
..#...#...#...#..................
.#...#...#...#...................
#......#.....................E
19 31
19 30
19 29
```

D:\A1\Lect2013\Exmp\8\LABIRINT.exe

```
0 2
0 1
0 0
*******************...#.***...#
..................#*..#.**#*..#.
.................#**.#.**#**.#..
................#**.#.**#**.#...
...............#**.#.**#**.#....
..............#**.#.**#**.#.....
.............#**.#.**#**.#......
............#**.#.**#**.#.......
...........#**.#.**#**.#........
..........#**.#.**#**.#.........
.........#**.#.**#**.#..........
........#**.#.**#**.#...........
.......#**.#.**#**.#............
......#**.#.**#**.#.............
.....#**.#.**#**.#..............
....#**.#.**#**.#...............
...#**.#.**#**.#................
..#.*.#.**#.*.#.................
.#..*#.**#..*#..................
#...****#...*******************
```

Сколько разбиений числа m на слагаемые ?

6

5 + 1

4 + 2	4 + 1 + 1	
3 + 3	3 + 2 + 1	3 + 1 + 1 + 1
2 + 2 + 2	2 + 2 + 1 + 1	2 + 1 + 1 + 1 + 1

1 + 1 + 1 + 1 + 1 + 1

P(6) = 11

Сколько разбиений m на слагаемые $\leq$ n

- $P(n) = Q(n,n)$
- $Q(m,1) = 1$ $\quad 1 + 1 + \ldots + 1 + 1$
- $Q(1,n) = 1$ $\quad \forall n$
- $Q(m,n) = Q(m,m)$ $\quad \forall n \geq m$
- $Q(m,m) = 1 + Q(m,m-1)$
- $Q(m,n) = Q(m-n,n) + Q(m,n-1) \quad \forall n < m$

Разбиения, содержащие слагаемое, равное n

Разбиения, не содержащие слагаемое, равное n

```
#include <iostream>
using namespace std;

int Q(int m,int n) {
    if( m==1 || n==1 ) return 1;
    if (n>=m) return Q(m,m-1)+1;
    return Q(m-n,n)+Q(m,n-1);
}

int main() {
    int N;
    while( cin >> N ) cout << Q(N,N) << endl << endl;

    return 0;
}
```

```
6
11

15
176

20
627

^Z
```

ЧИСЛА ФИБОНАЧЧИ

Или о том, что рекурсию следует использовать весьма осмотрительно…

Вычисление чисел Фибоначчи

Итеративный	Рекурсивный

`long F(int n) {`
`  long a=1,b=1;`
`  do { a+=b; b=a-b; }`
`  while(--n>2);`
`return a; }`
	`long F(int n) {`
	`  return`
	`  n<3 ? 1:F(n-1)+F(n-2);`
	`}`

```
#include <iostream>
#include <iomanip>

using namespace std;

unsigned long long start, end;
unsigned long long access_counter() { asm("rdtsc"); }

long IF(int n) {long a=1,b=1;
        do { a+=b; b=a-b; } while(--n>2);
        return a;
}

long RF(int n) {
        return n<3?1:RF(n-1)+RF(n-2);
}

int main() {

        start = access_counter();
        cout << "IF(45) = " << IF(45);
        end = access_counter();
        cout << " Time was:     " << setw(10) << setfill(' ') << (end - start)/1.4e6 << " ms\n";

        start = access_counter();
        cout << "RF(45) = " << RF(45);
        end = access_counter();
        cout << " Time was:     " << setw(10) << setfill(' ') << (end - start)/1.4e9 << " s\n";

        return 0;
}
```

```
D:\A1\Lect2013\Exmp\8>a
IF(45) = 1134903170 Time was:        0.46141 ms
RF(45) = 1134903170 Time was:        28.9569 s
```

ФУНКЦИЯ АККЕРМАНА

Или о том, когда рекурсию не следует использовать вообще…

Функция Аккермана

$$A(m,n) = \begin{cases} n+1, & m=0 \\ A(m-1,1), & n=0 \\ A\big(m-1, A(m,n-1)\big), & m,n>0 \end{cases}$$

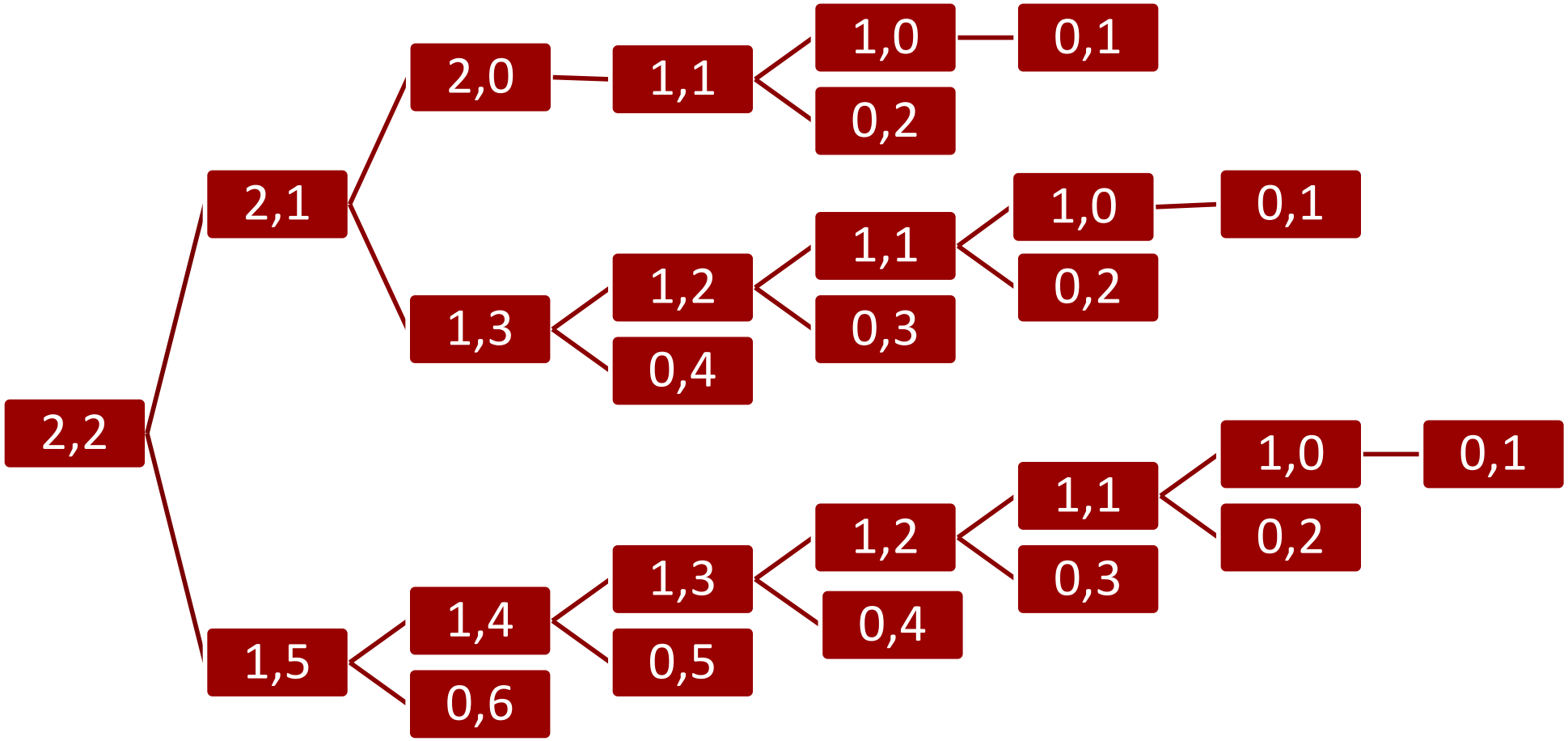
```
#include <iostream>
int A(int m,int n) { return
  !m ? n+1 : !n ? A(m-1,1) : A(m-1,A(m,n-1)); }
void main() { int n,m;
  cout << "Input m="; cin >> m;
  cout << "Input n="; cin >> n;
  cout << "A("<< m << ',' << n << «)= " << A(m,n) <<endl;
}
```

A(2,2) = A(1, A(2,1))

m\n	0	1	2	3	4	5	6	7	8	…
0	1	2	3	4	5	6	7	8	9	n+1
1										n+2
2										2n+3
3										2^{n+3}-3
4										
…										

$$A(m,n) = \begin{cases} n+1, & m = 0 \\ A(m-1,1), & n = 0 \\ A\big(m-1, A(m,n-1)\big), & m,n > 0 \end{cases}$$

Дерево вычисления функции A(2,2)

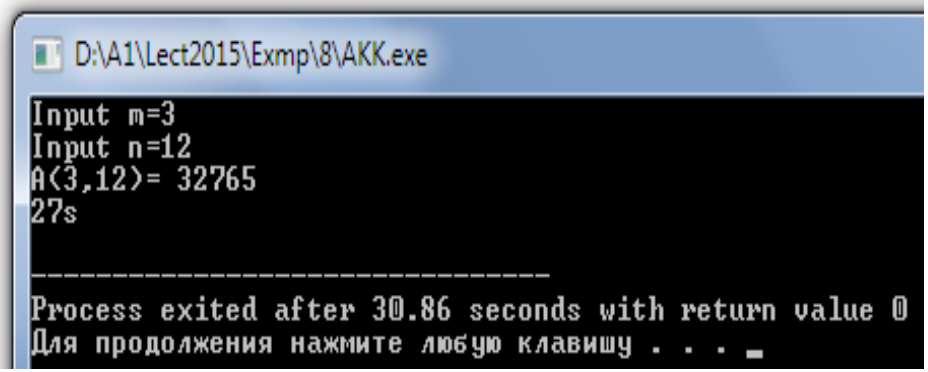


A(2,2)=7 Количество вершин дерева - 27

```
#include <iostream>
#include <time.h>
using namespace std;
typedef int T;

clock_t start,end;
T A(T m,T n) { return !m?n+1:!n?A(m-1,1):A(m-1,A(m,n-1)); }

int main() {
T n,m;
    cout << "Input m="; cin >> m;
    cout << "Input n="; cin >> n;
    start = clock();
    cout << "A(" << m << ',' << n << ")= " << A(m,n) << endl;
    end = clock();
    cout << (end-start)/CLK_TCK << "s\n";
    return 0;
}
```



Профилировка

Плоский вывод | Граф вызовов | Profiling Options

Function name	Index	% time	Self	Children	Called
<spontaneous>					
_mcount_private [1]	[1]	52.7	6.55	0.00	_mcount_private
A(int, int) [2]			715664090	A(int,	int)
main [3]			5.69	0.00	1/1
0 A(int, int) [2]	[2]	45.8	5.69	0.00	1+715664090
A(int, int) [2]			715664090	A(int,	int)
<spontaneous>					
main [3]	[3]	45.8	0.00	5.69	main
A(int, int) [2]			5.69	0.00	1/1
<spontaneous>					
_fentry__ [4]	[4]	1.5	0.19	0.00	_fentry__

```
Input m=3
Input n=13

--------------------------------
Process exited after 57.62 seconds with return value 32212
Для продолжения нажмите любую клавишу . . .
```

Биномиальные коэффициенты

$$C_n^k = \frac{n!}{k!\,(n-k)!}$$

$$C_{n+1}^k = C_n^k + C_n^{k-1}$$

1. На основе определения

```
int fact(int n) {
  switch (n) {
  case 0:
   case 1: return 1;
  default: return n*fact(n-1); }     }
                    …
for(n=0; n<=N; n++) for(k=0; k<=n; k++)
  cout << fact(n)/fact(k)/fact(n-k) <<
 (n==k?'\n':' ');
```

2. На основе рекуррентного соотношения

```
int Cnk(int n,int k) {
    if(!k) return 1;
    if(!n) return 1;
    if(n==k) return 1;
    return Cnk(n-1,k)+Cnk(n-1,k-1);
}
```

3. Треугольник Паскаля

```
1          ⓪
1 1        ①
1 2 1      ②
1 3 3 1    ③
1 4 6 4 1  ④
```

```
void C(int N) {
int *a, i, n;
    a = new int[++N];
    a[0]=1; if(N>1) a[1]=1;
    for(n=3; n<=N; n++) {
      for(i=n-2,a[n-1]=1; i>0; i--)
        a[i]+=a[i-1];
      for(i=0;i<n;i++)
        cout <<a[i]<<(i==n-1?'\n':' '); }
    delete a;
}
```

```
#include <iostream>
using namespace std;


unsigned long long start, end;
unsigned long long access_counter() { asm("rdtsc"); }


void C(int);


int fact(int n) {
    switch (n) {
            case 0:
            case 1: return 1;
            default: return n*fact(n-1);
            }
}


int Cnk(int n,int k) {
    if(!k) return 1;
    if(!n) return 1;
    if(n==k) return 1;
    return Cnk(n-1,k)+Cnk(n-1,k-1);
}
```

```
int main() {
 int b=0,N; T r,n,k;
     cin >> N;
     for(n=0; n<=N && !b; n++) for(k=0; k<=n; k++) {
                 cout << (r=fact(n)/fact(k)/fact(n-k)) << (k==n?'\n':' ');
                 if(k==1) b=r-n; }
     if(b) cout << "!!! ERROR !!!\n";

     start=access_counter();
     for(n=0; n<=N; n++) for(k=0; k<=n; k++) cout << Cnk(n,k) << (k==n?'\n':' ');
     end=access_counter();
     cout << "Elapsed time 1 = " << (end - start)/1.4e9 << endl;

     start=access_counter();
     C(N);
     end=access_counter();
     cout << "Elapsed time 2 = " << (end - start)/1.4e9 << endl;
     return 0;
}

void C(int N) {
 T i, n, *a = new T[++N];
     a[0]=1; if(N>1) a[1]=1;
     for(n=3; n<=N; n++) { for(i=n-2,a[n-1]=1; i>0; i--) a[i]+=a[i-1];
                           for(i=0;i<n;i++) cout << a[i] << (i==n-1?'\n':' '); }
     delete a;
}
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1
1 12 66 220 495 792 924 792 495 220 66 12 1
1 4 24 88 221 399 532 532 399 221 88 24 4 1
!!! ERROR !!!
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1
1 12 66 220 495 792 924 792 495 220 66 12 1
1 13 78 286 715 1287 1716 1716 1287 715 286 78 13 1
1 14 91 364 1001 2002 3003 3432 3003 2002 1001 364 91 14 1
1 15 105 455 1365 3003 5005 6435 6435 5005 3003 1365 455 105 1
```

lst — Блокнот

Файл Правка Формат Вид Справка

```
1 21 210 1330 5985 20349 54264 116280 203490 293930 352716 352716 293930 203490 116280 54264 20349 5985 1330 210 21 1
1 22 231 1540 7315 26334 74613 170544 319770 497420 646646 705432 646646 497420 319770 170544 74613 26334 7315 1540 231 22 1
1 23 253 1771 8855 33649 100947 245157 490314 817190 1144066 1352078 1352078 1144066 817190 490314 245157 100947 33649 8855 1771 253 2
1 24 276 2024 10626 42504 134596 346104 735471 1307504 1961256 2496144 2704156 2496144 1961256 1307504 735471 346104 134596 42504 1062
1 25 300 2300 12650 53130 177100 480700 1081575 2042975 3268760 4457400 5200300 5200300 4457400 3268760 2042975 1081575 480700 177100
1 26 325 2600 14950 65780 230230 657800 1562275 3124550 5311735 7726160 9657700 10400600 9657700 7726160 5311735 3124550 1562275 65780
1 27 351 2925 17550 80730 296010 888030 2220075 4686825 8436285 13037895 17383860 20058300 20058300 17383860 13037895 8436285 4686825
1 28 378 3276 20475 98280 376740 1184040 3108105 6906900 13123110 21474180 30421755 37442160 40116600 37442160 30421755 21474180 13123
1 29 406 3654 23751 118755 475020 1560780 4292145 10015005 20030010 34597290 51895935 67863915 77558760 77558760 67863915 51895935 345
1 30 435 4060 27405 142506 593775 2035800 5852925 14307150 30045015 54627300 86493225 119759850 145422675 155117520 145422675 11975985
Elapsed time 1 = 67.9953
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1
1 12 66 220 495 792 924 792 495 220 66 12 1
1 13 78 286 715 1287 1716 1716 1287 715 286 78 13 1
1 14 91 364 1001 2002 3003 3432 3003 2002 1001 364 91 14 1
1 15 105 455 1365 3003 5005 6435 6435 5005 3003 1365 455 105 15 1
1 16 120 560 1820 4368 8008 11440 12870 11440 8008 4368 1820 560 120 16 1
1 17 136 680 2380 6188 12376 19448 24310 24310 19448 12376 6188 2380 680 136 17 1
1 18 153 816 3060 8568 18564 31824 43758 48620 43758 31824 18564 8568 3060 816 153 18 1
1 19 171 969 3876 11628 27132 50388 75582 92378 92378 75582 50388 27132 11628 3876 969 171 19 1
1 20 190 1140 4845 15504 38760 77520 125970 167960 184756 167960 125970 77520 38760 15504 4845 1140 190 20 1
1 21 210 1330 5985 20349 54264 116280 203490 293930 352716 352716 293930 203490 116280 54264 20349 5985 1330 210 21 1
1 22 231 1540 7315 26334 74613 170544 319770 497420 646646 705432 646646 497420 319770 170544 74613 26334 7315 1540 231 22 1
1 23 253 1771 8855 33649 100947 245157 490314 817190 1144066 1352078 1352078 1144066 817190 490314 245157 100947 33649 8855 1771 253 2
1 24 276 2024 10626 42504 134596 346104 735471 1307504 1961256 2496144 2704156 2496144 1961256 1307504 735471 346104 134596 42504 1062
1 25 300 2300 12650 53130 177100 480700 1081575 2042975 3268760 4457400 5200300 5200300 4457400 3268760 2042975 1081575 480700 177100
1 26 325 2600 14950 65780 230230 657800 1562275 3124550 5311735 7726160 9657700 10400600 9657700 7726160 5311735 3124550 1562275 65780
1 27 351 2925 17550 80730 296010 888030 2220075 4686825 8436285 13037895 17383860 20058300 20058300 17383860 13037895 8436285 4686825
1 28 378 3276 20475 98280 376740 1184040 3108105 6906900 13123110 21474180 30421755 37442160 40116600 37442160 30421755 21474180 13123
1 29 406 3654 23751 118755 475020 1560780 4292145 10015005 20030010 34597290 51895935 67863915 77558760 77558760 67863915 51895935 345
1 30 435 4060 27405 142506 593775 2035800 5852925 14307150 30045015 54627300 86493225 119759850 145422675 155117520 145422675 11975985
Elapsed time 2 = 0.000351295
```

Стр 1, стлб 1

Правильные скобочные выражения

()

()() (())

()()() (())() ()(()) (()()) ((()))

Вставка «()» до и после «(« до первой закрывающейся скобки «)»
Избегая повторений вставки делаем перед каждой открывающейся скобкой и перед первой закрывающейся

```
#include <iostream>
using namespace std;
#include "My_str.h"

My_string a("()");
int K=0;

void next(int L) {
int k = a.size(),i=0;
    if(k==L) { K++; cout << a << endl; return; }
    while( a[i]=='(' ) { My_string b(&a[0],&a[i]);
                         My_string e(&a[i++]);
                         My_string old(&a[0]);
                         a = b + "()" + e;
                         next(L);
                         a = old; }
    My_string b(&a[0],&a[i]);
    My_string e(&a[i]);
    a = b + "()" + e;
    next(L);
}
int main() { int L; cin >> L;
    next(L); cout << K << endl;
    return 0;
}
```