Первая пара. Динамическое пр-е и рекурсия

Григорьева Анастасия Викторовна
мат-мех + Академическая гимназия СПбГУ

Тел. +7(904)646-03-15
Личные сообщения в vk:
https://vk.com/id969

С этой пары и далее

C++

Python останется вашим любимым калькулятором. То есть выручалкой для простых маленьких задач.

Почему C++?

Когда попадаешь на заключительный этап Всероссийской олимпиады школьников, почему-то оказывается, что 95% участников используют именно C++

Краткий синтаксис C++

Тут собраны самые распространенные операторы для начинающих:

https://docs.google.com/document/d/1fB68AchuPRgxv2kd39e_r3YIiRfMpyc0YFR4ZqKDEZM/edit

А именно…

```
min(a,b)
min(a, min(b,c))

#include <vector>
vector <long long> a(n);

int a, b; //целочисленные переменные
char g[1000]; массив из символов


Точка с запятой после каждого оператора

a = 6    //В переменную а запишется 6

a==6; //А так сравнится, не равно ли а шести.
```

```
/* Это комментарий
на несколько
строк*/

cin >>a >>b;

cout<< a+b;

Остаток от деления на 5:
d%5

Оператор условия
if ( (a>7) && (a%2==0) )
{
...
}
else
{
...
}

Имперсанты логических операций:
||   или
&& и
```

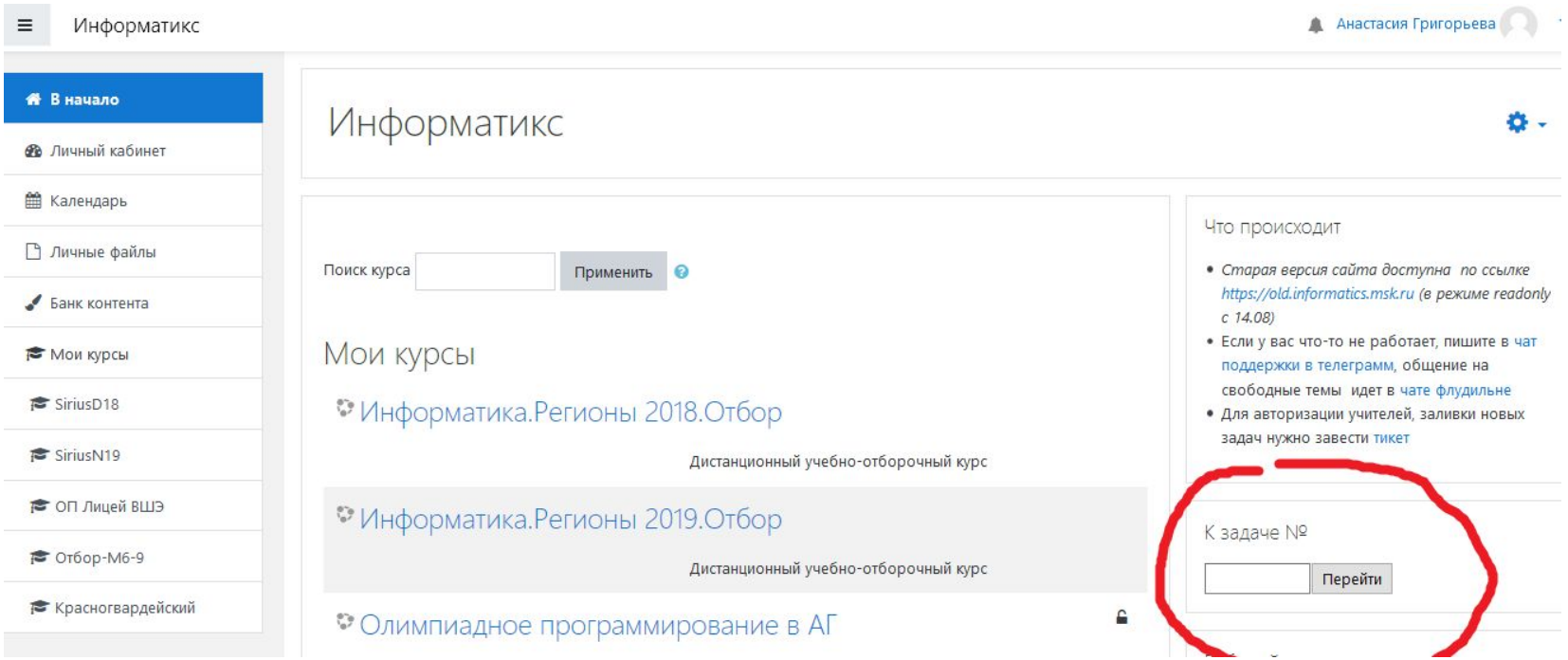
```
Цикл пошаговый:
for (int i=0; i<N; i++)
{
...
}

Или вот цикл "пока"
while (i<20)
{
  i = i +1;
}

Чтобы не закрывалось окно консольного
приложения, в конце программы добавить:
system("pause");
```

Синтаксис C++

```
cin>>a>>b;
cout<<y<<o;

if (…) {…;}
else {…;}
while (…) {…}

for (int i=0; i<N; i++) {…}

int A[n];
int A[100] = {0}
```

Инструменты

- Он-лайн компилятор например www.onlinegdb.com/
- informatics.msk.ru



Рекурсия

Лозунг

Рекурсия – мы хотим, чтобы наш код повторял какую-то мысль человека.

Рекурсия

- Чтобы понять рекурсию, надо понять рекурсию
- Не забывайте условие выхода из рекурсии
- Не забывайте возвращать значения из рекурсивной функции
- Вербовка по телефону
- Какая самая типичная задача для рекурсии?

Типичные задачи на рекурсию

- n!
- Числа Фибоначчи
- Ханойские башни
- Сортировки (**QuickSort, MergeSort**)
  Самые эффективные из универсальных – рекурсивные
- Быстрое возведение в степень
- Множество других

Первый пример. Факториал

n!

*n!=1*2*3*...*n. С другой стороны,*

$$n! = \begin{cases} 1, & \text{если } n \le 1, \\ (n-1)! \times n, & \text{если } n > 1. \end{cases}$$

```
double Factorial(int N)
{
  double F;
  if (N<=1) F=1.; else F=Factorial(N-1)*N;
  return F;
}
```

Еще пример

Определим функцию K(n), которая возвращает количество цифр в заданном натуральном числе n:

$$K(n) = \begin{cases} 1, & \text{если } n < 10, \\ K(n/10) + 1, & \text{если } n >= 10. \end{cases}$$

```
int K(int N)
 {
   int Kol;
   if (N<10) Kol=1; else Kol=K(N/10)+1;
   return Kol;
 }
```

Нод

Рекурсивная реализация алгоритма Евклида

```
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

Нерекурсивная реализация алгоритма Евклида

```
int gcd(int a, int b) {
    while (b > 0) {
        a %= b;
        swap(a, b);
    }
    return a;
}
```

Алгоритм быстрого возведения в степень

Применяя алгоритм быстрого возведения в степень для $595^{703} \pmod{991}$:

Имеем $n = 703 = (1010111111)_2 = 2^0+2^1+2^2+2^3+2^4+2^5+2^7+2^9$.

$595^{703} = (((((((595^2)^2*595)^2)^2* 595)^2*595)^2 * 595)^2*595)^2*595)^2*595$

$= (((((((238^2*595)^2)^2* 595)^2*595)^2 * 595)^2*595)^2*595)^2*595$

$= ((((((261^2)^2* 595)^2*595)^2 * 595)^2*595)^2*595)^2*595$

$= (((((733^2* 595)^2*595)^2 * 595)^2*595)^2*595)^2*595$

$= (((((167* 595)^2*595)^2 * 595)^2*595)^2*595)^2*595$

$= ((((265^2*595)^2 * 595)^2*595)^2*595)^2*595$

$= (((342^2 * 595)^2*595)^2*595)^2*595$

$= ((605^2*595)^2*595)^2*595$

$= (733^2*595)^2*595$

$= (167*595)^2*595$

$= 265^2*595$

$= \mathbf{342}$.

$$33^{52} = \left(\left(33 \cdot \left((33 \cdot 33^2)^2\right)^2\right)^2\right)^2$$

$$a^k = \begin{cases} \left(a^{k/2}\right)^2, & \text{если } k \text{ чётное} \\ a \cdot a^{k-1}, & \text{если } k \text{ нечётное} \end{cases}$$

Числа Фибоначчи №201

- Рекурсия – не волшебная пилюля
- Запустите кто-нибудь сейчас считать Числа Фибоначчи рекурсивно.

- Пятое (5), десятое (55), тридцатое (832040), 45-ое (1134903170).

- Эта задача решается с помощью ДП

Факториал №351

- 10! = 3628800
- Unsignet long fact(int n) {…}

Выражение № 612

В блокноте + на доске

Динамическое программирование

начало

Лозунг ДП

Действуем максимально лениво! Не пересчитываем то, что когда-то уже посчитали.

ДП – решение сложных задач через более простые

Схема. Всегда в голове.

- База
- Переход
- Общая задача = по структуре маленьким
- Что есть ответ?

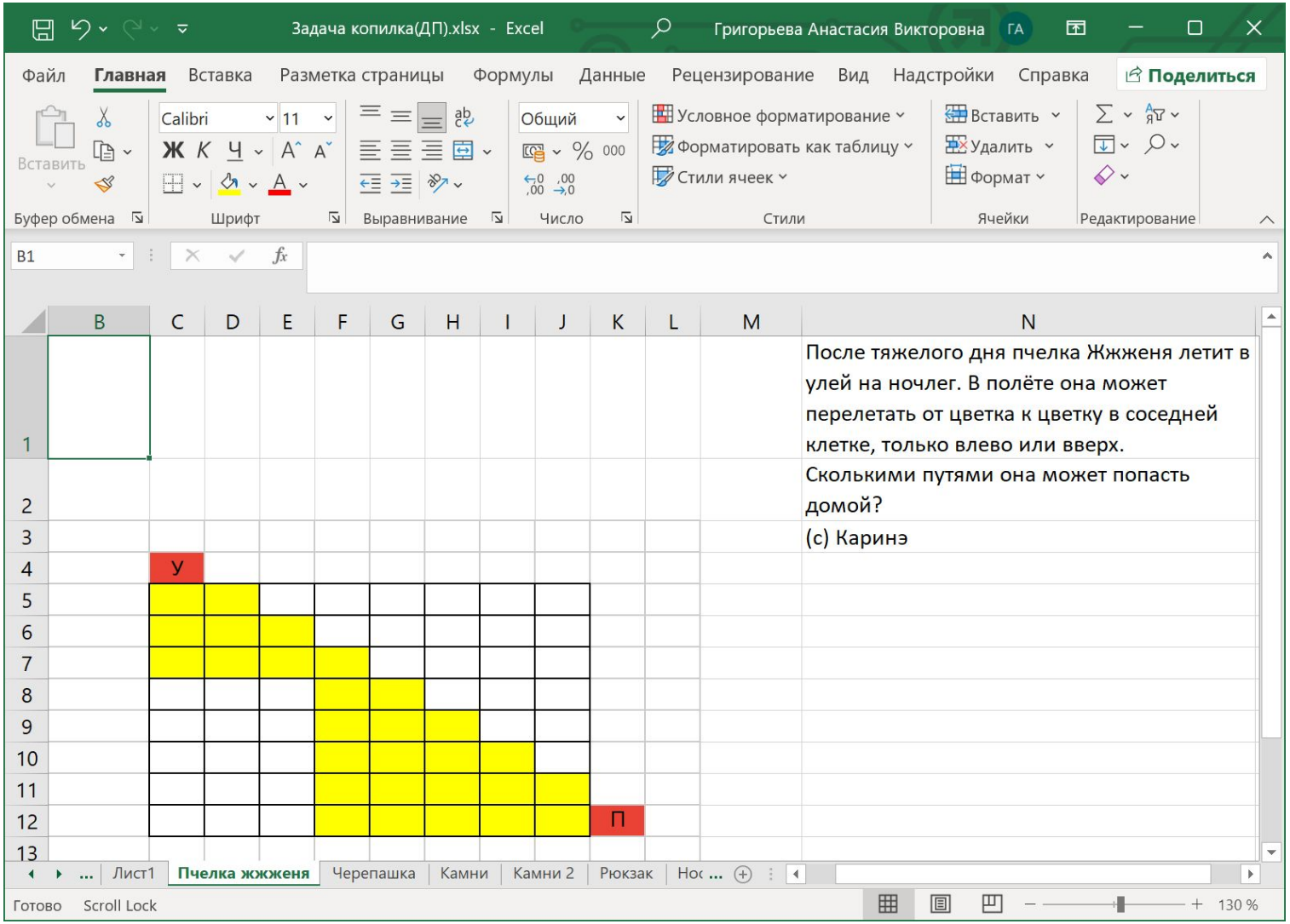
Для сильно опережающих

- Спиралька № 1470
- Ханойские башни №3050
- Рюкзак с выводом № 3090
- Таблица №1150

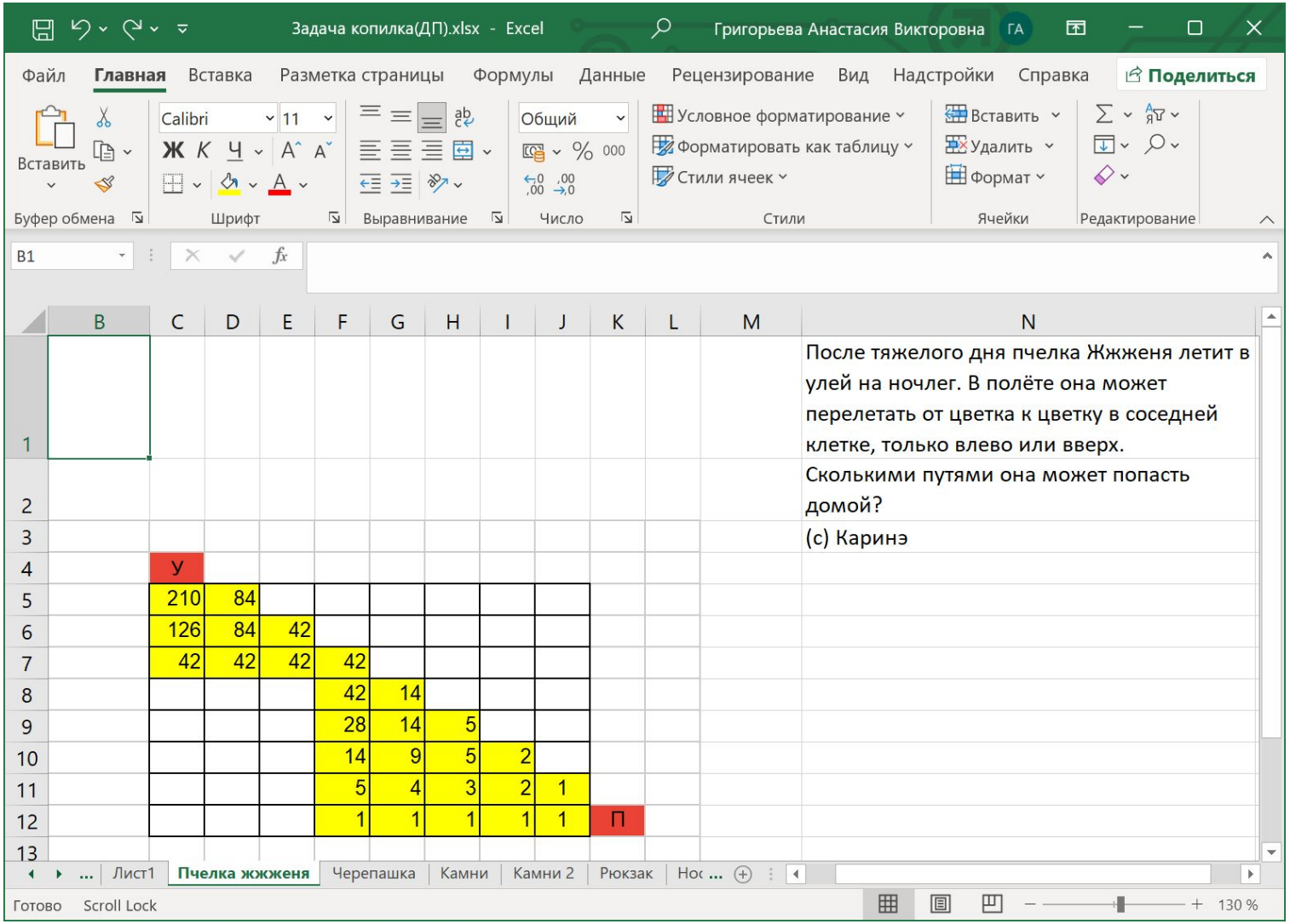
Снова Числа Фибоначчи №201

- Можно и в массив, главное – не пересчитывайте заново с первого
- Можно тремя переменными
- Как поменять два числа? Какие способы вы знаете? Помните как мы вычисляли среднее по величине из трёх?
- Какое число вмещается в тип int? 49ое вместится?

Пчелка Жжжженя



Пчелка Жжжженя



Камни. 1119

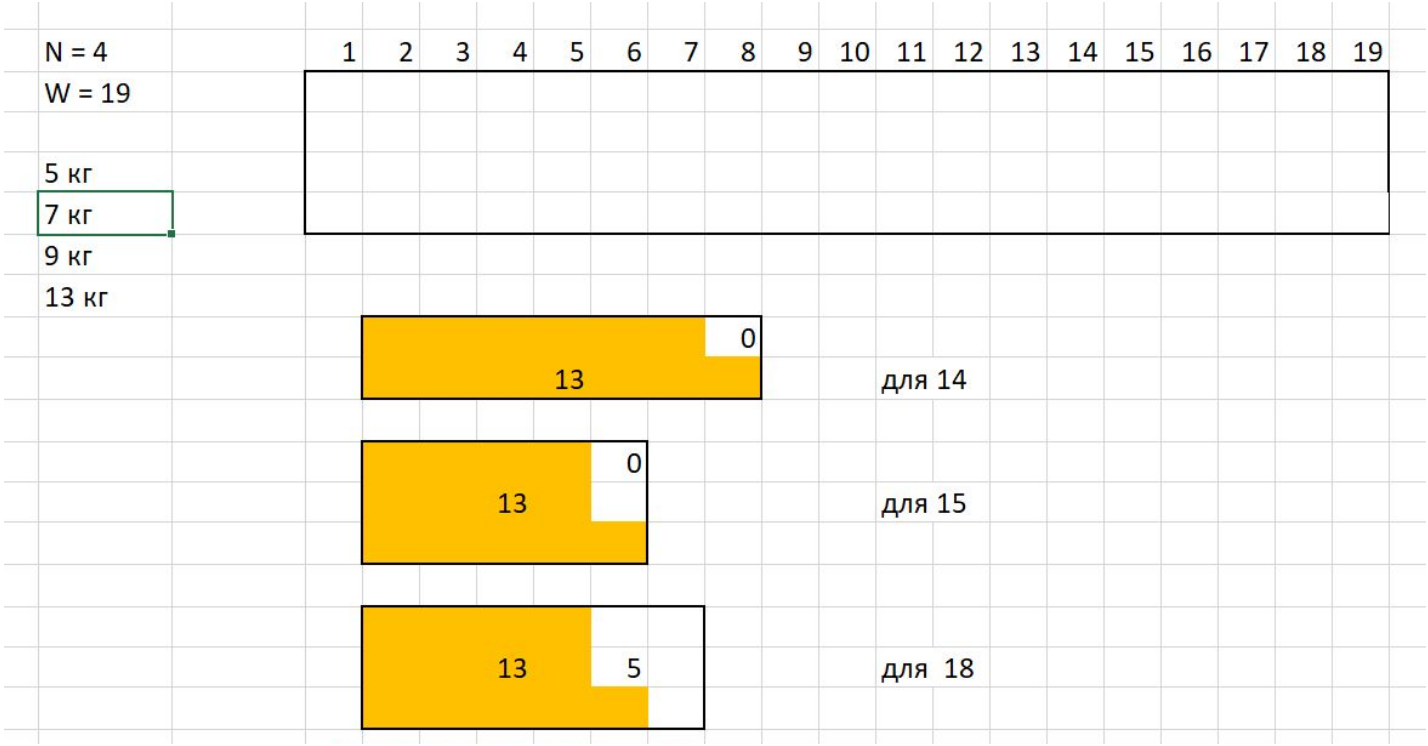
Дано N золотых слитков массой m_1, …, m_N. Ими наполняют рюкзак, который выдерживает вес не более M. Какую наибольшую массу золота можно унести в таком рюкзаке?

Решение

- Сформируем матрицу A, в которой номер строки – номер камня, номер столбца – набранный вес
- В нулевой столбец запишем нули
- Max и сортировки – в <algorithm>
- vector из vector-ов
- Кто найдет опечатку в таблице?

Камни. 1119

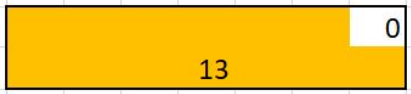
Дано N золотых слитков массой m_1, …, m_N. Ими наполняют рюкзак, который выдерживает вес не более M. Какую наибольшую массу золота можно унести в таком рюкзаке?



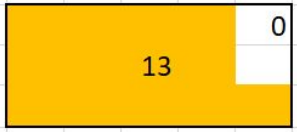
Камни. 1119

Дано N золотых слитков массой m_1, …, m_N. Ими наполняют рюкзак, который выдерживает вес не более M. Какую наибольшую массу золота можно унести в таком рюкзаке?

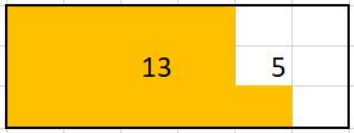
N = 4		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
W = 19	5 кг	0	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	7 кг	0	0	0	0	5	5	7	7	7	7	7	12	12	12	12	12	12	12	12
5 кг	9 кг	0	0	0	0	5	5	7	7	9	9	9	12	12	14	14	16	16	16	16
7 кг	13 кг	0	0	0	0	5	5	7	7	9	9	9	12	13	14					
9 кг																				
13 кг																				



для 14



для 15



для 18

Где опечатка?

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	*0*	0	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	*0*	0	0	0	0	5	5	*7*	7	7	7	7	12	12	12	12	12	12	12	12
3	0	0	0	0	0	5	5	*7*	7	9	9	9	12	12	14	14	16	16	16	16
4	0	0	0	0	0	5	5	*7*	7	9	9	11	12	12	14	14	16	16	18	*18*
5	0	0	0	0	0	5	5	7	7	9	9	11	12	13	14	14	16	16	18	*18*

- Аккуратно с выходом за границы массива
- Не забудьте: if ((j-p[i]) >=0) {сравниваем A[i,j], else {A[i,j] = из строки выше}}
- cout<<endl;

ДП бывает очень разное!

Это была двумерная динамика.

Теперь двумерная, но совсем другого рода.

Задача «о максимальном пути», она же - Черепашка

Путешествия развивают ум, если, конечно, он у вас есть.

Гилберт Честертон

Задача «Черепашка» № 2965

9	8	6	2
10	11	13	11
3	7	12	8
5	9	13	9

Дана прямоугольная таблица (*n* строк, *m* столбцов), в клетках которой записаны целые числа Черепашка находится в левой нижней клетке, и ей необходимо попасть в правую верхнюю клетку. За один ход Черепашка может переместиться в соседнюю верхнюю или правую клетку. Требуется найти путь Черепашки с максимальной суммой элементов.

Решение задачи «Черепашка». П.П.

- Полный перебор вариантов – универсальный способ решения. Но рассмотрим его потенциальные возможности

- Пусть дана таблица 4х4. Любой путь состоит из трёх перемещений вверх и трех перемещений вправо, т.е. длина пути равна шести. Другими словами, дано 6 шагов, из них 3 выбираются для перемещений вверх, оставшиеся 3 – для перемещений вправо определяются однозначно. Т. о. количество способов выбора трех перемещений из шести

  $C_6^3 = \frac{6!}{3! \cdot 3!} = 20$

  В общем случае C_{n+m-2}^{n-1}

- При нахождении суммы (стоимости) пути потребуется 5 операци сложения, всего 100 операций. Оценим время решения задачи для компьютера с миллионным быстродействием (см. презентация предыдущих занятий о сложности алгоритмов и быстродействии на примере задачи о тупоугольном треугольнике)

Длительность вычислений

Размер таблицы	Длина пути	Количество путей	Количество операций сложения	Приблизительное время решения задачи
4 × 4	6	20	100	0,0001 с
8 × 8	14	3432	44616	0,045 с
31 × 31	60	~10^{17}	~59×10^{17}	~200 000 лет

Итак, возможности полного перебора вариантов ограничены.

Решение задачи «Черепашка». Д.П.

9	8	6	2
10	11	13	11
3	7	12	8
5	9	13	9

а

27	40	58	65
18	32	52	63
8	21	39	47
5	14	27	36

б

Вычисление пути

27	**40**	**58**	**65**
18	**32**	**52**	**63**
8	**21**	**39**	**47**
5	**14**	**27**	**36**

После полного вычисления B мы находим стоимость пути Черепашки (для рассматриваемого примера она равна 65), но не сам путь (он выделен на рис. жирным шрифтом). Для нахождения пути Черепашки следует выполнить «обратный просмотр» массива B. Его суть: из значения $B[i,j]$ вычитаем $A[i,j]$ и смотрим, которое из двух чисел — $B[i-1,j]$ или $B[i,j-1]$ — равно полученному числу. Осуществляем переход по равенству и продолжаем до тех пор, пока не будет достигнут элемент $B[1,1]$.

Вычисление пути

Естественно, что следует предусмотреть ситуации наличия одной соседней клетки. Рекурсивный вариант реализации этой логики имеет следующий вид:

```
Procedure Way(i, j: LongInt);
  Begin
    If (i=1) And (j=1) Then Exit;
    If (i=1) And (j>1) Then Way(i,j-1)
      Else If (i>1) And (j=1) Then Way(i-1,j)
        Else
          If B[i,j]-A[i,j]=B[i-1,j]
            Then Way(i-1,j)
            Else Way(i,j-1);
    Write(i, ' ', j, '; ');
  End;
```

В рассмотренном варианте массив B формировался, начиная с элемента $B[1,1]$.

Временна́я сложность решения — $O(n \cdot m)$. Для вычисления каждого значения B требуется максимум две операции — сравнение и сложение. Для таблицы размером $n = 300$, $m = 300$ общее количество операций меньше 1 000 000, т. е. компьютер с миллионным быстродействием выполнит задачу менее чем за одну секунду.

С вычислением пути это задача № 619

А теперь одномерная динамика.

Но эта задача чем-то напоминает «Камни»

Копилка №625

- Она же – банкомат
- Разные виды монет, и их сколько угодно.
- Вес фиксированый
- Бывает недостижимый вес
- Python ловит TL на N>42. Это максимум 44 балла из 100

Если не откроется таблица

										w				
j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Pmin	0	-1	2	6	4	8	6	10	8	12	10	INT_MAX	INT_MAX	INT_MAX

	P	W
0	6	3
1	2	2
2	1	15

w[i] вес i-ой монеты

p[i] стоимость i-ой монеты

Pmin[w] = Pmin[j-w[kind]] +p[kind]

ДЗ

Количество конфет =
max(0; количество на 100 – количество на 0)

Списано = 0 обоим за задачу

Сдавать тут:
https://informatics.msk.ru/mod/statements/view.php?id=87032#1

Можно задавать мне вопросы по VK или телеграмм в процессе решения

1. Факториал №351
2. Числа Фибоначчи №201
3. Камни № 1119
4. Черепашка № 2965

Для опережающих

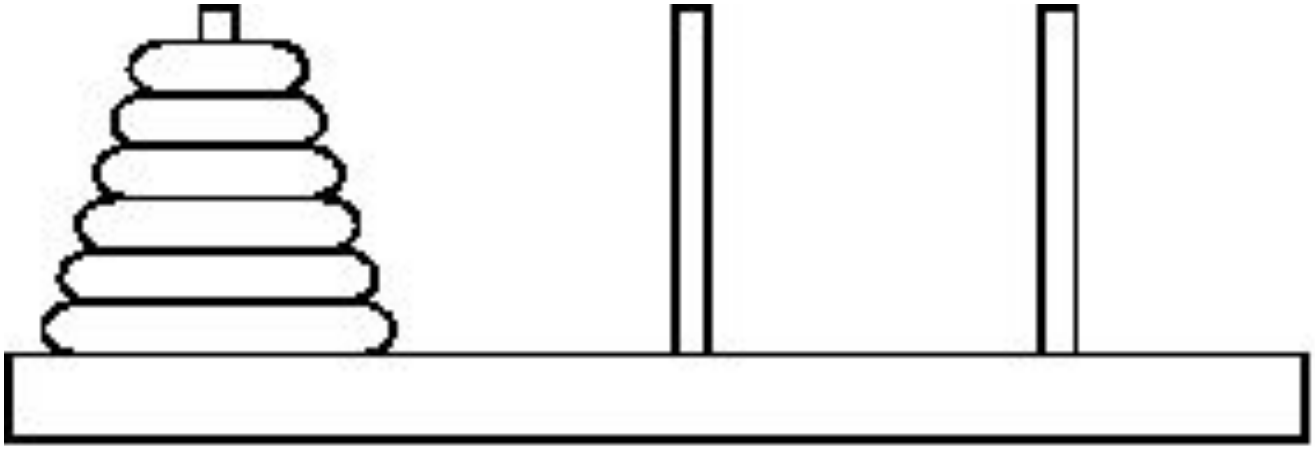
1. Выражение № 612
2. Спиралька № 1470
3. Черепашка (+путь) № 619
4. Копилка № 625

Вторая пара. Продолжение ДП и рекурсии

Григорьева Анастасия Викторовна
мат-мех + Академическая гимназия
СПбГУ

…еще раз про рекурсию

Ханойские башни № 3050



Void Hanoi(n, i, j, k)

Решение

```
#include <iostream>
using namespace std;
void hanoi(int number, int from, int to, int free)
{
        if (number != 0)
        {
                hanoi(number - 1, from, free, to);
                cout << number << ' ' << from << ' ' << to << '\n';
                hanoi(number - 1, free, to, from);
        }
}
int main()
{
        int n;
        cin >> n;
        hanoi(n, 1, 3, 2);
}
```

Интересные разбиения. Без номера

Недавно на кружке по математике Миша узнал про разбиения на слагаемые. Разбиением числа n на слагаемые называется представление его в виде суммы неубывающего набора натуральных чисел. Например, $9 = 1 + 2 + 2 + 4$ является разбиением числа 9 на слагаемые.

Миша называет разбиение интересным, если никакие два слагаемых в наборе не равны и не отличаются ровно на 1. Так, например, разбиение, приведенное выше не является интересным, а разбиение $9 = 1 + 3 + 5$ — является.

Помогите Мише вывести все интересные разбиения числа n на слагаемые.

Формат входных данных

На ввод подается одно целое число n $(1 \leq n \leq 80)$.

Формат выходных данных

Выведите все интересные разбиения числа n на слагаемые. Разбиения можно выводить в любом порядке. Соблюдайте формат из примера.

Пример

стандартный ввод	стандартный вывод
9	9=1+3+5 9=1+8 9=2+7 9=3+6 9=9

Решение

- Для генерации всех интересных разбиений применим рекурсию. В качестве параметров будем передавать уже сгенерированный фрагмент разбиения, последнее использованное число и сумму, которую осталось набрать.
- Когда разбиение полностью готово, выводим его.

Код «Интересные разбиения»

```
void doit(int ii){
    for (int i = ii; i <= n; i++){
        if (i + s == n){

            cout<<n << "=";
            for (int j = 0; j < sl.size(); j++)cout << sl[j] <<'+';
            cout << i;
            cout << endl;
            break;
        }
        else if (i + s < n){
            sl.push_back(i);
            s+=i;
            doit(i + 2);
            sl.pop_back();
            s-=i;
        }
        if (i + s > n) break;
    }
}
```

А теперь снова ДП

Возрастающая подпоследовательность №613

Даны N целых чисел $X_1, X_2, ..., X_N$. Требуется вычеркнуть из них минимальное количество чисел так, чтобы оставшиеся шли в порядке возрастания.

Ограничения: $1 \le N \le 10\,000$, $1 \le X_i \le 60\,000$, время 4 с.

Ввод из файла incseq.in. В первой строке находится число N. В следующей строке — N чисел через пробел.

Вывод в файл incseq.out. В первой строке выводится количество невычеркнутых чисел, во второй — сами невычеркнутые числа через пробел в исходном порядке. Если вариантов несколько, вывести любой.

Пример

Ввод

```
6
2 5 3 4 6 1
```

Вывод

```
4
2 3 4 6
```

1 5 3 7 1 4 10 15

Пример

Для каждого члена исходной последовательности нужно вычислить максимальную длину возрастающей подпоследовательности, оканчивающейся этим элементом. Для примера

2 5 3 4 6 1

эта характеристика будет выглядеть так:

1 2 2 3 4 1

Важно про НВПП

- Подпоследовательнось – это не обязательно числа, идущие подряд
- Считаем вместе:
- 1 4 5 6 7

  - База
  - Переход
  - Ответ

Код

```
n = int(input())
x = list(map(int, input().split()))
x.insert(0, 0)
r = [0]
q = [0]*(n+1)
ans = []
for i in range(1, n+1):
    a = x[i]
    for j in range(-1, -1-len(r), -1):
        if a > x[r[j]]:
            q[i] = r[j]
            if j == -1:
                r.append(i)
            else:
                if a < x[r[j+1]]:
                    r[j+1] = i
            break
c = r[-1]
while c != 0:
    ans.append(x[c])
    c = q[c]
print(len(ans))
ans.sort()
print(*ans)
```

Задача о рюкзаке

Рюкзак № 3089

Вектор из пар:
vector<pair<int, int>> K(n+1)

K[i].first
K[i].second

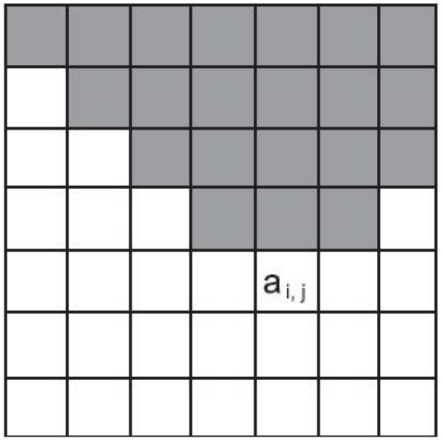
В остальном вспоминаем задачу Камни.

Рюкзак с восст.ответа № 3090

Задача «Таблица» №1150

Рассмотрим прямоугольную таблицу размером $n \times m$. Занумеруем строки таблицы числами от 1 до n, а столбцы — числами от 1 до m. Будем такую таблицу последовательно заполнять числами следующим образом.

Обозначим через a_{ij} число, стоящее на пересечении i-ой строки и j-ого столбца. Первая строка таблицы заполняется заданными числами — $a_{11}, a_{12}, \ldots, a_{1m}$. Затем заполняются строки с номерами от 2 до n. Число a_{ij} вычисляется как сумма всех чисел таблицы, находящихся в «треугольнике» над элементом a_{ij}. Все вычисления при этом выполняются по модулю r.



По модулю это значит остаток от деления.

23 mod 7 = 23%7 = 2

(a + b)%c = a%c + b%c

Пояснение

Например, если таблица состоит из трёх строк и четырёх столбцов, и первая строка состоит из чисел $2, 3, 4, 5$, а $r = 40$ то для этих исходных данных таблица будет выглядеть следующим образом (взятие по модулю показано только там, где оно приводит к изменению числа):

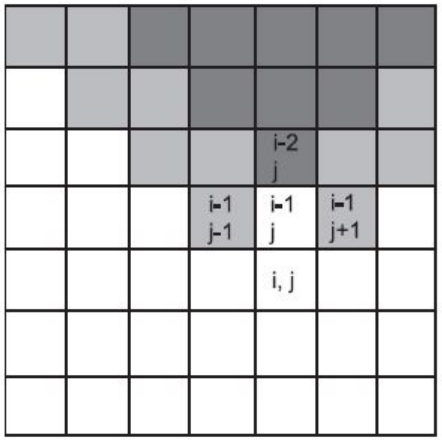
2	3	4	5
$5 = 2 + 3$	$9 = 2 + 3 + 4$	$12 = 3 + 4 + 5$	$9 = 4 + 5$
$23 = 2 + 3 + 4 + 5 + 9$	$0 = (2 + 3 + 4 + 5 + 5 + 9 + 12) \bmod 40 = 40 \bmod 40$	$4 = (2 + 3 + 4 + 5 + 9 + 12 + 9) \bmod 40 = 44 \bmod 40$	$33 = 3 + 4 + 5 + 12 + 9$

Требуется написать программу, которая по заданной первой строке таблицы (a_{11}, a_{12}, ..., a_{1m}), вычисляет последнюю строку, как описано выше.

Решение

База динамики (первая строка) нам явно задана. Рассмотрим два варианта вычисления значения в ячейке с использованием уже вычисленных данных.

В первом варианте нам достаточно завести массив B в котором мы будем накапливать ответы для ячеек. Очевидно, первая строка массива B совпадает с первой строкой массива A. Рассмотрим, как можно вычислить значение ячейки $B[i, j]$. Треугольник с вершиной в точке (i, j) состоит из треугольников с вершинами в точках $(i-1, j-1)$, $(i-1, j+1)$ а также из точек $(i-1, j)$ и (i, j). В то же время, треугольник с вершиной в точке $(i-2, j)$ перекрывается дважды, что приводит к двукратному суммированию значений в этом треугольнике. Избежать этого очень просто — достаточно один раз вычесть уже подсчитанную для этого треугольника сумму. Таким образом, в итоге получим следующую формулу $B[i][j] = B[i-1][j-1] + B[i-1][j+1] - B[i-2][j] + A[i-1][j] + A[i][j]$. Пользуясь этой формулой легко подсчитать массив, содержащий ответ.



Покупка билетов № 212

Выбираем минимум из трёх предыдущих

Сумма (без номера). Муниц.этап

Задача G. Сумма

Задано целое положительное число n. Требуется найти число способов представить его в виде суммы нечетных слагаемых. При этом разбиения, отличающиеся только порядком слагаемых, считаются одинаковыми.

Например, число 6 можно представить следующими способами: $1+1+1+1+1+1$, $1+1+1+3$, $3+3$, $1+5$.

Формат входных данных

На вход подается число n ($1 \le n \le 1000$).

Формат выходных данных

Выведите число способов представить n в виде суммы нечетных слагаемых.

Примеры

стандартный ввод	стандартный вывод
6	4

Разбор. Сумма. Видео: https://youtu.be/e9Y4iTUpSyA

Решение на 14 баллов. Можно было просто перебрать все возможные разбиения на нечётные слагаемые и посчитать их количество.

Правильное решение. Будем решать динамическим программированием и считать следующую величину: d[какая сумма получена][последнее слагаемое, которое мы брали]. Пересчёт следующий:

$$d[i][j] = \begin{cases} d[i][j-1], j - \text{чётное} \\ d[i][j-1] + d[i-j][j], j - \text{нечётное} \end{cases}.$$

Ответ будет храниться в $d[n][n]$.

Если всё считать в integer, то количество баллов равно 62.

Если всё считать в int64, то количество баллов равно 74.

Если всё считать с помощью длинной арифметики, то количество баллов равно 100.

Числа. ДП на подотрезках.

Дана последовательность чисел $a_1, a_2, ..., a_n$.

За одну операцию разрешается удалить любое (кроме крайних) число, заплатив за это штраф, равный произведению этого числа на сумму соседних. Требуется удалить все числа, кроме крайних, с минимальным суммарным штрафом.

Приведем пример оптимальной последовательности операций для начальной последовательности 1 50 51 50 1.

Удаляем четвертое число, штраф $50 \times (1 + 51) = 2600$, получаем 1 50 51 1.

Удаляем третье число, штраф $51 \times (50 + 1) = 2601$, получаем 1 50 1.

Удаляем второе число, штраф $50 \times (1 + 1) = 100$.

Итого, штраф 5301.

Требуется написать программу, которая по заданному целому числу n, и последовательности $a_1, a_2, \ldots, a_n$ найдет минимальный возможный суммарный штраф.

Формат входных данных

На ввод подается число n и затем n чисел $a_1, a_2, \ldots, a_n$ ($1 \le n \le 100$, $-100 \le a_i \le 100$).

Формат выходных данных

Вашей программе требуется вывести единственное число — минимальный возможный штраф.

Пример

стандартный ввод	стандартный вывод
5 1 50 51 50 1	5301

Решение ниже. Либо видео: https://youtu.be/pq4pA8PzP5w

- Заполняем сначала таблицу штрафов d бесконечно большим числом или недостижимым (для этой задачи хватит 3000000)
- В d[l][r] будем хранить минимально возможный штраф, получившийся, если мы верно решим задачу на промежутке от позиции l до позиции r. То есть минимальный, если удалять будем грамотно.
- Считаем для какого-то A[end], который будет последним внутри этого отрезка, который мы удалим.
- Тогда останется на предпоследнем шаге A[l], A[end], A[r]. А штраф к тому моменту накопится как сумма штрафов на промежутке (от l до end) и (от end до r).

Код. Числа.

```
cin >> n;

long long a[n];
long long d[n][n];

for (int i = 0; i < n; ++i){
    cin >> a[i];
}

for (int len = 2; len <= n; ++len){
    for (int l = 0; l <= n - len; ++l){
        int r = l + len - 1;
        if (len == 2) {
            d[l][r] = 0;
        } else {
            d[l][r] = 1000000000;
            for (int end = l + 1; end <= r - 1; ++end){
                d[l][r] = min(d[l][r], a[end] * (a[l] + a[r]) + d[l][end] + d[end][r]);
            }}}}
if (n == 1){
    d[0][n-1] = 0;
}
cout << d[0][n-1];
```

Эффективные алгоритмы