Яндекс

Сортировка списков в Perl и Python

Олег Алистратов
Руководитель группы разработки OTRS

Я.Субботник, Екатеринбург
6 июля 2013 года

Основы

Python

```
result = sorted(unsorted)

unsorted.sort()
```

Perl

```
my @sorted = sort @unsorted;
```

Универсальные алгоритмы

1. Сравнить два элемента
2. Переставить два элемента

Функция сравнения

- Принимает два элемента
- Возвращает целое число:
  - **< 0**, если первый элемент меньше второго
  - **0**, если элементы равны
  - **> 0**, если первый элемент больше второго

Списки, содержащие NaN

Если список содержит NaN, результат сортировки не определен.

```
>>> sorted([1, float('NaN'), 0])
[1, nan, 0]
```

Удаление NaN из списков

Python

```
import math
clean = filter(lambda x: not math.isnan(x), mixed)
```

Perl

```
@clean = grep { $_ == $_ } @mixed;
```

Orcish Maneuver (OM)

“OR” + “cache” = “Orcish” —
pronounced the same way a reader of Tolkien literature would say it.

Joseph Hall

Сортировка файлов по времени модификации

Python

```
def compare_mtime(a, b):
    return cmp(os.path.getmtime(a), os.path.getmtime(b))

sorted(files, cmp=compare_mtime)
```

Perl

```
sort { -M $a <=> -M $b } @files;
```

Всё, что происходит в функции сравнения, происходит $O(n \log n)$ раз.

Кеширование ключей

Python

```
def cmp_cached(a, b):
    if a not in cache:
        cache[a] = os.path.getmtime(a)
    if b not in cache:
        cache[b] = os.path.getmtime(b)
    return cmp(cache[a], cache[b])

sorted(files, cmp=cmp_cached)
```

Perl

```
my %cache;
my @sorted = sort {
    ( $cache{$a} //= -M $a )
              <=>
    ( $cache{$b} //= -M $b )
} @files;
```

Schwartzian transform (ST)

Decorate-Sort-Undecorate

Сохранение ключа с данными

1. `[ "abc", "de", "fghi", "k" ]`
2. `[ (3, "abc"), (2, "de"), (4, "fghi"), (1, "k") ]`
3. `[ (1, "k"), (2, "de"), (3, "abc"), (4, "fghi") ]`
4. `[ "k", "de", "abc", "fghi" ]`

ST в действии

Python

```
decorated = [ (calckey(x), x ) for x in data ]
decorated.sort()
result = [ x for k, x in decorated ]
```

Perl

```
my @s = map  { $_->[1] }
        sort { $a->[0] <=> $b->[0] }
        map  { [calckey(@$_), $_] }
             @list;
```

ST в Python

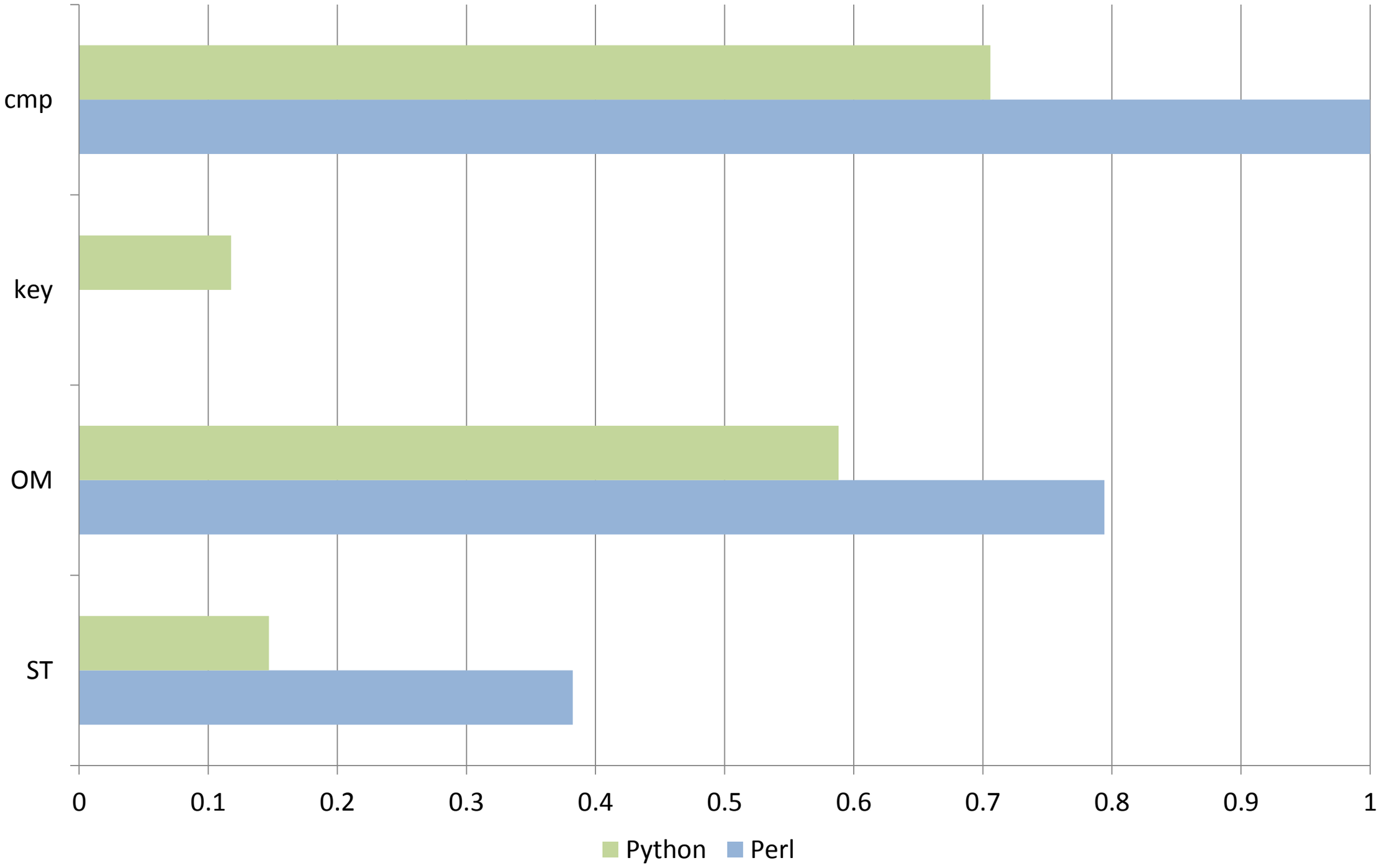
Функция sorted() принимает параметр key – и делает ST за вас.

```
sorted(os.listdir('/'), key=lambda x: os.path.getmtime(x))
```

Сортировка по полям

operator.itemgetter(), operator.attrgetter()

```
sorted(
    goods,
    key=operator.attrgetter('price'),
    reverse=True
)
```



Производительность пользовательской функции сравнения, OM и ST

Guttman-Rosler Transform (GRT)

Исходные данные

Кортежи из трех целых чисел в диапазоне [0, 99]

```
[
    ( 8, 85, 27),
    (41, 65, 81),
    (51, 66, 75),
    (17, 71, 84),
    (41, 13, 91),
    ...
]
```

Арифметическое кодирование

Python

```
def unpack(a):
    x = int(a / 100**2)
    y = int(a / 100) - x * 100
    z = 99 - a % 100
    return (x, y, z)
def pack(s):
    return s[0] * 100**2 + s[1] * 100 + 99 - s[2]

map(unpack, sorted(map(pack, data)))
```

Perl

```
my @s = map
        {
            my $x = int($_ / 100**2);
            my $y = int($_ / 100) - $x * 100;
            my $z = 99 - $_ % 100;
            [ $x, $y, $z ];
        }
        sort { $a <=> $b }
        map  { $_->[0] * 100**2 + $_->[1] * 100 + 99 - $_->[2] }
        @data;
```

Кодирование в строки

Python

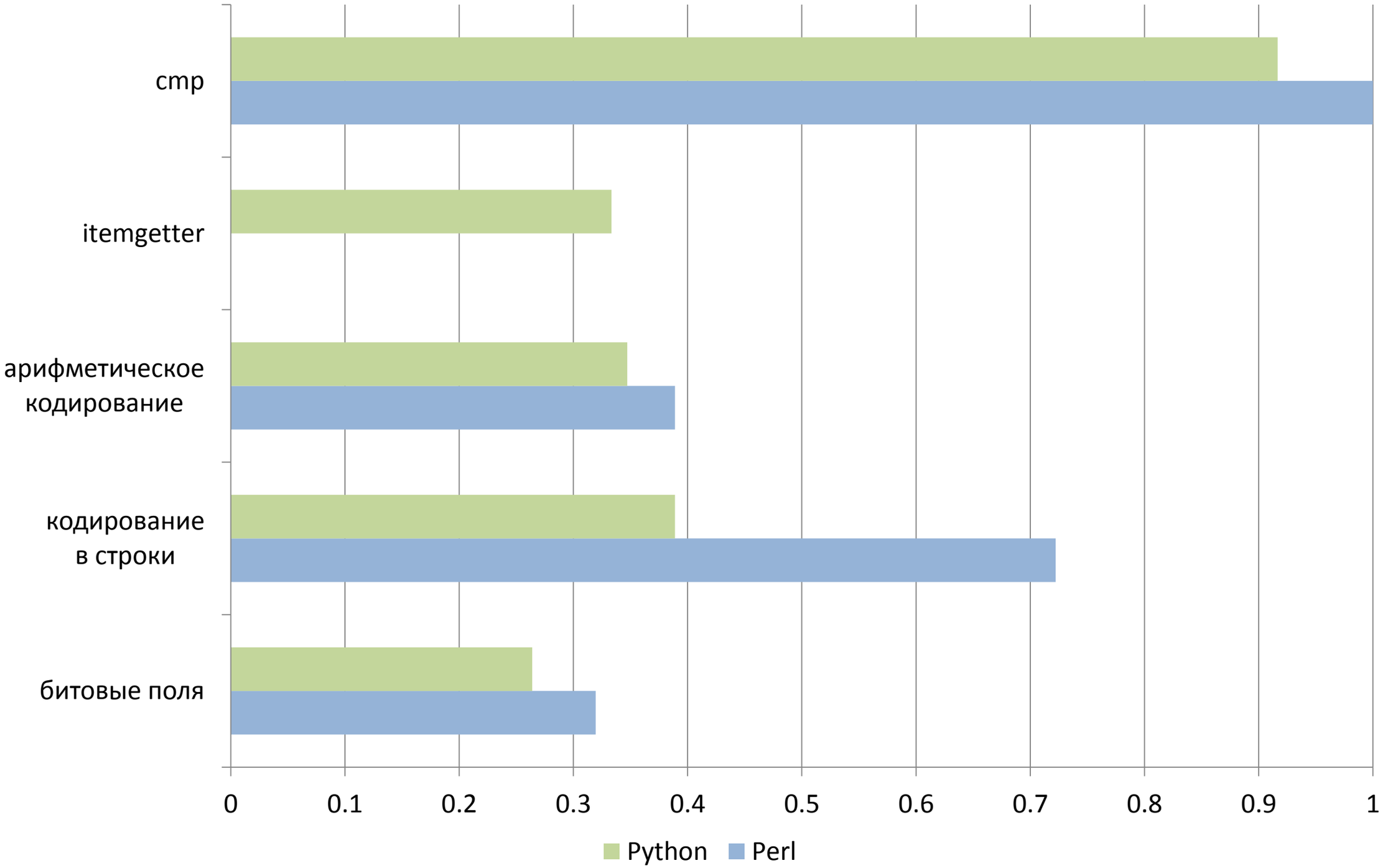
```
def unpack(a):
    return (ord(a[0]), ord(a[1]), 99 - ord(a[2]))

def pack(s):
    return chr(s[0]) + chr(s[1]) + chr(99 - s[2])

map(unpack, sorted(map(pack, data)))
```

Perl

```
my @s = map  { [ unpack "C3", $_ ] }
        sort
        map  { pack "C3", @$_ }
        @data;
```



Производительность пользовательской функции сравнения и GRT

Сериализация

- Запись в битовые поля
- Арифметическое кодирование
- Формирование строк фиксированной длины (*padding*)
- Формирование строк с разделителями (например, нулевыми байтами)

Модули и библиотеки

CPAN

- Sort::Maker
- Sort::MultipleFields
- Sort::Key
- Sort::Fields
- Sort::XS
- Sort::External

use sort

Прагма sort позволяет узнавать и выбирать алгоритм сортировки.

```
use sort qw(stable);
```

numpy.sort, numpy.lexsort

- Выбор алгоритма сортировки
- Сортировка многомерных массивов
- Сортировка по нескольким полям
- Поиск в упорядоченных списках

Заключение

- Функция сравнения вызывается $O(n \log n)$ раз
- Сохраняйте ключи сортировки
- Упакуйте элементы списка в числа или строки и сравнивайте их одной операцией
- Пользуйтесь готовым

Исходные тексты

https://github.com/alistratov/articles/tree/master/sorting/benchmarks

Олег Алистратов

Руководитель группы
разработки OTRS

wmute@yandex-team.ru

Спасибо