Игорь Бурдонов, <igor@ispras.ru>

Андрей Максимов, <andrew@ispras.ru>

Институт системного программирования РАН им. В.П. Иванникова,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

**ДВАДЦАТЬ ФУНКЦИЙ ПОДОБИЯ ДВУХ КОНЕЧНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ**

**Аннотация**. В статье рассматриваются различные числовые функции, определяющие степень «похожести» двух заданных конечных последовательностей. Эти меры подобия основаны на определяемом нами понятии вложения в последовательность. Частным случаем такого вложения является обычная подпоследовательность (subsequence). Другие случаи дополнительно требуют равенства расстояний между соседними символами подпоследовательности в обеих последовательностях. Это является обобщением понятия отрезка последовательности (substring), в котором эти расстояния единичны. Дополнительно может требоваться равенство расстояний от начала последовательностей до первого символа вложения или от последнего символа вложения до конца последовательностей. Кроме этих двух последних случаев, вложение может входить в последовательность несколько раз. В литературе используются такие функции как число общих вложений или числа пар вхождений вложений в последовательности. Кроме них, мы вводим ещё три функции: сумма длин общих вложений, сумма минимумов числа вхождений общего вложения в обе последовательности и функция подобия на основе наибольшего по числу символов общего вложения. Всего рассматриваются 20 числовых функций, для 17 из которых предложены алгоритмы (в том числе новые) полиномиальной сложности, ещё для двух функций алгоритмы имеют экспоненциальную сложность с уменьшенным показателем степени. В Заключении даётся краткая сравнительная характеристика этих вложений и функций.

**Ключевые слова:** анализ последовательностей, общие подпоследовательности, наибольшие и максимальные общие подпоследовательности, каноническое вложение, соответствующие совместные вложения, комбинаторные алгоритмы для подпоследовательностей и вложений, аксиомы подобия.

**Keywords:** sequence analysis, common subsequences, longest and maximal common subsequences, canonical embedding, matching joint embeddings, algorithms for subsequence and embeddings combinatorics, similarity axioms.

1. **Введение**

Анализ последовательностей широко используется в социальных, управленческих, политических, демографических, психологических науках, в химии, биоинформатике и при обработке текстов. Последние десятилетия появилось много работ, посвящённых этой тематике [1-6]. Применяются различные метрики и меры сходства (подобия) последовательностей [4, 5].

В этой статье мы рассматриваем различные числовые функции, определяющие степень «похожести» двух заданных конечных последовательностей. Эти меры подобия основаны на определяемом нами понятии вложения в последовательность. Частным случаем такого вложения является подпоследовательность. Другие случаи учитывают дополнительно расстояния между символами подпоследовательности в обеих последовательностях. Например, последовательности «МЕТРИКА» и «МАРОККО» имеют наибольшую общую подпоследовательность «МРК», с учётом расстояний между символами «Р-К», с учётом расстояний между символами и от последнего символа вложения до конца последовательности «К-», с учётом расстояний между символами и от начала последовательности до первого символа вложения «М----К».

Понятие вложения вводится с использованием пустого символа, не принадлежащего алфавиту заданных последовательностей. Всего вводятся пять типов вложения: *E*‑вложение получается заменой некоторых символов пустым символом, *L*‑вложение получается из *E*‑вложения удалением пустого префикса, *R*‑вложение получается из *E*‑вложения удалением пустого постфикса, *O*‑вложение получается из *E*‑вложения удалением пустых префикса и постфикса, наконец, *A*‑вложение получается из *E*‑вложения удалением всех пустых символов, что совпадает с понятием подпоследовательности. Мнемоника обозначения вложений образована от английских слов: *E* — **E**mpty symbol (пустой символ), далее указание места, где пустых символов нет (there are no empty symbols): *L* — on the **L**eft (слева), *R* — on the **R**ight (справа), *O* — **O**utside (снаружи, т.е. слева и справа), *A* — **A**nywhere (в любом месте, т.е. нигде нет пустых символов).

Каждое вложение может иметь несколько вхождений в последовательность, т.е. несколько *E*‑вложений, из которых данное вложение получается удалением префикса, постфикса, префикса и постфикса или всех пустых символов. Например, подпоследовательность «МРК» входит один раз в последовательность «МЕТРИКА» (соответствующее *E*‑вложение «М--Р-К-») и два раза в последовательность «МАРОККО» (соответствующие *E*‑вложения «М-Р-К--» и «М-Р--К-». Для вложений, которые могут содержать пустые символы, определяется понятие μ‑длины как число непустых символов (для *A*‑вложения совпадает с длиной подпоследовательности).

Для каждого из четырёх типов вложения (*L*, *R*, *O* и *A*) определяются пять функций: 0) число общих вложений, 1) сумма μ‑длин общих вложений, 2) сумма минимумов чисел вхождения общих вложений в заданные последовательности, 3) сумма произведений чисел вхождения общих вложений в заданные последовательности, а также 4) мера похожести, основанная на наибольшей (по длине) общей подпоследовательности (*longest common subsequence*, *lcs*).

Некоторые из этих двадцати функций хорошо известны, например, число общих подпоследовательностей (число общих *A*‑вложений) или сумма произведений чисел вхождения общих подпоследовательностей в заданные последовательности (сумма произведений чисел *E*‑вложений общих *A*‑вложений в заданные последовательности) [3]. Другие функции, особенно учитывающие «расстояния» между символами, мы вводим в данной статье.

После настоящего Введения в разделе 2 вводятся основные понятия и обозначения. В разделе 3 рассматривается оптимизация общая для всех типов вложений: замена пустым символом тех символов, которые входят только в одну из двух последовательностей. Далее в разделах 4-7 рассматриваются четыре типа вложений *L*, *R*, *O* и *A*, и для каждого типа вложения — алгоритмы вычисления пяти функций 0, 1, 2, 3, 4. В Заключении подводятся итоги и намечаются направления дальнейших исследований.

1. **Определения и обозначения**

Для целых чисел *i* и *j* будем обозначать: *i*..*j* = { *i*, *i* + 1, ..., *j* }, если *i* ≤ *j*, *i*..*j* = ∅, если *i* > *j*.

Конечная последовательность в алфавите *H* длиной *m* ≥ 0 — это инъекция множества 1..*m* в множество *H*: 1..*m* → *H*. Множество конечных последовательностей в алфавите *H* обозначим *H*\*. Пустую последовательность (длины 0, пустая инъекция) обозначим (). Для непустой конечной последовательности *x i*‑й элемент последовательности, *i* ∈ 1..|*x*|, обозначим *xi* = *x*(*i*). Отрезок *xi*, *xi*+1, ..., *xj* для 1 ≤ *i* ≤ *j* ≤ |*x*| обозначим *x*[*i*..*j*]. Для *i* > *j* определим *x*[*i*..*j*] = (). Префикс обозначим как *x*[*j*] = *x*[1..*j*], пустой префикс (длины 0) определён, в том числе, и для пустой последовательности ()[0] = (). Конечная последовательность из *k* ≥ 0 повторений символа *h* ∈ *H* будем обозначать *hk*: |*hk*| = *k* & ∀ *i* = 1..|*k*| *hki* = *h*. Также вместо *h*1 будем писать просто *h*.

Конкатенация *x*⋅*y* конечных последовательностей *x* и *y* определяется условиями: |*x*⋅*y*| = |*x*| + |*y*|, ∀ *i* ∈ 1..|*x*| (*x*⋅*y*)*i* = *xi* и ∀ *j* ∈ 1..|*y*| (*x*⋅*y*)|*x*| +*j* = *yj*. Нам также понадобится конкатенация пары (*X*, *Y*) конечных множеств конечных последовательностей с парой (*z*, *t*) конечных последовательностей, которую определим так: (*X*, *Y*)⋅(*z*, *t*) = { (*x*⋅*z*, *y*⋅*t*) : (*x*, *y*) ∈ (*X*, *Y*) }. Будем считать, что в выражениях операция конкатенации приоритетнее операций над множествами (объединение, пересечение и разность).

Композицию функций *f* и *g* будем обозначать *fg*.

Введём *пустой символ* ε ∉ *H* и обозначим *H*ε*= H* ∪ {ε}.

Обозначим множество конечных последовательностей в алфавите *H*ε:

* не начинающихся пустым символом *L*(*H*) = { *v* ∈ *H*ε\* : |*v*| > 0 ⇒ *v*1 ≠ ε };
* не заканчивающихся пустым символом *R*(*H*) = { *v* ∈ *H*ε\* : |*v*| > 0 ⇒ *v*|*v*| ≠ ε };
* не начинающихся и не заканчивающихся пустым символом

*O*(*H*) = { *v* ∈ *H*ε\* : |*v*| > 0 ⇒ *v*1 ≠ ε & *v*|*v*| ≠ ε } = *L*(*H*) ∩ *R*(*H*).

Введём операции удаления пустых символов из конечной последовательности в алфавите *H*ε:

* *удаление префикса пустых символов* λ: *H*ε\* → *L*(*H*) определяется условием:

∀ *v* ∈ *H*ε\* λ(ε⋅*v*) = λ(*v*) & ∀ *u* ∈ *L*(*H*) λ(*u*) = *u*;

* *удаление постфикса пустых символов* ρ: *H*ε\* → *R*(*H*) определяется условием:

∀ *v* ∈ *H*ε\* ρ(*v*⋅ε) = ρ(*v*) & ∀ *u* ∈ *R*(*H*) ρ(*u*) = *u*.

* *удаление всех пустых символов* μ: *H*ε\* → *H*\* определяется условием:

∀ *v*, *w* ∈ *H*ε\* μ(*v*⋅ε⋅*w*) = μ(*v*⋅*w*) & ∀ *u* ∈ *H*\* μ(*u*) = *u*.

Определим для последовательности *x* ∈ *H*ε\*:

* *E‑вложение* получается из *x* заменой некоторых символов пустыми символами;
* *L‑вложение* получается из *E‑*вложения удалением префикса пустых символов;
* *R‑вложение* получается из *E‑*вложения удалением постфикса пустых символов;
* *O‑вложение* получается из *E‑*вложения удалением префикса и постфикса пустых символов, или из *L‑*вложения удалением постфикса пустых символов, или из *R‑*вложения удалением префикса пустых символов;
* *A‑вложение* получается из *E‑*, *L‑*, *R‑* или *O‑*вложения удалением пустых символов.

Для *x* ∈ *H*\* понятие *A‑*вложения совпадает с понятием подпоследовательности, а *E‑*вложение *v* соответствует понятию вхождения подпоследовательности μ(*v*) в *x*.

Обозначим множества вложений в последовательность *x* ∈ *H*\*:

* множество *E‑*вложений в *x*:

*E*(*x*) = { *u* ∈ *H*ε\* : |*u*| = |*x*| & ∀ *i* ∈ 1..|*u*| *ui* = *xi* ∨ *ui* = ε };

* множество *L‑*вложений в *x*:

*L*(*x*) = λ*E*(*x*);

* множество *R‑*вложений в *x*:

*R*(*x*) = ρ*E*(*x*);

* множество *O‑*вложений в *x*:

*O*(*x*) = λρ*E*(*x*) = λ*R*(*x*) = ρ*L*(*x*);

* множество *A‑*вложений в *x*:

*A*(*x*) = μ*E*(*x*) = μ*L*(*x*) = μ*R*(*x*) = μ*O*(*x*).

Для последовательности *x* в алфавите *H*ε\* введём понятие μ‑*длины* *x* как число непустых символов в *x*, очевидно, равное |μ(*x*)|. Для *x* ∈ *H*\* μ‑длина совпадает с длиной последовательности.

Обозначим для последовательностей *x* ∈ *H*\*:

* множество *E‑*вложений в *x* *L‑*вложения *u* ∈ *L*(*x*): *l*(*u*, *x*) = { *v* ∈ *E*(*x*) : λ(*v*) = *u* };
* множество *E‑*вложений в *x* *R‑*вложения *u* ∈ *R*(*x*): *r*(*u*, *x*) = { *v* ∈ *E*(*x*) : ρ(*v*) = *u* };
* множество *E‑*вложений в *x* *O‑*вложения *u* ∈ *O*(*x*): *o*(*u*, *x*) = { *v* ∈ *E*(*x*) : λρ(*v*) = *u* };
* множество *E‑*вложений в *x* *A‑*вложения *u* ∈ *A*(*x*): *a*(*u*, *x*) = { *v* ∈ *E*(*x*) : μ(*v*) = *u* };

Обозначим для последовательностей *x* ∈ *H*\* и *y* ∈ *H*\* множества пар *E‑*вложений:

* множество пар *E‑*вложений общих *L‑*вложений

*L*(*x*, *y*) = ∪ { *l*(*u*, *x*) × *l*(*u*, *y*) : *u* ∈ *L*(*x*) ∩ *L*(*y*) };

* множество пар *E‑*вложений общих *R‑*вложений

*R*(*x*, *y*) = ∪ { *r*(*u*, *x*) × *r*(*u*, *y*) : *u* ∈ *R*(*x*) ∩ *R*(*y*) };

* множество пар *E‑*вложений общих *O‑*вложений

*O*(*x*, *y*) = ∪ { *o*(*u*, *x*) × *o*(*u*, *y*) : *u* ∈ *O*(*x*) ∩ *O*(*y*) };

* множество пар *E‑*вложений общих *A‑*вложений

*A*(*x*, *y*) = ∪ { *a*(*u*, *x*) × *a*(*u*, *y*) : *u* ∈ *A*(*x*) ∩ *A*(*y*) }.

Обозначим для последовательностей *x* ∈ *H*\* и *y* ∈ *H*\* наибольшую μ‑длину общего вложения:

* для *L‑*вложений: *lcL*(*x*, *y*) = *max*{ μ(*u*) : *u* ∈ *L*(*x*) ∩ *L*(*y*) };
* для *R‑*вложений: *lcR*(*x*, *y*) = *max*{ μ(*u*) : *u* ∈ *R*(*x*) ∩ *R*(*y*) };
* для *O‑*вложений: *lcO*(*x*, *y*) = *max*{ μ(*u*) : *u* ∈ *O*(*x*) ∩ *O*(*y*) };
* для *A‑*вложений: *lcA*(*x*, *y*) = *max*{ |*u*| : *u* ∈ *A*(*x*) ∩ *A*(*y*) };

Наибольшую μ‑длину общего вложения естественно рассматривать как ещё одну функцию «похожести» последовательностей *x* и *y*: *lcI*(*x*, *y*) для *I* ∈ { *L*, *R*, *O*, *A* }. По этому критерию последовательность *x* больше всего «похожа» на саму себя, поскольку является наибольшим *I*‑вложением в себя: *lcI*(*x*, *x*) = *max*{ μ(*u*) : *u* ∈ *I*(*x*) } = |*x*| ≥ *max*{ μ(*u*) : *u* ∈ *I*(*x*) ∩ *I*(*y*) } = *lcI*(*x*, *y*).

Нас будут интересовать функции от последовательностей *x* ∈ *H*\* и *y* ∈ *H*\*, задаваемые таб. 1.

Таб. 1. Функции для общих вложений двух последовательностей *x* и *y*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| функциявложение | числообщихвложений | суммаμ‑длинобщихвложений | суммаминимумовчисел*E‑*вложенийобщих вложений | суммапроизведенийчисел*E‑*вложенийобщих вложений | наибольшаяμ‑длинаобщеговложения |
| 0 | 1 | 2 | 3 | 4 |
| *L* | *L*0(*x*, *y*) =|*L*(*x*) ∩ *L*(*y*)| | *L*1(*x*, *y*) = Σ { |μ(*u*)|: *u* ∈ *L*(*x*) ∩ *L*(*y*) } | *L*2(*x*, *y*) =Σ { *min*{ |*l*(*u*, *x*)|, |*l*(*u*, *y*)| }: *u* ∈ *L*(*x*) ∩ *L*(*y*) } | *L*3(*x*, *y*) = |*L*(*x*, *y*)| | *L*4(*x*, *y*) =*lcL*(*x*, *y*) |
| *R* | *R*0(*x*, *y*) =|*R*(*x*) ∩ *R*(*y*)| | *R*1(*x*, *y*) = Σ { |μ(*u*)|: *u* ∈ *R*(*x*) ∩ *R*(*y*) } | *R*2(*x*, *y*) =Σ { *min*{ |*r*(*u*, *x*)|, |*r*(*u*, *y*)| }: *u* ∈ *R*(*x*) ∩ *R*(*y*) } | *R*3(*x*, *y*) = |*R*(*x*, *y*)| | *R*4(*x*, *y*) =*lcR*(*x*, *y*) |
| *O* | *O*0(*x*, *y*) =|*O*(*x*) ∩ *O*(*y*)| | *O*1(*x*, *y*) = Σ { |μ(*u*)|: *u* ∈ *O*(*x*) ∩ *O*(*y*) } | *O*2(*x*, *y*) =Σ { *min*{ |*o*(*u*, *x*)|, |*o*(*u*, *y*)| }: *u* ∈ *O*(*x*) ∩ *O*(*y*) } | *O*3(*x*, *y*) = |*O*(*x*, *y*)| | *O*4(*x*, *y*) =*lcO*(*x*, *y*) |
| *A* | *A*0(*x*, *y*) =|*A*(*x*) ∩ *A*(*y*)| | *A*1(*x*, *y*) = Σ { |*u*|: *u* ∈ *A*(*x*) ∩ *A*(*y*) } | *A*2(*x*, *y*) =Σ { *min*{ |*a*(*u*, *x*)|, |*a*(*u*, *y*)| }: *u* ∈ *A*(*x*) ∩ *A*(*y*) } | *A*3(*x*, *y*) = |*A*(*x*, *y*)| | *A*4(*x*, *y*) =*lcA*(*x*, *y*) |

Заметим, что для *I* ∈ { *L*, *R*, *O*, *A* } и *j* ∈ 0..4 *Ij*(*x*, *y*) = *Ij*(*y*, *x*). Если *j* ≠ 1, то *Ij*(*x*, ()) = *Ij*((), *y*) = 1; *I*1(*x*, ()) = *I*1((), *y*) = 0.

Обозначим для непустой последовательности *x* ∈ *H*\*:

* самое левое *E‑*вложение в *x* *L‑*вложения *u* ∈ *L*(*x*): *ll*(*u*, *x*) = *v*, если *v* ∈ *l*(*u*, *x*) и ∀ *w* ∈ *l*(*u*, *x*) \ {*v*} *w*( *min* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε;
* самое правое *E‑*вложение в *x* *L‑*вложения *u* ∈ *O*(*x*): *lr*(*u*, *x*) = *v*, если *v* ∈ *l*(*u*, *x*) и ∀ *w* ∈ *l*(*u*, *x*) \ {*v*} *w*( *max* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε.
* самое левое *E‑*вложение в *x* *R‑*вложения *u* ∈ *R*(*x*): *rl*(*u*, *x*) = *v*, если *v* ∈ *r*(*u*, *x*) и ∀ *w* ∈ *r*(*u*, *x*) \ {*v*} *w*( *min* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε;
* самое правое *E‑*вложение в *x* *R‑*вложения *u* ∈ *O*(*x*): *rr*(*u*, *x*) = *v*, если *v* ∈ *r*(*u*, *x*) и ∀ *w* ∈ *r*(*u*, *x*) \ {*v*} *w*( *max* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε.
* самое левое *E‑*вложение в *x* *O‑*вложения *u* ∈ *O*(*x*): *ol*(*u*, *x*) = *v*, если *v* ∈ *o*(*u*, *x*) и ∀ *w* ∈ *o*(*u*, *x*) \ {*v*} *w*( *min* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε;
* самое правое *E‑*вложение в *x* *O‑*вложения *u* ∈ *O*(*x*): *or*(*u*, *x*) = *v*, если *v* ∈ *o*(*u*, *x*) и ∀ *w* ∈ *o*(*u*, *x*) \ {*v*} *w*( *max* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε.
* самое левое *E‑*вложение в *x* *A‑*вложения *u* ∈ *A*(*x*): *al*(*u*, *x*) = *v*, если *v* ∈ *a*(*u*, *x*) и ∀ *w* ∈ *a*(*u*, *x*) \ {*v*} *w*( *min* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε;
* самое правое *E‑*вложение в *x* *A‑*вложения *u* ∈ *A*(*x*): *ar*(*u*, *x*) = *v*, если *v* ∈ *a*(*u*, *x*) и ∀ *w* ∈ *a*(*u*, *x*) \ {*v*} *w*( *max* { *i* ∈ 1..|*x*| : *wi* ≠ *vi* } ) = ε.

В литературе самые левые (*left-most*) *E‑*вложения общих вложений называют также (для подпоследовательностей) каноническими (*canonical*) [3].

Обозначим для непустых последовательностей *x* ∈ *H*\* и *y* ∈ *H*\* множества пар *E‑*вложений:

* множество пар самых левых *E‑*вложений общих *L‑*вложений

*Ll*(*x*, *y*) = { (*ll*(*u*, *x*), *ll*(*u*, *y*)) : *u* ∈ *L*(*x*) ∩ *L*(*y*) };

* множество пар самых правых *E‑*вложений общих *L‑*вложений

*Lr*(*x*, *y*) = { (*lr*(*u*, *x*), *lr*(*u*, *y*)) : *u* ∈ *L*(*x*) ∩ *L*(*y*) };

* множество пар самых левых *E‑*вложений общих *R‑*вложений

*Rl*(*x*, *y*) = { (*rl*(*u*, *x*), *rl*(*u*, *y*)) : *u* ∈ *R*(*x*) ∩ *R*(*y*) };

* множество пар самых правых *E‑*вложений общих *R‑*вложений

*Rr*(*x*, *y*) = { (*rr*(*u*, *x*), *rr*(*u*, *y*)) : *u* ∈ *R*(*x*) ∩ *R*(*y*) };

* множество пар самых левых *E‑*вложений общих *O‑*вложений

*Ol*(*x*, *y*) = { (*ol*(*u*, *x*), *ol*(*u*, *y*)) : *u* ∈ *O*(*x*) ∩ *O*(*y*) };

* множество пар самых правых *E‑*вложений общих *O‑*вложений

*Or*(*x*, *y*) = { (*or*(*u*, *x*), *or*(*u*, *y*)) : *u* ∈ *O*(*x*) ∩ *O*(*y*) };

* множество пар самых левых *E‑*вложений общих *A‑*вложений

*Al*(*x*, *y*) = { (*al*(*u*, *x*), *al*(*u*, *y*)) : *u* ∈ *A*(*x*) ∩ *A*(*y*) };

* множество пар самых правых *E‑*вложений общих *A‑*вложений

*Ar*(*x*, *y*) = { (*ar*(*u*, *x*), *ar*(*u*, *y*)) : *u* ∈ *A*(*x*) ∩ *A*(*y*) };

Заметим, что |*Ol*(*x*, *y*)| = |*Or*(*x*, *y*)| = |*O*(*x*) ∩ *O*(*y*)| и |*Al*(*x*, *y*)| = |*Ar*(*x*, *y*)| = |*A*(*x*) ∩ *A*(*y*)|.

Далее через *x* и *y* будем обозначать две непустые последовательности в алфавите *H* с длинами *m* = |*x*|, *n* = |*y*|. Будем считать, что *m* ≤ *n*.

1. **Замена необщих символов пустым символом**

Общей оптимизацией для вычисления всех функций для всех вложений является замена необщих символов пустым символом: для *i* ∈ 1..*m* определим *x*^*i* = *xi*, если *xi* ∈ ***Im*** *y*, *x*^*i* = ε*i*, если *xi* ∉ ***Im*** *y*. Аналогично определяется *y*^. Эта замена выполняется за время ***O***(*mn*). В случае *A*‑вложений вместо замены необщего символа пустым символом можно просто удалять необщий символ. Описанные ниже алгоритмы можно применять после выполнения этой замены (удаления для *A*‑вложений).

1. ***A‑*вложения (подпоследовательности)**
	1. **Число общих *A‑*вложений**

Здесь мы докажем теорему, аналогичную лемме 6 в [3], но дадим своё доказательство, поскольку мы используем другие определения и обозначения.

Для непустой последовательности *z* ∈ *H*\* и символа *h* ∈ *H* обозначим максимальный индекс, по которому в последовательности *z* находится символ *h*, или 0, если *h* не входит в *z*:

*p*(*z*, *h*) = *max*{ *i* ∈ 1..|*z*|: *zi* = *h* }, если *h* ∈ ***Im*** *z*; *p*(*z*, *h*) = 0, если *h* ∉ ***Im*** *z*.

Обозначим: *k* = *p*(*x*[*m*‑1], *xm* }, *l* = *p*(*y*, *xm* }.

1.

*A*0(*x*, *y*) = *A*0(*x*[*m*‑1], *y*), если *xm* ∉ ***Im*** *y*;

*A*0(*x*, *y*) = *A*0(*x*[*m*‑1], *y*) + *A*0(*x*[*m*‑1], *y*[*l*‑1]), если *xm* ∈ ***Im*** *y* и *xm* ∉ ***Im*** *x*[*m*‑1];

*A*0(*x*, *y*) = *A*0(*x*[*m*‑1], *y*) + *A*0(*x*[*m*‑1], *y*[*l*‑1]) ‑ *A*0(*x*[*k*‑1], *y*[*l*‑1]), если *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1].

Доказательство.

Рассматриваем множества пар ( *er*(*u*, *x*), *er*(*u*, *y*) ) самых правых *E‑*вложений общих *A‑*вложений *u* последовательностей *x* и *y*. Обозначим пары последовательностей длины *m* и *n*:

*E* = *Ar*(*x*, *y*);

*Em* = *Ar*(*x*[*m*‑1], *y*)⋅(ε, ());

*Eml* = *Ar*(*x*[*m*‑1], *y*[*l*‑1])⋅(*xm*, *xm*⋅ε*n*‑*l*), если *xm* ∈ ***Im*** *y*, иначе *Eml* = ∅;

*Ekl* = *Ar*(*x*[*k*‑1], *y*[*l*‑1])⋅(*xm*⋅ε*m*‑*k*‑1⋅*xm*, *xm*⋅ε*n*‑*l*), если *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1], иначе *Ekl* = ∅.

Поскольку для любых *x`*, *y`* имеет место *A*0(*x`*, *y`*) = |*A*(*x`*) ∩ *A*(*y`*)| = |*Ar*(*x`*, *y`*)|, утверждение теоремы можно переписать в виде:

|*E*| = |*Em*|, если *xm* ∉ ***Im*** *y*;

|*E*| = |*Em*| + |*Eml*|, если *xm* ∈ ***Im*** *y* и *xm* ∉ ***Im*** *x*[*m*‑1];

|*E*| = |*Em*| + |*Eml*| ‑ |*Ekl*|, если *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1].

Поскольку *E‑*вложения из множества *Em* имеют вид (...ε, ...), а множества *Eml* и *Ekl* либо пусты, либо *E‑*вложения из них имеют вид (...*xm*, ...), имеем *Em* ∩ *Eml* = *Em* ∩ *Ekl* = ∅. Поскольку *m*‑1 ≥ *k*‑1, имеем *Eml* ⊇ *Ekl*.

Рассмотрим случай *xm* ∉ ***Im*** *y*. Поскольку *xm* ∉ ***Im*** *y*, переход от последовательности *x*[*m*‑1] к последовательности *x* = *x*[*m*‑1]⋅*xm* не добавляет новых общих *A‑*вложений. Поэтому *E* = *Em*. Отсюда |*E*| = |*Em*|, что и требовалось доказать в этом случае.

Рассмотрим случай *xm* ∈ ***Im*** *y* и *xm* ∉ ***Im*** *x*[*m*‑1]. Поскольку *xm* ∈ ***Im*** *y*, переход от последовательности *x*[*m*‑1] к последовательности *x* = *x*[*m*‑1]⋅*xm* может добавлять новые общие *A‑*вложения, эти общие *A‑*вложения должны заканчиваться на *xm*, но этих общих *A‑*вложений раньше, в *x*[*m*‑1] и *y*, не было, поскольку *xm* ∉ ***Im*** *x*[*m*‑1]. Заметим, что для такого нового общего *A‑*вложения *u* в его самых правых *E‑*вложениях в *x* и *y* последний непустой символ равен *xm* по индексам *m* и *l*, соответственно, т.е. *er*(*u*, *x*)*m* = *er*(*u*, *y*)*l* = *xm*. Имеем *E* = *Em* ∪ *Eml*. Поскольку *Em* ∩ *Eml* = ∅, имеем |*E*| = |*Em*| + |*Eml*|, что и требовалось доказать в этом случае.

Теперь рассмотрим случай *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1]. Поскольку *xm* ∈ ***Im*** *y*, переход от последовательности *x*[*m*‑1] к последовательности *x* = *x*[*m*‑1]⋅*xm* может добавлять новые общие *A‑*вложения *u*, но они новые только в том случае, когда таких *A‑*вложений не было раньше в *x*[*m*‑1] и *y*. Эти новые общие *A‑*вложения должны заканчиваться на *xm*. Заметим, что если такое общее *A‑*вложение *u* было раньше, то в его самых правых *E‑*вложениях в *x*[*m*‑1] и *y* последний непустой символ равен *xm* по индексам *k* и *l*, соответственно, т.е. *er*(*u*, *x*[*m*‑1])*k* = *er*(*u*, *y*)*l* = *xm*. В любом случае в самых правых *E‑*вложениях *u* в *x* и *y* последний непустой символ равен *xm* по индексам *m* и *l*, соответственно, т.е. *er*(*u*, *x*)*m* = *er*(*u*, *y*)*l* = *xm*. Имеем *E* = *Em* ∪ ( *Eml* \ *Ekl* ). Поскольку *Em* ∩ *Eml* = *Em* ∩ *Ekl* = ∅, имеем *Em* ∩ (*Eml* \ *Ekl* ) = ∅ и поэтому |*E*| = |*Em*| + |*Eml* \ *Ekl*|. Поскольку *Eml* ⊇ *Ekl*, имеем |*Eml* \ *Ekl*| = |*Eml*| ‑ |*Ekl*|. Поэтому |*E*| = |*Em*| + |*Eml*| ‑ |*Ekl*|, что и требовалось доказать в этом случае.

Теорема 1 определяет алгоритм вычисления *A*0(*x*, *y*). Число шагов алгоритма равно ***O***(*mn*), что определяется числом функций вида *A*0(*x*[*m* ‑ *i*], *y*[*n* ‑ *j*]), где *i* ∈ 0..*m* и *j* ∈ 0..*n*, при условии, что каждая функция вычисляется не более одного раза (после чего её значение сохраняется). На каждом шаге проверка условий *xm*‑*i* ∈ ***Im*** *y*[*n* ‑ *j*] и *xm*‑*i* ∈ ***Im*** *x*[*m* ‑ *i* ‑ 1] имеет сложность ***O***(*m* + *n*), а остальные вычисления имеют сложность ***O***(1). Сложность алгоритма равна ***O***(*m*2*n* + *mn*2), что для *m* ≤ *n* равно ***O***(*mn*2).

* 1. **Сумма длин общих *A‑*вложений**
1.

*A*1(*x*, *y*) = *A*1(*x*[*m*‑1], *y*), если *xm* ∉ ***Im*** *y*;

*A*1(*x*, *y*) = *A*1(*x*[*m*‑1], *y*) + *A*1(*x*[*m*‑1], *y*[*l*‑1]) + *A*0(*x*[*m*‑1], *y*[*l*‑1]), если *xm* ∈ ***Im*** *y* и *xm* ∉ ***Im*** *x*[*m*‑1];

*A*1(*x*, *y*) = *A*1(*x*[*m*‑1], *y*) + *A*1(*x*[*m*‑1], *y*[*l*‑1]) + *A*0(*x*[*m*‑1], *y*[*l*‑1]) ‑ *A*1(*x*[*k*‑1], *y*[*l*‑1]) ‑ *A*0(*x*[*k*‑1], *y*[*l*‑1]), если *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1].

Доказательство.

Доказательство аналогично доказательству теоремы 1. Используем те же обозначения:

*E* = *Ar*(*x*, *y*);

*Em* = *Ar*(*x*[*m*‑1], *y*)⋅(ε, ());

*Eml* = *Ar*(*x*[*m*‑1], *y*[*l*‑1])⋅(*xm*, *xm*⋅ε*n*‑*l*), если *xm* ∈ ***Im*** *y*, иначе *Eml* = ∅;

*Ekl* = *Ar*(*x*[*k*‑1], *y*[*l*‑1])⋅(*xm*⋅ε*m*‑*k*‑1⋅*xm*, *xm*⋅ε*n*‑*l*), если *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1], иначе *Ekl* = ∅.

Имеем *Em* ∩ *Eml* = *Em* ∩ *Ekl* = ∅ и *Eml* ⊇ *Ekl*.

Также имеем *A*0(*x*[*m*‑1], *y*[*l*‑1]) = |*Eml*| и |*A*0(*x*[*k*‑1], *y*[*l*‑1])| = |*Ekl*|.

Обозначим:

*S* = Σ { |*u*| : *u* ∈ *A*(*x*) ∩ *A*(*y*) & ( *er*(*u*, *x*), *er*(*u*, *y*) ) ∈ *E* },

*Sm* = Σ { |*u*| : *u* ∈ *A*(*x*) ∩ *A*(*y*) & ( *er*(*u*, *x*), *er*(*u*, *y*) ) ∈ *Em* },

*Sml* = Σ { |*u*| : *u* ∈ *A*(*x*) ∩ *A*(*y*) & ( *er*(*u*, *x*), *er*(*u*, *y*) ) ∈ *Eml* }, если *xm* ∈ ***Im*** *y*, иначе *Eml* = ∅,

*Skl* = Σ { |*u*| : *u* ∈ *A*(*x*) ∩ *A*(*y*) & ( *er*(*u*, *x*), *er*(*u*, *y*) ) ∈ *Ekl* }, если *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1], иначе *Ekl* = ∅.

Длина общего *A‑*вложения *u* равна числу непустых символов в каждом его *E‑*вложении, в том числе, в самом правом *E‑*вложении. Поэтому в этих обозначениях утверждение теоремы можно переписать в виде:

*S* = *Sm*, если *xm* ∉ ***Im*** *y*;

*S* = *Sm* + *Sml* + |*Eml*|, если *xm* ∈ ***Im*** *y* и *xm* ∉ ***Im*** *x*[*m*‑1];

*S* = *Sm* + *Sml* + |*Eml*| ‑ *Skl* ‑ |*Ekl*|, если *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1].

При переходе от последовательности *x*[*m*‑1] к последовательности *x* = *x*[*m*‑1]⋅*xm* число непустых символов в самом правом *E‑*вложении в *x* общего *A‑*вложения *u* не меняется, если в нём по индексу *m* оказывается пустой символ *er*(*u*, *x*)*m* = ε, и увеличивается на 1 в противном случае, когда *er*(*u*, *x*)*m* = *xm*.

Это влечёт следующие утверждения.

В случае *xm* ∉ ***Im*** *y* имеем *E* = *Em*, для каждого (*u*, *v*) ∈ *Em* имеет место *um* = ε, поэтому *S* = *Sm*, что и требовалось доказать в этом случае.

В случае *xm* ∈ ***Im*** *y* и *xm* ∉ ***Im*** *x*[*m*‑1] имеем *E* = *Em* ∪ *Eml* и *Em* ∩ *Eml* = ∅, для каждого (*u*, *v*) ∈ *Em* имеет место *um* = ε, для каждого (*u*, *v*) ∈ *Eml* имеет место *um* = *xm*, поэтому *S* = *Sm* + ( *Sml* + |*Eml*| ) , что и требовалось доказать в этом случае.

В случае *xm* ∈ ***Im*** *y* и *xm* ∈ ***Im*** *x*[*m*‑1] имеем *E* = *Em* ∪ ( *Eml* \ *Ekl* ) и *Em* ∩ *Eml* = *Em* ∩ *Ekl* = ∅, для каждого (*u*, *v*) ∈ *Em* имеет место *um* = ε, для каждого (*u*, *v*) ∈ *Eml* имеет место *um* = *xm*, для каждого (*u*, *v*) ∈ *Ekl* имеет место *um* = *xm*, поэтому *S* = *Sm* + ( *Sml* + |*Eml*| ) ‑ ( *Skl* + |*Ekl*| ), что и требовалось доказать в этом случае.

Теорема 2 определяет алгоритм вычисления *A*1(*x*, *y*), сложность которого по порядку, очевидно, не превышает сложности алгоритма вычисления *A*0(*x*, *y*), т.е. равна ***O***(*m*2*n* + *mn*2), что для *m* ≤ *n* равно ***O***(*mn*2).

* 1. **Cумма минимумов чисел *E‑*вложений общих *A‑*вложений**

Для функции *A*2(*x*, *y*) мы не знаем хорошего (отличного от полного перебора) алгоритма. Единственная полезная оптимизация — это предварительное удаление необщих символов.

* 1. **Сумма произведений чисел *E‑*вложений общих *A‑*вложений**

Здесь мы докажем теорему, аналогичную теореме 2 в [3] с тем отличием, что мы учитываем пустую подпоследовательность, а в теореме 2 в [3] она не учитывается. Также мы дадим своё доказательство, поскольку мы используем другие определения и обозначения.

1. *A*3(*x*, *y*) = *A*3(*x*[*m*‑1], *y*) + *A*3(*x*, *y*[*n*‑1]) ‑ *A*3(*x*[*m*‑1], *y*[*n*‑1]), если *xm* ≠ *yn*;

*A*3(*x*, *y*) = *A*3(*x*[*m*‑1], *y*) + *A*3(*x*, *y*[*n*‑1]), если *xm* = *yn*.

Доказательство.

Обозначим следующие множества пар *E‑*вложений общих *A‑*вложений:

*E* = *A*(*x*, *y*),

*E*00 = *A*(*x*[*m*‑1], *y*[*n*‑1])⋅(ε, ε) — пара последних элементов (ε, ε),

*E*01 = ( *A*(*x*[*m*‑1], *y*])⋅(ε, ()) ) \ *E*00 — пара последних элементов (ε, *yn*),

*E*10 = ( *A*(*x*, *y*[*n*‑1])⋅((), ε) ) \ *E*00 — пара последних элементов (*xm*, ε),

*E*11 = *E* \ (*E*00 ∪ *E*01 ∪ *E*10) — пара последних элементов (*xm*, *yn*), поскольку *E*00 ∪ *E*01 ∪ *E*10 содержат все пары *E‑*вложений общих *A‑*вложений, в которых пара последних элементов содержит ε.

Очевидно, *E* = *E*00 ∪ *E*01 ∪ *E*10 ∪ *E*11. Пары последних элементов пар *E‑*вложений из разных множеств *E*00, *E*01, *E*10, *E*11 разные, поэтому эти множества попарно не пересекаются. Поэтому |*E*| = |*E*00| + |*E*01| + |*E*10| + |*E*11|, |*E*00 ∪ *E*01| = |*E*00| + |*E*01|, |*E*00 ∪ *E*10| = |*E*00| + |*E*10|.

Поскольку *A*3(*x*, *y*) = |*A*(*x*, *y*)|, в этих обозначениях утверждение теоремы имеет вид

|*E*| = (|*E*00| + |*E*01|) + (|*E*00| + |*E*10|) ‑ |*E*00| = |*E*00| + |*E*01| + |*E*10|, если *xm* ≠ *yn*,

|*E*| = (|*E*00| + |*E*01|) + (|*E*00| + |*E*10|) = 2|*E*00| + |*E*01| + |*E*10|, если *xm* = *yn*.

Рассмотрим случай *xm* ≠ *yn*. В этом случае *E*11 = ∅, что влечёт |*E*11| = 0 и |*E*| = |*E*00| + |*E*01| + |*E*10|, что и требовалось доказать в этом случае.

Рассмотрим случай *xm* = *yn* = *h*. Каждая пара *E‑*вложений из *E*11 имеет вид (*u*⋅ε*m*‑|*u*|‑1⋅*h*, *v*⋅ε*n*‑|*v*|‑1⋅*h*), где (*u*⋅ε*m*‑|*u*|‑1⋅ε, *v*⋅ε*n*‑|*v*|‑1⋅ε) ∈ *E*00 (последовательности *u* и *v* могут быть обе пустыми). Будем говорить, что пара (*u*⋅ε*m*‑|*u*|‑1⋅*h*, *v*⋅ε*n*‑|*v*|‑1⋅*h*) соответствует паре (*u*⋅ε*m*‑|*u*|‑1⋅ε, *v*⋅ε*n*‑|*v*|‑1⋅ε). Это соответствие, очевидно, является биекцией множеств *E*11 и *E*00. Тем самым, |*E*11| = |*E*00|. Поэтому |*E*| = |*E*00| + |*E*01| + |*E*10| + |*E*11| = |*E*00| + |*E*01| + |*E*10| + |*E*00| = 2|*E*00| + |*E*01| + |*E*10|, что и требовалось доказать в этом случае.

Теорема 3 определяет алгоритм вычисления *A*3(*x*, *y*). Число шагов алгоритма равно ***O***(*mn*), что определяется числом функций вида *A*3(*x*[*m* ‑ *i*], *y*[*n* ‑ *j*]), где *i* ∈ 0..*m* и *j* ∈ 0..*n*, при условии, что каждая функция вычисляется не более одного раза (после чего её значение сохраняется). На каждом шаге вычисления имеют сложность ***O***(1). Тем самым сложность алгоритма равна ***O***(*mn*).

* 1. **Функция похожести на основе наибольшей длины общего *A*‑вложения**

Задача вычисления длины наибольшей общей подпоследовательности (*longest common subsequence*, *lcs*) хорошо известна [1]. Простейший алгоритм сложности ***O***(*mn*) основан на следующих соотношениях:

*lcA*(*x*, *y*) = *lcA*(*x*[*m* ‑1], *y*[*n* ‑1]) + 1, если *xm* = *yn*;

*lcA*(*x*, *y*) = *max*{ *lcA*(*x*, *y*[*n* ‑1]), *lcA*(*x*[*m* ‑1], *y*) }, если *xm* ≠ *yn*.

Тем самым, функция *A*4(*x*, *y*) = *lcA*(*x*, *y*) вычисляется за время ***O***(*mn*).

1. ***L‑*вложения**

В отличие от *A*‑ и *O*‑вложений *L*‑вложению *u* ∈ *L*(*x*) соответствует только одно *E*‑вложение *v* такое, что λ(*v*) = *u*. Общему *L*‑вложению *u* ∈ *L*(*x*) ∩ *L*(*y*) соответствует пара *E*‑вложений в *x* и *y*, которые могут отличаться только префиксом пустых символов. Поэтому множество *l*(*u*, *x*) является синглетоном, { *ll*(*u*, *x*) } = { *lr*(*u*, *x*) } = *l*(*u*, *x*), *L*(*x*, *y*) = *Ll*(*x*, *y*) = *Lr*(*x*, *y*).

Обозначим через γ(*x*, *y*) последовательность *z* длиной *m* (напомним, что |*x*| = *m* ≤ *n* = |*y*|), совпадающую с *x* и *y* по тем позициям, считая справа налево, по которым совпадают *x* и *y*, и содержащую пустой символ по остальным позициям: |*z*| = *m* и ∀ *i* ∈ 1..*m* ( *xm*+ 1 ‑*i* = *yn*+ 1 ‑*i* ⇒ *zm*+ 1 ‑*i* = *xm*+ 1 ‑*i*) & ( *xm*+ 1 ‑*i* ≠ *yn*+ 1 ‑*i* ⇒ *zm*+ 1 ‑*i* = ε). Функция γ(*x*, *y*) устанавливает позиционное соответствие совпадающих символов *x* и *y* при счёте позиций справа налево.

* 1. **Число общих *L‑*вложений**
1. *L*0(*x*, *y*) = *L*0( *x*[*m*‑1], *y*[*n*‑1]) ), если *xm* ≠ *yn*,

*L*0(*x*, *y*) = 2*L*0( *x*[*m*‑1], *y*[*n*‑1] ), если *xm* = *yn*.

Доказательство.

Обозначим *L* = *L*(*x*) ∩ *L*(*y*) и *L*‑1 = *L*(*x*[*m*‑1]) ∩ *L*(*y*[*n*‑1]). Если *xm* ≠ *yn*, то *L*‑вложению *u* ∈ *L*‑1 соответствует одно *L*‑вложение *u*⋅ε ∈ *L*. Тем самым, *L*0(*x*, *y*) = |*L*| = |*L*‑1| = *L*0( *x*[*m*‑1], *y*[*n*‑1]) ). Если *xm* = *yn*, то *L*‑вложению *u* ∈ *L*‑1 соответствуют два *L*‑вложения *u*⋅ε ∈ *L* и *u*⋅*xm* ∈ *L*. Тем самым, *L*0(*x*, *y*) = |*L*| = 2|*L*‑1| = 2*L*0( *x*[*m*‑1], *y*[*n*‑1]) ).

Теорема 4 определяет алгоритм вычисления *L*0(*x*, *y*). Для *m* ≤ *n* число шагов алгоритма равно ***O***(*m*), что определяется числом функций вида *L*0(*x*[*m* ‑ *i*], *y*[*n* ‑ *i*]), где *i* ∈ 0..*m*. На каждом шаге вычисления имеют сложность ***O***(1). Тем самым сложность алгоритма равна ***O***(*m*).

1. *L*0(*x*, *y*) = 2|μγ(*x*,*y*)|.

Доказательство.

Из теоремы 4 следует, что при добавлении к обеим последовательностям справа по одному символу число общих *L*‑вложений увеличивается вдвое, если добавляются равные символы, и не меняется в противном случае. Поскольку |*L*(*x*) ∩ *L*(())| = |*L*(()) ∩ *L*(*y*)| = |ε| = 1, а |μγ(*x*, *y*)| равно числу совпадающих символов в одинаковых позициях *x* и *y* при отсчёте справа налево, имеем *L*0(*x*, *y*) = |*L*(*x*) ∩ *L*(*y*)| = 2|μγ(*x*,*y*)|, что и требовалось доказать.

Теорема 5 определяет алгоритм вычисления *L*0(*x*, *y*). Сложность алгоритма равна сложности вычисления μγ(*x*, *y*), которая, очевидно, для *m* ≤ *n* равна ***O***(*m*), плюс сложность возведения числа два в степень |μγ(*x*, *y*)|, которая тоже равна ***O***(*m*). Тем самым сложность алгоритма равна ***O***(*m*).

* 1. **Сумма μ‑длин общих *L*‑вложений**
1. *L*1(*x*, *y*) = *L*1( *x*[*m*‑1], *y*[*n*‑1]) ), если *xm* ≠ *yn*,

*L*1(*x*, *y*) = 2*L*1( *x*[*m*‑1], *y*[*n*‑1] + *L*0( *x*[*m*‑1], *y*[*n*‑1] ), если *xm* = *yn*.

Доказательство.

Обозначим *L* = *L*(*x*) ∩ *L*(*y*) и *L*‑1 = *L*(*x*[*m*‑1]) ∩ *L*(*y*[*n*‑1]). Если *xm* ≠ *yn*, то *L*‑вложению *u* ∈ *L*‑1 соответствует одно *L*‑вложение *u*⋅ε ∈ *L* той же μ‑длины. Тем самым, *L*1(*x*, *y*) = Σ { |μ(*u*)| : *u* ∈ *L* } = = Σ { |μ(*u*)| : *u* ∈ *L*‑1 } = *L*1( *x*[*m*‑1], *y*[*n*‑1]) ). Если *xm* = *yn*, то *L*‑вложению *u* ∈ *L*‑1 соответствуют два *L*‑вложения *u*⋅ε ∈ *L* той же μ‑длины и *u*⋅*xm* ∈ *L* с μ‑длиной на 1 большей. Тем самым, *L*1(*x*, *y*) = = Σ { |μ(*u*)| : *u* ∈ *L* } = Σ { |μ(*u*)| : *u* ∈ *L*‑1 } + ( Σ { |μ(*u*)| : *u* ∈ *L*‑1 } + *L*0( *x*[*m*‑1], *y*[*n*‑1]) ) = = 2Σ { |μ(*u*)| : *u* ∈ *L*‑1 } + *L*0( *x*[*m*‑1], *y*[*n*‑1]) = 2*L*1( *x*[*m*‑1], *y*[*n*‑1] + + *L*0( *x*[*m*‑1], *y*[*n*‑1] ).

Теорема 6 определяет алгоритм вычисления *L*1(*x*, *y*), сложность которого по порядку, очевидно, не превышает сложности алгоритма вычисления *L*0(*x*, *y*), т.е. равна ***O***(*m*).

1. *L*1(*x*, *y*) = 1\*C*l*1 + 2\*C*l*2 + ... + *l*\*C*ll* ([A001787](http://oeis.org/A001787)), где *l* = |μγ(*x*, *y*)|.

Доказательство.

Утверждение теоремы непосредственно следует из того факта, что число общих *L*‑вложений μ‑длины *i* равно C|μγ(*x*,*y*)|*i*.

Теорема 7 определяет алгоритм вычисления *L*1(*x*, *y*). Сложность алгоритма равна сложности вычисления μγ(*x*, *y*), которая, очевидно, для *m* ≤ *n* равна ***O***(*m*), плюс сложность вычисления факториалов *i*! для *i* ∈ 0..*l*, равная ***O***(*m*), плюс сложность суммирования *l* чисел, равная ***O***(*m*). Тем самым сложность алгоритма равна ***O***(*m*).

* 1. **Cумма минимумов и сумма произведений чисел *E‑*вложений общих *L‑*вложений**
1. *L*2(*x*, *y*) = *L*3(*x*, *y*) = *L*0(*x*, *y*).

Доказательство.

Утверждение теоремы непосредственно следует из того факта, что *L*‑вложению *u* ∈ *L*(*x*) соответствует ровно одно *E*‑вложение *v* такое, что λ(*v*) = *u*: *L*2(*x*, *y*) = = Σ { *min*{ |*l*(*u*, *x*)|, |*l*(*u*, *y*)| } : *u* ∈ *L*(*x*) ∩ *L*(*y*) } = Σ { *min*{ 1, 1 } : *u* ∈ *L*(*x*) ∩ *L*(*y*) } = | *L*(*x*) ∩ *L*(*y*) | = = *L*0(*x*, *y*); *L*3(*x*, *y*) = |*L*(*x*, *y*)| = |∪ { *l*(*u*, *x*) × *l*(*u*, *y*) : *u* ∈ *L*(*x*) ∩ *L*(*y*) }| = | *L*(*x*) ∩ *L*(*y*) | = *L*0(*x*, *y*).

Теорема 8 определяет алгоритм вычисления *L*2(*x*, *y*) и *L*3(*x*, *y*) той же сложности, что алгоритм вычисления *L*0(*x*, *y*), т.е. для *m* ≤ *n* сложности ***O***(*m*).

* 1. **Функция похожести на основе наибольшей длины общего *L*‑вложения**
1.

*lcL*(*x*, *y*) = *lcL*(*x*[*m* ‑1], *y*[*n* ‑1]) + 1, если *xm* = *yn*;

*lcL*(*x*, *y*) = *lcL*(*x*[*m* ‑1], *y*[*n* ‑1]), если *xm* ≠ *yn*.

Доказательство. Функция γ(*x*, *y*) устанавливает позиционное соответствие совпадающих символов *x* и *y* при счёте позиций справа налево. Поэтому *lcL*(*x*, *y*) = |μγ(*x*, *y*)|. Отсюда непосредственно следует утверждение теоремы.

Теорема 9 определяет алгоритм вычисления функции *L*4(*x*, *y*) = *lcL*(*x*, *y*) сложности ***O***(*m*) для *m* ≤ *n*.

1. ***R‑*вложения**

Аналогично *L*‑вложению *R*‑вложению *u* ∈ *R*(*x*) соответствует только одно *E*‑вложение *v* такое, что ρ(*v*) = *u*. Общему *R*‑вложению *u* ∈ *R*(*x*) ∩ *R*(*y*) соответствует пара *E*‑вложений в *x* и *y*, отличающихся только постфиксом пустых символов. Поэтому множество *r*(*u*, *x*) является синглетоном, { *rl*(*u*, *x*) } = { *rr*(*u*, *x*) } = *r*(*u*, *x*), *R*(*x*, *y*) = *Rl*(*x*, *y*) = *Rr*(*x*, *y*).

Обозначим через δ(*x*, *y*) последовательность *z* длиной *m* (напомним, что |*x*| = *m* ≤ *n* = |*y*|), совпадающую с *x* и *y* по тем позициям, считая слева направо, по которым совпадают *x* и *y*, и содержащую пустой символ по остальным позициям: |*z*| = *m* и ∀ *i* ∈ 1..*m* ( *xi* = *yi* ⇒ *zi* = *xi*) & ( *xi* ≠ *yi* ⇒ *zi* = ε). Функция δ(*x*, *y*) устанавливает позиционное соответствие совпадающих символов *x* и *y* при счёте позиций слева направо.

* 1. **Число общих *R‑*вложений**
1. *R*0(*x*, *y*) = *R*0( *x*[2..*m*], *y*[2..*n*]) ), если *x*1 ≠ *y*1,

*R*0(*x*, *y*) = 2*R*0( *x*[2..*m*], *y*[2..*n*] ), если *x*1 = *y*1.

Доказательство.

Обозначим *R* = *R*(*x*) ∩ *R*(*y*) и *R*‑1 = *R*(*x*[2..*m*]) ∩ *R*(*y*[2..*n*]). Если *x*1 ≠ *y*1, то *R*‑вложению *u* ∈ *R*‑1 соответствует одно *R*‑вложение ε⋅*u* ∈ *R*. Тем самым, *R*0(*x*, *y*) = |*R*| = |*R*‑1| = *R*0( *x*[2..*m*], *y*[2..*n*]) ). Если *x*1 = *y*1, то *R*‑вложению *u* ∈ *R*‑1 соответствуют два *R*‑вложения ε⋅*u* ∈ *R* и *x*1⋅*u*. Тем самым, *R*0(*x*, *y*) = |*R*| = 2|*R*‑1| = 2*R*0( *x*[2..*m*], *y*[2..*n*]) ).

Теорема 10 определяет алгоритм вычисления *R*0(*x*, *y*). Для *m* ≤ *n* число шагов алгоритма равно ***O***(*m*), что определяется числом функций вида *R*0(*x*[*i*..*m*], *y*[*i*..*n*]), где *i* ∈ 0..*m*. На каждом шаге проверка вычисления имеют сложность ***O***(1). Тем самым сложность алгоритма равна ***O***(*m*).

1. *R*0(*x*, *y*) = 2|μδ(*x*,*y*)|.

Доказательство.

Из теоремы 10 следует, что при добавлении к обеим последовательностям слева по одному символу число общих *R*‑вложений увеличивается вдвое, если добавляются равные символы, и не меняется в противном случае. Поскольку |*R*(*x*) ∩ *R*(())| = |*R*(()) ∩ *R*(*y*)| = |ε| = 1, а |μδ(*x*, *y*)| равно числу совпадающих символов в одинаковых позициях *x* и *y* при отсчёте слева направо, имеем *R*0(*x*, *y*) = |*R*(*x*) ∩ *R*(*y*)| = 2|μδ(*x*,*y*)|, что и требовалось доказать.

Теорема 11 определяет алгоритм вычисления *R*0(*x*, *y*). Сложность алгоритма равна сложности вычисления μδ(*x*, *y*), которая, очевидно, для *m* ≤ *n* равна ***O***(*m*), плюс сложность возведения числа два в степень |μδ(*x*, *y*)|, которая тоже равна ***O***(*m*). Тем самым сложность алгоритма равна ***O***(*m*).

* 1. **Сумма μ‑длин общих *R*‑вложений**
1. *R*1(*x*, *y*) = *R*1( *x*[2..*m*], *y*[2..*n*]) ), если *x*1 ≠ *y*1,

*R*1(*x*, *y*) = 2*R*1( *x*[2..*m*], *y*[2..*n*] + *R*0( *x*[2..*m*], *y*[2..*n*] ), если *x*1 = *y*1.

Доказательство.

Обозначим *R* = *R*(*x*) ∩ *R*(*y*) и *R*‑1 = *R*(*x*[2..*m*]) ∩ *R*(*y*[2..*n*]). Если *x*1 ≠ *y*1, то *R*‑вложению *u* ∈ *R*‑1 соответствует одно *R*‑вложение ε⋅*u* ∈ *R* той же μ‑длины. Тем самым, *R*1(*x*, *y*) = Σ { |μ(*u*)| : *u* ∈ *R* } = = Σ { |μ(*u*)| : *u* ∈ *R*‑1 } = *R*1( *x*[2..*m*], *y*[2..*n*]) ). Если *x*1 = *y*1, то *R*‑вложению *u* ∈ *R*‑1 соответствуют два *R*‑вложения ε⋅*u* ∈ *R* той же μ‑длины и *x*1⋅*u* ∈ *R* с μ‑длиной на 1 большей. Тем самым, *R*1(*x*, *y*) = = Σ { |μ(*u*)| : *u* ∈ *R* } = Σ { |μ(*u*)| : *u* ∈ *R*‑1 } + ( Σ { |μ(*u*)| : *u* ∈ *R*‑1 } + + *R*0( *x*[2..*m*], *y*[2..*n*]) ) = = 2Σ { |μ(*u*)| : *u* ∈ *R*‑1 } + *R*0( *x*[2..*m*], *y*[2..*n*]) = 2*R*1( *x*[2..*m*], *y*[2..*n*] + + *R*0( *x*[2..*m*], *y*[2..*n*] ).

Теорема 12 определяет алгоритм вычисления *R*1(*x*, *y*), сложность которого по порядку, очевидно, не превышает сложности алгоритма вычисления *R*0(*x*, *y*), т.е. для *m* ≤ *n* равна ***O***(*m*).

1. *R*1(*x*, *y*) = 1\*C*r*1 + 2\*C*r*2 + ... + *r*\*C*rr* ([A001787](http://oeis.org/A001787)), где *r* = |μδ(*x*, *y*)|.

Доказательство.

Утверждение теоремы непосредственно следует из того факта, что число общих *R*‑вложений μ‑длины *i* равно C|μδ(*x*,*y*)|*i*.

Теорема 13 определяет алгоритм вычисления *R*1(*x*, *y*). Сложность алгоритма равна сложности вычисления μδ(*x*, *y*), которая, очевидно, для *m* ≤ *n* равна ***O***(*m*), плюс сложность вычисления факториалов *i*! для *i* ∈ 0..*r*, равная ***O***(*m*), плюс сложность суммирования *r* чисел, равная ***O***(*m*). Тем самым сложность алгоритма равна ***O***(*m*).

* 1. **Cумма минимумов и сумма произведений чисел *E‑*вложений общих *R‑*вложений**
1. *R*2(*x*, *y*) = *R*3(*x*, *y*) = *R*0(*x*, *y*).

Доказательство.

Утверждение теоремы непосредственно следует из того факта, что *R*‑вложению *u* ∈ *R*(*x*) соответствует ровно одно *E*‑вложение *v* такое, что ρ(*v*) = *u*: *R*2(*x*, *y*) = = Σ { *min*{ |*r*(*u*, *x*)|, |*r*(*u*, *y*)| } : *u* ∈ *R*(*x*) ∩ *R*(*y*) } = Σ { *min*{ 1, 1 } : *u* ∈ *R*(*x*) ∩ *R*(*y*) } = | *R*(*x*) ∩ *R*(*y*) | = = *R*0(*x*, *y*); *R*3(*x*, *y*) = |*R*(*x*, *y*)| = |∪ { *r*(*u*, *x*) × *r*(*u*, *y*) : *u* ∈ *R*(*x*) ∩ *R*(*y*) }| = | *R*(*x*) ∩ *R*(*y*) | = *R*0(*x*, *y*).

Теорема 14 определяет алгоритм вычисления *R*2(*x*, *y*) и *R*3(*x*, *y*) той же сложности, что алгоритм вычисления *R*0(*x*, *y*), т.е. для *m* ≤ *n* сложности ***O***(*m*).

* 1. **Функция похожести на основе наибольшей длины общего *R*‑вложения**
1.

*lcR*(*x*, *y*) = *lcR*(*x*[2..*m*], *y*[2..*n*]) + 1, если *x*1 = *y*1;

*lcR*(*x*, *y*) = *lcR*(*x*[2..*m*], *y*[2..*n*]), если *x*1 ≠ *y*1.

Доказательство. Функция δ(*x*, *y*) устанавливает позиционное соответствие совпадающих символов *x* и *y* при счёте позиций слева направо. Поэтому *lcR*(*x*, *y*) = |μδ(*x*, *y*)|. Отсюда непосредственно следует утверждение теоремы.

Теорема 15 определяет алгоритм вычисления функции *R*4(*x*, *y*) = *lcR*(*x*, *y*) сложности ***O***(*m*) для *m* ≤ *n*.

1. ***O‑*вложения**
	1. **Число общих *O‑*вложений**

Для случая *xm* = *yn* = *h* обозначим через *I* = { *i* ∈ 1..*m* : *xi* = *h* } и *J* = { *j* ∈ 1..*n* : *yj* = *h* } множества индексов по которым находится символ *h* в *x* и *y*, соответственно.

Обозначим для *i* ∈ *I*, *j* ∈ *J*: *Li*,*j* = *L*(*x*[*i*‑1]) ∩ *L*(*y*[*j*‑1]).

Обозначим *K* = ( *I* × *J* ) \ { (*m*, *n*) } и *Lh*(*x*, *y*) = *Lm*,*n* \ ∪ { *Li*,*j* : (*i*, *j*)∈ *K* }.

Для *i* ∈ *I* и *j* ∈ *J* обозначим через *ui*,*j* максимальное общее *L*‑вложение в *x*[*i*‑1] и *y*[*j*‑1]: *ui*,*j* = λ(*vi*,*j*), где *vi*,*j* общее *E*‑вложение в *x*[*i*‑1] и *y*[*j*‑1], определяемое условием: ∀ *t* = 1..*min*{*i*, *j*}‑1 ( *xi*‑*t* = *yi*‑*t* ⇒ *vi*,*j*(*i* ‑ *t*) = *xi*‑*t* ) & ( *xi*‑*t* ≠ *yi*‑*t* ⇒ *vi*,*j*(*i* ‑ *t*) = ε ). Множество всех общих *L*‑вложений в *x*[*i*‑1] и *y*[*j*‑1] равно λ*E*(*ui*,*j*), т.е. получается из *ui*,*j* всеми возможными заменами некоторых символов на пустой символ и последующим удалением префикса пустых символов.

1. Пусть *xm* = *yn* = *h*. Вычисление |*Lh*(*x*, *y*)| эквивалентно вычислению числа слагаемых СДНФ некоторой конъюнкции дизъюнкций переменных без отрицаний, где число переменных равно числу непустых символов в максимальном общем *L*‑вложении *um*,*n* в *x*[*m*‑1] и *y*[*n*‑1].

Доказательство.

*Lh*(*x*, *y*) = *Lm*,*n* \ ∪ { *Li*,*j* : (*i*, *j*)∈ *K* } = *Lm*,*n* \ ∪ { *Lm*,*n* ∩ *Li*,*j* : (*i*, *j*)∈ *K* }. Для получения *Lh*(*x*, *y*) нам нужно из *Lm*,*n* удалить множество *Lm*,*n* ∩ *Li*,*j* для каждых (*i*, *j*)∈ *K*.

Определим «пересечение» *ui*,*j*^ вложений *um*,*n* и *ui*,*j*, имеющее длину |*um*,*n*| и совпадающее с *um*,*n* и *ui*,*j* в тех позициях, считая справа налево, в которых они совпадают между собой, а во всех остальных позициях содержит пустой символ: |*ui*,*j*^| = |*um*,*n*| и ∀ *t* = 1..|*um*,*n*| ( *t* > |*ui*,*j*| ⇒ *ui*,*j*^(|*um*,*n*| ‑ *t*) = ε ) & & ( *t* ≤ |*ui*,*j*| & *um*,*n*(|*um*,*n*| ‑ *t*) = *ui*,*j*(|*ui*,*j*| ‑ *t*) ⇒ *ui*,*j*^(|*um*,*n*| ‑ *t*) = *um*,*n*(|*um*,*n*| ‑ *t*) ) & & ( *t* ≤ |*ui*,*j*| & *um*,*n*(|*um*,*n*| ‑ *t*) ≠ *ui*,*j*(|*ui*,*j*| ‑ *t*) ⇒ *ui*,*j*^(|*um*,*n*| ‑ *t*) = ε ). Очевидно, *um*,*n^* = *um*,*n*.

Удалим из вложения *ui*,*j*^ символы в тех позициях, в которых *um*,*n* содержит пустой символ, и обозначим результат *wi*,*j*: если *ui*,*j*^ = *u*1⋅*h*1⋅*u*2⋅*h*2⋅...⋅*uk*⋅*hk*‑1⋅*uk*, *um*,*n* = *v*1⋅ε⋅*v*2⋅ε⋅...⋅*vk*‑1⋅ε⋅*vk* и для *i* ∈ 1..*k* |*ui*| = |*vi*| и *vi* ∈ *H*\*, то *wi*,*j* = *u*1⋅*u*2⋅...⋅*uk*‑1⋅*uk*. Очевидно, *wm*,*n* = μ(*um*,*n*).

Все *wi*,*j* имеют одинаковую длину |*wm*,*n*|, равную числу непустых символов в *um*,*n*. Каждому *L*‑вложению из *Lm*,*n* ∩ *Li*,*j* взаимно-однозначно соответствует *E*‑вложение в *wi*,*j*, число таких вложений равно 2*k*, где *k* = |μ(*wi*,*j*)| число непустых символов в *wi*,*j*.

Поставим каждому *t* ∈ 1..|*wm*,*n*| в соответствие булеву переменную α*t*, означающую, что в позиции *t* может быть непустой символ. Тогда *E*‑вложения в *wi*,*j* задаются булевой функцией *Fi*,*j* = &{ ¬α*t* : *t* ∈ 1..|*wm*,*n*| & *wi*,*j*(*t*) = ε }, которая принимает значение ***true*** на тех и только тех наборах α1, ..., в которых α*t* = ***false*** для всех тех индексов *t*, по которым в *wi*,*j* и, следовательно, в любом *E*‑вложении в *wi*,*j* находится пустой символ. Если по индексу *t* в *wi*,*j* находится непустой символ, то в одних *E*‑вложениях в *wi*,*j* в этой позиции будет пустой символ, а в других непустой символ, что означает, что функция *Fi*,*j* не зависит от α*t*. Очевидно, что *Fm*,*n* = &∅ = ***true***.

Разность *Lm*,*n* \ ∪ { *Lm*,*n* ∩ *Li*,*j* : (*i*, *j*)∈ *K* } задаётся булевой функцией *F* = *Fm*,*n* \ ∨ { *Fi*,*j* : (*i*, *j*)∈ *K* } = = & { ¬*Fi*,*j* : (*i*, *j*)∈ *K* }, и ¬*Fi*,*j* = ∨{ α*t* : *t* ∈ 1..|*wm*,*n*| & *wi*,*j*(*t*) = ε }. Таким образом, *F* – это конъюнкция дизъюнкций переменных без отрицаний, и число |*Lh*(*x*, *y*)| равно числу слагаемых в СДНФ функции *F*.

1.

*O*0(*x*, *y*) = *O*0(*x*[*m*‑1], *y*) + *O*0(*x*, *y*[*n*‑1]) ‑ *O*0(*x*[*m*‑1], *y*[*n*‑1]), если *xm* ≠ *yn*.

*O*0(*x*, *y*) = *O*0(*x*[*m*‑1], *y*) + *O*0(*x*, *y*[*n*‑1]) ‑ *O*0(*x*[*m*‑1], *y*[*n*‑1]) + *Lh*(*x*, *y*]), если *xm* = *yn*.

Доказательство.

Обозначим следующие множества пар *E‑*вложений:

*E* = *Or*(*x*, *y*),

*E*00 = *Or*(*x*[*m*‑1], *y*[*n*‑1])⋅(ε, ε) — пара последних элементов (ε, ε),

*E*01 = ( *Or*(*x*[*m*‑1], *y*])⋅(ε, ()) ) \ *E*00 — пара последних элементов (ε, *yn*),

*E*10 = ( *Or*(*x*, *y*[*n*‑1])⋅((), ε) ) \ *E*00 — пара последних элементов (*xm*, ε),

*E*11 = *E* \ (*E*00 ∪ *E*01 ∪ *E*10) — пара последних элементов (*xm*, *yn*), поскольку *E*00 ∪ *E*01 ∪ *E*10 содержат все пары *E‑*вложений общих *O‑*вложений, в которых пара последних элементов содержит ε.

Очевидно, *E* = *E*00 ∪ *E*01 ∪ *E*10 ∪ *E*11. Пары последних элементов пар *E‑*вложений из разных множеств *E*00, *E*01, *E*10, *E*11 разные, поэтому эти множества попарно не пересекаются. Поэтому |*E*| = |*E*00| + |*E*01| + |*E*10| + |*E*11|, |*E*00 ∪ *E*01| = |*E*00| + |*E*01|, |*E*00 ∪ *E*10| = |*E*00| + |*E*10|.

Поскольку для любых *x*, *y* имеет место *O*0(*x*, *y*) = |*O*(*x*) ∩ *O*(*y*)| = |*Or*(*x*, *y*)| и *L*0(*x*, *y*) = *L*3(*x*, *y*) = |*L*(*x*, *y*)|, утверждение теоремы можно переписать в виде:

|*E*| = (|*E*00| + |*E*01|) + (|*E*00| + |*E*10|) ‑ |*E*00| = |*E*00| + |*E*01| + |*E*10|, если *xm* ≠ *yn*,

|*E*| = (|*E*00| + |*E*01|) + (|*E*00| + |*E*10|) ‑ |*E*00| + |*Lh*(*x*, *y*)| = |*E*00| + |*E*01| + |*E*10| + |*Lh*(*x*, *y*)|, если *xm* = *yn*.

Рассмотрим случай *xm* ≠ *yn*. В этом случае *E*11 = ∅, что влечёт |*E*11| = 0 и |*E*00| = |*E*00| + |*E*01| + |*E*10|, что и требовалось доказать в этом случае.

Рассмотрим случай *xm* = *yn* = *h*. В этом случае каждая пара *E‑*вложений из *E*11 имеет вид (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*), где (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*) ∈ *Lh*(*x*, *y*), т.е. при переходе от *x*[*m*‑1] и *y*[*n*‑1] к *x* и *y* эта пара образована добавлением справа символа *h* к паре *E‑*вложений общих *L*‑вложений в *x*[*m*‑1] и *y*[*n*‑1], но таких, которых не было раньше, т.е. которые не являются парой *E‑*вложений общих *L*‑вложений в *x*[*i*‑1] и *y*[*j*‑1], где *xi* = *yj* = *h* и *i* < *m* или *j* < *n*. Будем говорить, что пара (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*) соответствует паре (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*). Это соответствие, очевидно, является биекцией множеств *E*11 и *Lh*(*x*, *y*). Тем самым, |*E*11| = |*Lh*(*x*, *y*)|. Поэтому |*E*| = |*E*00| + |*E*01| + |*E*10| + |*E*11| = |*E*00| + |*E*01| + |*E*10| + |*Lh*(*x*, *y*)|, что и требовалось доказать в этом случае.

Теорема 16 определяет алгоритм вычисления *O*0(*x*, *y*). Число шагов алгоритма равно ***O***(*mn*), что определяется числом функций вида *O*0(*x*[*m* ‑ *i*], *y*[*n* ‑ *j*]), где *i* ∈ 0..*m* и *j* ∈ 0..*n*, при условии, что каждая функция вычисляется не более одного раза (после чего её значение сохраняется). На каждом шаге вычисления имеют сложность ***O***(1). Тем самым сложность алгоритма равна ***O***(*mn*)\****O***(*Lh*(*x*, *y*)), где ***O***(*Lh*(*x*, *y*)) сложность вычисления функции |*Lh*(*x*, *y*)|.

1. Пусть имеется *k* не обязательно различных множеств *a*(*s*), *s* ∈ 1..*k*. Обозначим для *S* ⊆ 1..*k*, *S* ≠ ∅, пересечение множеств *a*(*s*), индексы которых пробегают множество *S*, через *c*(*S*) = ∩{ *a*(*s*) : *s* ∈ *S* } и сумму числа подмножеств множества *c*(*S*) по всем *S*, для которых |*S*| = *l*, через *E*(*l*) = Σ{ 2|*c*(*S*)| : *S* ⊆ 1..*k* & |*S*| = *l* }. Тогда число различных множеств, вложенных хотя бы в одно из множеств *a*(*s*), *s* ∈ 1..*k*, равно чередующейся сумме *E*(1) ‑ *E*(2) + *E*(3) ‑ *E*(4) ... (‑1)*k*+1*E*(*k*), а сумма *F* размеров этих множеств, увеличенных на 1, равна чередующейся сумме *F*(1) ‑ *F*(2) + *F*(3) ‑ *F*(4) ... (‑1)*k*+1*F*(*k*), где *F*(*l*) = Σ{ (|*c*(*S*)| + 2) 2|*c*(*S*)|‑1 : *S* ⊆ 1..*k* & |*S*| = *l* }. Эти чередующиеся суммы могут быть вычислены за время ***O***(2*k*) при условии, что за время ***O***(1) могут выполняться операции пересечения двух множеств (а также арифметические операции над целыми числами).

Доказательство.

Обозначим число различных множеств, вложенных хотя бы в одно из множеств *a*(*s*), *s* ∈ 1..*k*, через *E*(*a*(1), ..., *a*(*k*)), а сумму их размеров, увеличенных на 1, через *F*(*a*(1), ..., *a*(*k*)).

Будем вести доказательство индукцией по *k*. Для *k* = 1 имеется единственное множество *S* ⊆ 1..1, *S* ≠ ∅, а именно *S* = {1}, и *c*({1}) = ∩{ *a*(*s*) : *s* ∈ {1} } = *a*(1). Число различных подмножеств множества *a*(1) равно 2|*a*(1)| и *E*(*a*(1)) = *E*(1) = 2|*a*(1)|, а сумма их длин, увеличенных на 1, равна 1\**Cr*0 + 2\**Cr*1 + ... + *r*\**Crr*‑1 + (*r*+1)\**Crr* = (*r* + 2)2*r*‑1 ([A001792](http://oeis.org/A001792)), и *F*(*a*(1)) = *F*(1) = (*r* + 2) 2*r*‑1, где *r* = |*a*(1)|.

Пусть утверждение верно для *k* и докажем его для *k*+1.

Число различных множеств, вложенных хотя бы в одно из множеств *a*(*s*), где *s* ∈ 1..*k*+1, равно числу различных множеств, вложенных хотя бы в одно из множеств *a*(*s*), где *s* ∈ 1..*k*, т.е. *E*(*a*(1), ..., *a*(*k*)), плюс число различных множеств, вложенных в *a*(*k*+1), т.е. 2|*a*(*k*+1)|, кроме тех, что вложены в пересечение *a*(*k*+1) с объединением множеств *a*(1), ..., *a*(*k*), число которых равно *E*(*a*(*k*+1) ∩ *a*(1), ..., *a*(*k*+1) ∩ *a*(*k*)).

Обозначим *Ek*+1 = *E*(*a*(1), ..., *a*(*k*+1)), *Ek* = *E*(*a*(1), ..., *a*(*k*)), *E^k* = *E*(*a*(*k*+1) ∩ *a*(1), ..., *a*(*k*+1) ∩ *a*(*k*)). Имеем *Ek*+1 = *Ek* + 2|*a*(*k*+1)| ‑ *E^k*.

Рассмотрим *S* ⊆ 1..*k*+1. Пусть |*S*| = 1, тогда *S* = {*i*}, *i* ∈ 1..*k*+1. Для *i* ∈ 1..*k* слагаемое 2|*сS*)| = 2|*a*(*i*)| один раз входит в сумму *Ek* со знаком «+», в результате для  *i* ∈ 1..*k*+1 слагаемое 2|*сS*)| = 2|*a*(*i*)| один раз входит в сумму *Ek*+1 с тем же знаком. Пусть |*S*| > 1. Если *k*+1 ∉ *S*, то слагаемое 2|*c*(*S*)| один раз входит в сумму *Ek* со знаком «(‑1)|*S*|+1», в результате для  *i* ∈ 1..*k*+1 слагаемое 2|*c*(*S*)| один раз входит в сумму *Ek*+1 с тем же знаком. Если *k*+1 ∈ *S*, то *c*(*S*) = { *a*(*i*1) ∩ ... ∩ *a*(*i*|*S*|‑1) ∩ *a*(*k*+1) } = { ( *a*(*k*+1) ∩ *a*(*i*1) ) ∩ ... ∩ ( *a*(*k*+1) ∩ *a*(*i*|*S*|‑1) ) }. Поэтому слагаемое 2|*c*(*S*)| один раз входит в сумму *E^k* со знаком «(‑1)|*S*|», но, поскольку сумма *E^k* вычитается, слагаемое 2|*c*(*S*)| один раз входит в сумму *Ek*+1 со знаком «(‑1)|*S*|+1».

Аналогично сумма длин, увеличенных на 1, различных множеств, вложенных хотя бы в одно из множеств *a*(*s*), где *s* ∈ 1..*k*+1, равно сумме длин, увеличенных на 1, различных множеств, вложенных хотя бы в одно из множеств *a*(*s*), где *s* ∈ 1..*k*, т.е. *F*(*a*(1), ..., *a*(*k*)), плюс сумма длин, увеличенных на 1, различных множеств, вложенных в *a*(*k*+1), т.е. (|*a*(*k*+1)| + 2)2|*a*(*k*+1)|‑1, кроме тех, что вложены в пересечение *a*(*k*+1) с объединением множеств *a*(1), ..., *a*(*k*), сумма длин которых, увеличенных на 1, равна *F*(*a*(*k*+1) ∩ *a*(1), ..., *a*(*k*+1) ∩ *a*(*k*)). Обозначим *Fk*+1 = *F*(*a*(1), ..., *a*(*k*+1)), *Fk* = *F*(*a*(1), ..., *a*(*k*)), *F^k* = *F*(*a*(*k*+1) ∩ *a*(1), ..., *a*(*k*+1) ∩ *a*(*k*)). Имеем *Fk*+1 = *Fk* + (|*a*(*k*+1)| + 2)2|*a*(*k*+1)|‑1 ‑ *F^k*. Рассмотрим *S* ⊆ 1..*k*+1. Пусть |*S*| = 1, тогда *S* = {*i*}, *i* ∈ 1..*k*+1. Для *i* ∈ 1..*k* слагаемое (|*c*(*S*)| + 2)2|*c*(*S*)|‑1 = (|*a*(*i*)| + 2)2|*a*(*i*)|‑1 один раз входит в сумму *Fk* со знаком «+», в результате для  *i* ∈ 1..*k*+1 слагаемое (|*c*(*S*)| + 2)2|*c*(*S*)|‑1 = (|*a*(*i*)| + 2)2|*a*(*i*)|‑1 один раз входит в сумму *Fk*+1 с тем же знаком. Пусть |*S*| > 1. Если *k*+1 ∉ *S*, то слагаемое (|*c*(*S*)| + 2)2|*c*(*S*)|‑1 один раз входит в сумму *Fk* со знаком «(‑1)|*S*|+1», в результате для  *i* ∈ 1..*k*+1 слагаемое (|*c*(*S*)| + 2)2|*c*(*S*)|‑1 один раз входит в сумму *Fk*+1 с тем же знаком. Если *k*+1 ∈ *S*, то слагаемое (|*c*(*S*)| + 2)2|*c*(*S*)|‑1 один раз входит в сумму *F^k* со знаком «(‑1)|*S*|», но, поскольку сумма *F^k* вычитается, слагаемое (|*c*(*S*)| + 2)2|*c*(*S*)|‑1 один раз входит в сумму *Fk*+1 со знаком «(‑1)|*S*|+1».

Утверждение доказано.

Число (не обязательно различных) множеств *c*(*S*) равно числу непустых подмножеств *S* множества 1..*k*, которое равно 2*k* ‑ 1, поэтому сложность вычисления *E* равна ***O***(2*k*) при условии, что за время ***O***(1) могут выполняться операции пересечения двух множеств (а также арифметические операции над целыми числами).

Обозначим множество индексов последовательности *x*, по которым находится символ *h*: *Kx*(*h*) = { *i* ∈ 1..*m* : *x*(*i*) = *h* }. Для *i* ∈ *Kx*(*h*) обозначим множество пар (символ в *x*, позиция символа в *x* относительно позиции *i*), кроме пары (*h*, 0): *Px*(*i*) = { (*x*(*t*), *t* – *i*) : *t* ∈ 1..*m* & *t* ≠ *i* }. Пусть символ *h* входит |*Kx*(*h*)| > 0 раз в *x* и |*Ky*(*h*)|> 0 раз в *y*. Обозначим *k* = |*Kx*(*h*)|\*|*Ky*(*h*)|. Для *i* ∈ *Kx*(*h*) и *j* ∈ *Ky*(*h*) для краткости обозначим *P*(*i*, *j*) = *Px*(*i*) ∩*Py*(*j*). Для *S* ⊆ *Kx*(*h*) × *Ky*(*h*), *S* ≠ ∅ обозначим *c*(*S*) = ∩{ *P*(*i*, *j*) : (*i*, *j*) ∈ *S* }.

1. Число различных общих *O*‑вложений в *x* и *y*, содержащих символ *h*, равно чередующейся сумме *E* = *E*(1) ‑ *E*(2) + *E*(3) ‑ *E*(4) ... (‑1)*k*+1*E*(*k*), где слагаемое *E*(*l*) это сумма числа всех подмножеств множества *c*(*S*) по всем *S* размера *l*, |*S*| = *l*: *E*(*l*) = Σ{ 2|*c*(*S*)| : *S* ⊆ 1..*k* & |*S*| = *l* }. Сложность вычисления равна ***O***(*n*2*k*).

Доказательство.

Рассмотрим общее *O*‑вложение *u* и пару его *E*‑вложений *v*(*x*) ∈ *o*(*u*, *x*) и *v*(*y*) ∈ *o*(*u*, *y*) таких, что *v*(*x*)*i* = *xi* = *h* и *v*(*y*)*j* = *yj* = *h*. Этому взаимно-однозначно соответствует подмножество множества пар *P*(*i*, *j*). Нам нужно вычислить число различных множеств, вложенных хотя бы в одно из множеств *P*(*i*, *j*), где *i* ∈ *Kx*(*h*) и *j* ∈ *Ky*(*h*). Число таких множеств пар *P*(*i*, *j*) равно *k*. По лемме 2 оно равно *E* = *E*(1) ‑ *E*(2) + *E*(3) ‑ *E*(4) ... (‑1)*k*+1*E*(*k*), где *E*(*l*) = Σ{ 2|*c*(*S*)| : *S* ⊆ 1..*k* & |*S*| = *l* }, *l* = 1..*k*, сумма числа всех подмножеств множества *c*(*S*) по всем *S* ⊆ *Kx*(*h*) × *Ky*(*h*) размера *l*, т.е. |*S*| = *l*, а *c*(*S*) = ∩{ *P*(*i*, *j*) : (*i*, *j*) ∈ *S* }.

Просматриваем последовательность *x* длиной *m*, вычисляем *Kx*(*h*). Просматривая *Kx*(*h*), вычисляем множества *Px*(*i*). Вычисление множества *Px*(*i*) требует просмотра последовательности *x* длиной *m*. Тем самым, все множества *Px*(*i*), *i* ∈ *Kx*(*h*), вычисляются за ***O***(*m*|*Kx*(*h*)|). Аналогично все множества *Py*(*j*), *j* ∈ *Ky*(*h*), вычисляются за ***O***(*n*|*Kx*(*h*)|). Можно считать, что множество *Px*(*i*) = { (*x*(*t*), *t* ‑ *i*) : *t* ∈ 1..*m* & *t* ≠ *i* } линейно упорядочено по возрастанию относительного индекса *t* – *i*; размер этого множества равен *m* – 1. Аналогично для множества *Py*(*j*); размер этого множества равен *n* – 1. Тогда для построения пересечения *P*(*i*, *j*) = *Px*(*i*) ∩*Py*(*j*) требуется просмотр этих множеств, т.е. время ***O***(*n* + *m*), а все множества *P*(*i*, *j*), *i* ∈ *Kx*(*h*) и *j* ∈ *Ky*(*h*), вычисляются за время ***O***(*k*(*n* + *m*)). Аналогично каждое пересечение множеств *P*(*i*, *j*) строится за время ***O***(*n* + *m*). Сложность вычисления суммы *E* по лемме 2 равна ***O***(2*k*), но при условии, что пересечение двух множеств строится за время ***O***(1), поэтому в данном случае потребуется время ***O***((*n* + *m*)2*k*). Общая сложность равна ***O***(*m*) + ***O***(*n*) + ***O***(*m*|*Kx*(*h*)|) + ***O***(*n*|*Ky*(*h*)|) + ***O***(*k*(*n* + *m*)) + ***O***((*n* + *m*)2*k*) = ***O***((*n* + *m*)2*k*), что для *m* ≤ *n* равно ***O***(*n*2*k*).

Теорема 17 определяет следующий алгоритм вычисления *O*0(*x*, *y*):

1. SUMMA = 0.
2. Просматривая последовательность *x*, ищем непустой символ *h*, входящий в *x*.
	1. Если не нашли, то конец алгоритма.
	2. Если нашли, то ищем символ *h* в *y*.
		1. Если не нашли, то в *x* заменяем *h* на пустой символ и переходим на п. 2.
		2. Если нашли, то:
			1. Вычисляем множество *Kx*(*h*) индексов в *x*, по которым находится *h*, и множество *Ky*(*h*) индексов в *y*, по которым находится *h*.
			2. Для каждой пары (*i*, *j*) ∈ *Kx*(*h*) × *Ky*(*h*) строим множество пар *P*(*i*, *j*).
			3. Для каждого множества пар индексов *S* ⊆ *Kx*(*h*) × *Ky*(*h*), *S* ≠ ∅ строим пересечение *c*(*S*) = ∩{ *P*(*i*, *j*) : (*i*, *j*) ∈ *S* }.
			4. Вычисляем *E* = *E*(1) ‑ *E*(2) + *E*(3) ‑ *E*(4) ... (‑1)*k*+1*E*(*k*).
			5. SUMMA = SUMMA + *E*.
			6. В *x* и в *y* заменяем *h* на пустой символ и переходим на п. 2.
	3. **Сумма μ‑длин общих *O*‑вложений**
3. Сумма μ‑длин различных общих *O*‑вложений в *x* и *y*, содержащих символ *h*, равно чередующейся сумме *F* = *F*(1) ‑ *F*(2) + *F*(3) ‑ *F*(4) ... (‑1)*k*+1*F*(*k*), где *F*(*l*) = Σ{ (|*c*(*S*)| + 2) 2|*c*(*S*)|‑1 : *S* ⊆ 1..*k* & |*S*| = *l* } сумма мощностей всех подмножеств всех множеств *c*(*S*) для |*S*| = *l*. Сложность вычисления ***O***(*n*2*k*).

Доказательство аналогично доказательству теоремы 17.

Теорема 18 определяет следующий алгоритм вычисления *O*1(*x*, *y*):

1. Просматривая последовательность *x*, ищем непустой символ *h*, входящий в *x*.
	1. Если не нашли, то конец алгоритма.
	2. Если нашли, то ищем символ *h* в *y*.
		1. Если не нашли, то в *x* заменяем *h* на пустой символ и переходим на п. 2.
		2. Если нашли, то:
			1. Вычисляем множество *Kx*(*h*) индексов в *x*, по которым находится *h*, и множество *Ky*(*h*) индексов в *y*, по которым находится *h*.
			2. Для каждой пары (*i*, *j*) ∈ *Kx*(*h*) × *Ky*(*h*) строим множество пар *P*(*i*, *j*).
			3. Для каждого множества пар индексов *S* ⊆ *Kx*(*h*) × *Ky*(*h*), *S* ≠ ∅ строим пересечение *c*(*S*) = ∩{ *P*(*i*, *j*) : (*i*, *j*) ∈ *S* }.
			4. Вычисляем *F* = *F*(1) ‑ *F*(2) + *F*(3) ‑ *F*(4) ... (‑1)*k*+1*F*(*k*).
			5. SUMMA = SUMMA + *F*.
			6. В *x* и в *y* заменяем *h* на пустой символ и переходим на п. 2.
	3. **Cумма минимумов чисел *E‑*вложений общих *O‑*вложений**
2. Cумма минимумов чисел *E‑*вложений общих *O‑*вложений в *x* и *y*, содержащих символ *h*, равна Σ{ *min*{|*I*|, |*J*|} \* 2|*A*(*I*,*J*)| ‑ 1 : *I* ⊆ *Kx*(*h*) & *I* ≠ ∅ & *J* ⊆ *Ky*(*h*) & *J* ≠ ∅ }, где *A*(*I*, *J*) = ( ∩{ *P*(*i*, *j*) : *i* ∈ *I*, *j* ∈ *J* } ) \ ( ∪{ *P*(*i*, *j*): *i* ∈ *Kx*(*h*) \ *I* ∨ *j* ∈ *Ky*(*h*) \ *J* } ).

Сложность вычисления равна ***O***(*n*2*k*).

Доказательство. Множество *A*(*I*, *J*) представляет все общие *O*‑вложения, содержащие символ *h*, которые имеют *E*‑вложения в *x*, представляемые множеством *I*, и *E*‑вложения в *y*, представляемые множеством *J*. Для каждого из таких *O*‑вложений минимум чисел их *E‑*вложений в *x* и в *y* равен *min*{|*I*|, |*J*|}. Число таких *O*‑вложений равно 2|*A*(*I*,*J*)|, поэтому сумма минимумов чисел *E‑*вложений таких общих *O‑*вложений равна *min*{|*I*|, |*J*|} \* 2|*A*(*I*,*J*)|. Суммируя по всем парам множеств *I* ⊆ *Kx*(*h*), *I* ≠ ∅ и *J* ⊆ *Ky*(*h*) , *J* ≠ ∅, получаем искомую сумму минимумов чисел *E‑*вложений общих *O‑*вложений в *x* и *y*, содержащих символ *h*.

При вычислении этой суммы операции сложения, умножения, возведения в степень числа 2 и вычисление минимума из двух чисел выполняются ***O***(2*k*) раз. Для вычисления всех *A*(*I*, *J*) операции вычисления разности двух множеств, пересечения двух множеств, объединения двух множеств выполняются ***O***(2*k*) раз, а каждая такая операция выполняется за время ***O***(*n* + *m*).

Просматриваем последовательность *x* длиной *m*, вычисляем *Kx*(*h*). Просматривая *Ky*(*h*), вычисляем множества *Px*(*i*). Вычисление множества *Px*(*i*) требует просмотра последовательности *x* длиной *m*. Тем самым, все множества *Px*(*i*), *i* ∈ *Kx*(*h*), вычисляются за ***O***(*m*|*Kx*(*h*)|). Аналогично все множества *Py*(*j*), *j* ∈ *Ky*(*h*), вычисляются за ***O***(*n*|*Ky*(*h*)|). Можно считать, что множество *Px*(*i*) = { (*x*(*t*), *t* ‑ *i*) : *t* ∈ 1..*m* & *t* ≠ *i* } линейно упорядочено по возрастанию относительного индекса *t* – *i*; размер этого множества равен *m* – 1. Аналогично для множества *Py*(*j*); размер этого множества равен *n* – 1. Тогда для построения пересечения *P*(*i*, *j*) = *Px*(*i*) ∩*Py*(*j*) требуется просмотр этих множеств, т.е. время ***O***(*n* + *m*), а все множества *P*(*i*, *j*), *i* ∈ *Kx*(*h*) и *j* ∈ *Ky*(*h*), вычисляются за время ***O***(*k*(*n* + *m*)). Для вычисления множества *A*(*I*, *J*) операции вычисления разности двух множеств, пересечения двух множеств, объединения двух множеств выполняются за время ***O***(*n* + *m*), а число таких множеств *A*(*I*, *J*) равно ***O***(2*k*). В результате все множества *A*(*I*, *J*) строятся за время ***O***((*n* + *m*)2*k*), после чего для вычисления суммы арифметические операции выполняются ***O***(2*k*) раз. Общая сложность равна ***O***(*k*(*n* + *m*)) + ***O***((*n* + *m*)2*k*) + ***O***(2*k*) = ***O***((*n* + *m*)2*k*), что для *m* ≤ *n* равно ***O***(*n*2*k*).

Теорема 19 определяет следующий алгоритм вычисления *O*2(*x*, *y*):

1. Просматривая последовательность *x*, ищем непустой символ *h*, входящий в *x*.
	1. Если не нашли, то конец алгоритма.
	2. Если нашли, то ищем символ *h* в *y*.
		1. Если не нашли, то в *x* заменяем *h* на пустой символ и переходим на п. 2.
		2. Если нашли, то:
			1. Вычисляем множество *Kx*(*h*) индексов в *x*, по которым находится *h*, и множество *Ky*(*h*) индексов в *y*, по которым находится *h*.
			2. Для каждой пары (*i*, *j*) ∈ *Kx*(*h*) × *Ky*(*h*) строим множество пар *P*(*i*, *j*).
			3. Для каждых *I* ⊆ *Kx*(*h*), *I* ≠ ∅, *J* ⊆ *Ky*(*h*), *J* ≠ ∅ строим *A*(*I*, *J*).
			4. Вычисляем *M* = Σ{ *min*{|*I*|, |*J*|} \* 2|*A*(*I*,*J*)| ‑ 1 : *I* ⊆ *Kx*(*h*) & *I* ≠ ∅ & *J* ⊆ *Ky*(*h*) & *J* ≠ ∅ }.
			5. SUMMA = SUMMA + *M*.
			6. В *x* и в *y* заменяем *h* на пустой символ и переходим на п. 2.
	3. **Сумма произведений чисел *E‑*вложений общих *O‑*вложений**
2. *O*3(*x*, *y*) = *O*3(*x*[*m*‑1], *y*) + *O*3(*x*, *y*[*n*‑1]) ‑ *O*3(*x*[*m*‑1], *y*[*n*‑1]), если *xm* ≠ *yn*;

*O*3(*x*, *y*) = *O*3(*x*[*m*‑1], *y*) + *O*3(*x*, *y*[*n*‑1]) ‑ *O*3(*x*[*m*‑1], *y*[*n*‑1]) + *L*3(*x*[*m*‑1], *y*[*n*‑1]), если *xm* = *yn*.

Доказательство. Обозначим следующие множества пар *E‑*вложений:

*E* = *O*(*x*, *y*),

*L*00 = *L*(*x*[*m*‑1], *y*[*n*‑1]),

*E*00 = *O*(*x*[*m*‑1], *y*[*n*‑1])⋅(ε, ε) — пара последних элементов (ε, ε),

*E*01 = ( *O*(*x*[*m*‑1], *y*])⋅(ε, ()) ) \ *E*00 — пара последних элементов (ε, *yn*),

*E*10 = ( *O*(*x*, *y*[*n*‑1])⋅((), ε) ) \ *E*00 — пара последних элементов (*xm*, ε),

*E*11 = *E* \ (*E*00 ∪ *E*01 ∪ *E*10) — пара последних элементов (*xm*, *yn*), поскольку *E*00 ∪ *E*01 ∪ *E*10 содержат все пары *E‑*вложений общих *O‑*вложений, в которых пара последних элементов содержит ε.

Очевидно, *E* = *E*00 ∪ *E*01 ∪ *E*10 ∪ *E*11. Пары последних элементов пар *E‑*вложений из разных множеств *E*00, *E*01, *E*10, *E*11 разные, поэтому эти множества попарно не пересекаются. Поэтому |*E*| = |*E*00| + |*E*01| + |*E*10| + |*E*11|, |*E*00 ∪ *E*01| = |*E*00| + |*E*01|, |*E*00 ∪ *E*10| = |*E*00| + |*E*10|.

В этих обозначениях утверждение теоремы имеет вид

|*E*| = (|*E*00| + |*E*01|) + (|*E*00| + |*E*10|) ‑ |*E*00| = |*E*00| + |*E*01| + |*E*10|, если *xm* ≠ *yn*,

|*E*| = (|*E*00| + |*E*01|) + (|*E*00| + |*E*10|) ‑ |*E*00| + |*L*00| = |*E*00| + |*E*01| + |*E*10| + |*L*00|, если *xm* = *yn*.

Рассмотрим случай *xm* ≠ *yn*. В этом случае *E*11 = ∅, что влечёт |*E*11| = 0 и |*E*00| = |*E*00| + |*E*01| + |*E*10|, что и требовалось доказать в этом случае.

Рассмотрим случай *xm* = *yn* = *h*. Каждая пара *E‑*вложений из *E*11 имеет вид (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*), где (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*) ∈ *L*00. Будем говорить, что пара (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*⋅*h*) соответствует паре (ε*m*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*, ε*n*‑|*v*|‑*k*‑1⋅*v*⋅ε*k*). Это соответствие, очевидно, является биекцией множеств *E*11 и *L*00. Тем самым, |*E*11| = |*L*00|. Поэтому |*E*| = |*E*00| + |*E*01| + |*E*10| + |*E*11| = |*E*00| + |*E*01| + |*E*10| + |*L*00|, что и требовалось доказать в этом случае.

Теорема 20 определяет алгоритм вычисления *O*3(*x*, *y*). Число шагов алгоритма равно ***O***(*mn*), что определяется числом функций вида *O*3(*x*[*m* ‑ *i*], *y*[*n* ‑ *j*]) и *L*3(*x*[*m ‑ i*], *y*[*n ‑ i*]), где *i* ∈ 0..*m* и *j* ∈ 0..*n*, при условии, что каждая функция вычисляется не более одного раза (после чего её значение сохраняется). На каждом шаге вычисления имеют сложность ***O***(1). Тем самым сложность алгоритма равна ***O***(*mn*).

* 1. **Функция похожести на основе наибольшей длины общего *O*‑вложения**
1.

*lcO*(*x*, *y*) = *max*{ *lcL*(*x*[*m* ‑1], *y*[*n* ‑1]) + 1, *lcO*(*x*[*m* ‑1], *y*[*n* ‑1]) }, если *xm* = *yn*:

*lcO*(*x*, *y*) = *max*{ *lcO*(*x*, *y*[*n* ‑1]), *lcO*(*x*[*m* ‑1], *y*) }, если *xm* ≠ *yn*.

Доказательство. Если *xm* = *yn*, то самые правые *E*‑вложения общего *O*‑вложения могут иметь в позиции *m* в *x* и в позиции *n* в *y* либо 1) символ *xm* = *yn*, либо 2) пустой символ. Если общее *O*‑вложение наибольшей длины относится к случаю 1, то оно имеет вид *u*⋅*xm*, где *u* общее *L*‑вложение префиксов *x*[*m* ‑1] и *y*[*n* ‑1], т.е. его μ‑длина на 1 больше наибольшей длины общего *L*‑вложения префиксов *x*[*m* ‑1] и *y*[*n* ‑1]. Если общее *O*‑вложение наибольшей длины относится к случаю 2, то оно является *O*‑вложением наибольшей длины префиксов *x*[*m* ‑1] и *y*[*n* ‑1] и имеет такую же μ‑длину. Отсюда следует утверждение теоремы для случая *xm* = *yn*.

Если *xm* ≠ *yn*, то самые правые *E*‑вложения общего *O*‑вложения имеют пустой символ либо 1) в позиции *m* в *x*, либо 2) в позиции *n* в *y*. Если общее *O*‑вложение наибольшей длины относится к случаю 1, то оно является *O*‑вложением наибольшей длины префикса *x*[*m* ‑1] и последовательности *y* и имеет такую же μ‑длину. Если общее *O*‑вложение наибольшей длины относится к случаю 2, то оно является *O*‑вложением наибольшей длины последовательности *x* и префикса *y*[*n* ‑1] и имеет такую же μ‑длину. Отсюда следует утверждение теоремы для случая *xm* ≠ *yn*.

Теорема 21 определяет алгоритм вычисления функции *O*4(*x*, *y*) = *lcO*(*x*, *y*) сложности ***O***(*mn*).

1. **Заключение**

Некоторые из введённых нами функций подобия играют вспомогательную роль. Например, *L*3 используется для вычисления *O*3, а *lcL* используется для вычисления *lcO*, *A*0 имеет как самостоятельное значение, так и используется для вычисления *A*1. Хотя могут быть приложения, в которых эти вспомогательные функции окажутся как раз основными. *L*‑вложения полезны там, где важно не только расстояние между символами вложения, но и расстояние от последнего символа вложения до конца последовательности, соответственно, *R*‑вложения полезны там, где важно расстояние от начала последовательности до первого символа вложения.

Сравнивая разные типы функций (независимо от типа вложения), можно отметить следующее. Число общих вложений (функция 0) является хорошей числовой характеристикой, но она не учитывает длины вложений. Например, последовательности 11112222 и 1122111 имеют 9 общих подпоследовательностей (включая пустую), а сумма их длин равна 17, в то же время первая последовательность имеет с последовательностью 22221111 тоже 9 общих подпоследовательностей, но сумма их длин равна 20 за счёт того, что есть две длинные подпоследовательности 1111 и 2222. Поэтому сумма длин общих вложений (функция 1) имеет самостоятельное значение.

Обе эти характеристики (функции 0 и 1) не учитывают тот факт, что одно общее вложение может входить в одну последовательность много раз, а в другую мало. Это пытается учесть сумма числа пар вхождений общих вложений (сумма произведений числа вхождений общих вложений — функций 3). Для того же примера: последовательности 11112222 и 1122111 имеют 279 пар вхождений общих вложений, а последовательности 11112222 и 22221111 — 139 пар за счёт того, что в первой паре последовательностей очень много вхождений в обе последовательности общих вложений 12, 112 и 122, отсутствующих для второй пары.

В то же время функция 3 не удовлетворяет естественной *аксиоме направленности (direction) сходства* [5], иначе называемой *ограниченность самоподобием (bounded by self-similarity)* [4]: *f*(𝑥, 𝑦) ≤ *min*{ *f*(𝑥, 𝑥), *f*(𝑦, 𝑦) }. В частности эта функция строго возрастает, когда одна и та же непустая последовательность *x* сравнивается с последовательностями *x*⋅*x*, *x*⋅*x*⋅*x*, *x*⋅*x*⋅*x*⋅*x*, ... Этот недостаток преодолевает функция 4 — сумма минимумов числа вхождений общих вложений. К сожалению, эта функция плохо поддаётся алгоритмической оптимизации, в частности, для *A*‑вложений, т.е. подпоследовательностей, мы не знаем алгоритма, отличного от полного перебора. Частичную оптимизацию можно было бы провести на основе эффективного алгоритма перечисления общих подпоследовательностей сложности *C*(*x*, *y*) меньшей, чем сложность полного перебора. Поскольку вычисление числа вхождений данного вложения *u* в данную последовательность *x* имеет сложность ***O***(|*u*|\*|*x*|) [3, лемма 8], мы имели бы алгоритм вычисления функции 4 сложности *C*(*x*, *y*)\****O***(*mn*). Также если бы был эффективный (возможно, на подклассе последовательностей) алгоритм перечисления подпоследовательностей только одной данной последовательности сложности *C*1(*x*, *y*), то мы имели бы алгоритм вычисления функции 4 сложности *C*1(*x*, *y*)\****O***(*mn*).

Функция 5, основанная на наибольшей длине общего вложения, также удовлетворяет аксиоме направленности. Но у неё тот недостаток, что не учитываются общие вложения, не являющиеся частью наибольшего (по длине) общего вложения. В нашем примере последовательности 11112222 и 1122111 имеют наибольшую общую подпоследовательность 1122, частью которой не является подпоследовательность 111, а последовательности 11112222 и 22221111 имеют две наибольшие общие подпоследовательности: 1111, не учитывающую все подпоследовательности из двоек, и 2222, не учитывающую все подпоследовательности из единиц.

Проблема перечисления общих вложений также исследуется. В качестве примера можно привести работу [6], в которой предлагается алгоритм перечисления максимальных по вложенности (а не по длине) общих подпоследовательностей, которому требуется полиномиальные время и память на каждую найденную максимальную общую подпоследовательность. Такие максимальные общие вложения обладают тем полезным свойством, что каждое общее вложение является частью одного или нескольких максимальных вложений. Можно было бы предложить функцию подобия, основанную на максимальных по вложенности общих вложениях: число таких вложений, сумма их длин и др. Это могло бы стать темой дальнейших исследований.

Другим направлением дальнейших исследований могли бы стать функции подобия последовательностей в алфавите, в котором символам приписаны те или иные веса. Пустому символу можно было бы приписать нулевой вес. Соответственно, вложению соответствует не число входящих в него символов, а сумма их весов. Отрицательный вес мог бы означать «штраф» за использование такого символа в общем вложении.

**Литература**

1. Wagner, Robert; Fischer, Michael (January 1974). "The string-to-string correction problem". Journal of the ACM. 21 (1): 168–173.

<https://dl.acm.org/doi/10.1145/321796.321811>

1. H. Wang, All common subsequences, in: M.M. Veloso (Ed.), IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 2007.

<https://www.aaai.org/Papers/IJCAI/2007/IJCAI07-101.pdf>

1. Cees Elzinga, Sven Rahmann, Hui Wang: Algorithms for Subsequence Combinatorics. Theoretical Computer Science 409(3): 394–404, 28 December 2008.

<http://dx.doi.org/10.1016/j.tcs.2008.08.035>

1. Gilbert Ritschard and Matthias Studer (editors). Proceedings of the International Conference on Sequence Analysis and Related Methods (LaCOSA II). Lausanne, Switzerland, June 8-10, 2016.

<https://www.academia.edu/83294569/Proceedings_of_the_International_Conference_on_Sequence_Analysis_and_Related_Methods_LaCOSA_II_Lausanne_Switzerland_June_8_10_2016>

1. С. В. Знаменский, Модель и аксиомы метрик сходства, Программные системы: теория и приложения, 2017, том 8, выпуск 4, 347–357.

DOI: <https://doi.org/10.25209/2079-3316-2017-8-4-347-357>

1. Conte, A., Grossi, R., Punzi, G. et al. Enumeration of Maximal Common Subsequences Between Two Strings. Algorithmica 84, 757–783 (2022). <https://doi.org/10.1007/s00453-021-00898-5>

[**ДВАДЦАТЬ ФУНКЦИЙ ПОДОБИЯ ДВУХ КОНЕЧНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ** 1](#_Toc125463893)

[**1.** **Введение** 1](#_Toc125463894)

[**2.** **Определения и обозначения** 2](#_Toc125463895)

[**3.** **Замена необщих символов пустым символом** 5](#_Toc125463896)

[**4.** ***A‑*вложения (подпоследовательности)** 6](#_Toc125463897)

[**4.1.** **Число общих *A‑*вложений** 6](#_Toc125463898)

[**4.2.** **Сумма длин общих *A‑*вложений** 7](#_Toc125463899)

[**4.3.** **Cумма минимумов чисел *E‑*вложений общих *A‑*вложений** 8](#_Toc125463900)

[**4.4.** **Сумма произведений чисел *E‑*вложений общих *A‑*вложений** 8](#_Toc125463901)

[**4.5.** **Функция похожести на основе наибольшей длины общего *A*‑вложения** 9](#_Toc125463902)

[**5.** ***L‑*вложения** 9](#_Toc125463903)

[**5.1.** **Число общих *L‑*вложений** 9](#_Toc125463904)

[**5.2.** **Сумма μ‑длин общих *L*‑вложений** 10](#_Toc125463905)

[**5.3.** **Cумма минимумов и сумма произведений чисел *E‑*вложений общих *L‑*вложений** 10](#_Toc125463906)

[**5.4.** **Функция похожести на основе наибольшей длины общего *L*‑вложения** 11](#_Toc125463907)

[**6.** ***R‑*вложения** 11](#_Toc125463908)

[**6.1.** **Число общих *R‑*вложений** 11](#_Toc125463909)

[**6.2.** **Сумма μ‑длин общих *R*‑вложений** 12](#_Toc125463910)

[**6.3.** **Cумма минимумов и сумма произведений чисел *E‑*вложений общих *R‑*вложений** 12](#_Toc125463911)

[**6.4.** **Функция похожести на основе наибольшей длины общего *R*‑вложения** 13](#_Toc125463912)

[**7.** ***O‑*вложения** 13](#_Toc125463913)

[**7.1.** **Число общих *O‑*вложений** 13](#_Toc125463914)

[**7.2.** **Сумма μ‑длин общих *O*‑вложений** 17](#_Toc125463915)

[**7.3.** **Cумма минимумов чисел *E‑*вложений общих *O‑*вложений** 17](#_Toc125463916)

[**7.4.** **Сумма произведений чисел *E‑*вложений общих *O‑*вложений** 18](#_Toc125463917)

[**7.5.** **Функция похожести на основе наибольшей длины общего *O*‑вложения** 19](#_Toc125463918)

[**8.** **Заключение** 20](#_Toc125463919)

[**Литература** 21](#_Toc125463920)