4. Пространственно-временной компромисс при разработке алгоритмов

« Значащее много никогда не должно находиться во власти значащего мало»

- Иоганн Вольфганг фон Гете.

Основная идея

Осуществляется полная или частичная обработка входных данных с сохранением полученной дополнительной информации для ускорения позднейшего решения поставленной задачи.

Основные подходы к решению задачи пространственно-временного компромисса

1. Улучшение входных данных
   (- метод сортировки подсчетом,
   - алгоритм Хорспула для поиска подстрок);
2. Предварительная структуризация
   (- хеширование,
   - индексирование при помощи B-деревьев);
3. Динамическое программирование.

4.1. Сортировка подсчетом

Основная идея: подсчитать для каждого элемента сортируемого списка общее количество элементов, меньших данного, и занести полученный результат в таблицу.

Полученные числа указывают позиции элементов в отсортированном списке.

Пример работы алгоритма сортировки подсчетом сравнений

Массив A[0..5]		62	31	84	96	19	47
Изначально	*Count*	0	0	0	0	0	0
После прохода							
i=0	*Count*	3	0	1	1	0	0
i=1	*Count*		1	2	2	0	1
i=2	*Count*			4	3	0	1
i=3	*Count*				5	0	1
i=4	*Count*					0	2
Конечный результат	*Count*	3	1	4	5	0	2
Массив S[0..5]		19	31	47	62	84	96

4.2. Улучшение входных данных в поиске подстрок

Задача поиска подстрок состоит в поиске данной подстроки из m символов, именуемой шаблон или образец, в более длинной строке из n символов, называемой текст.

Общее количество сравнений в наихудшем случае $C(n)=m*(n-m+1)$

Производительность алгоритма на основе «грубой силы» равна $O(n*m)$.

Для случайного естественного текста эффективность в среднем $O(n)$.

Алгоритм Хорспула

Пример. Поиск подстроки BARBER в некотором тексте:

S0 …. …. ***c*** …….Sn-1

BARBER

Алгоритм Хорспула определяет величину сдвига, рассматривая символ ***c*** текста, который при выравнивании находится напротив последнего символа образца.

Случай 1.

Если символа ***c*** в образце нет, то можно сдвинуть образец на всю его длину.

S_0 …. …. S …… S_{n-1}

$\neq$

B A R B E R

B A R B E R

Случай 2.

Если символ ***c*** в образце есть, но он не последний.

S_0 …. …. B …….S_{n-1}

≠

B A R B E R

B A R B E R

Сдвиг должен выровнять образец так, чтобы напротив ***c*** в тексте было первое справа вхождение символа в образец.

Случай 3.

Если символ ***c*** последний символ образца и среди остальных (m-1) символов образца такого символа нет, то сдвиг должен быть подобен сдвигу в случае 1, т.е. на величину m.

S_0 …. …. M E R …….S_{n-1}

$\neq = =$

L E A D E R

LEADER

Случай 4.

Если символ ***c*** последний символ образца и среди остальных (m-1) символов образца имеются вхождения этого символа, то сдвиг должен быть подобен сдвигу в случае 2.

S_0 …. …. O R ……S_{n-1}

$\neq =$

R E O R D E R

R E O R D E R

Величины сдвигов для символа ***c*** :

Длина образца m,

если ***c*** нет среди

первых m-1 символов образца.

t(c)=

Расстояние от крайнего справа символа ***c*** среди первых (m-1) символов образца до его последнего символа

Для образца B A R B E R все элементы таблицы равны 6 , а для символов :

E –1, B – 2, R – 3, A – 4.

Алгоритм ShiftTable (P[0..m-1])

// Заполнение таблицы сдвигов

// Вх. данные: Образец P[0..m-1] и алфавит возможных символов

// Вых. данные: таблица Table [0..size-1]

Инициализация всех элементов Table значениями m

for j←0 **to** m-2 **do**

    Table[P[j]] ←m-1-j

return Table

Алгоритм Хорспула

Шаг 1. Для данного образца длиной m и алфавита, используемого в тексте и образце, строится таблица сдвигов.

Шаг 2. Выравнивается начало образца с началом текста.

Шаг 3. До тех пор, пока не будет найдена искомая подстрока или пока образец не достигнет последнего символа текста, повторять следующие действия.

1. Начиная с последнего символа образца, сравниваем соответствующие символы в шаблоне и в тексте, пока не будет обнаружена пара разных символов.

2. Находим элемент t(c) из таблицы символов, где c − символ текста, находящийся напротив последнего символа образца , и сдвигаем образец вдоль текста на t(c) символов вправо.

Алгоритм Horspool (P [0..m-1], T [0..m-1])
// Вх. данные: Образец P [0..m-1], и текст T [0..m-1]
// Вых. данные: Индекс левого конца первой найденной подстроки или -1, если подстроки в тексте нет.

ShiftTable (P[0..m-1])
i←m-1
while i≤n-1 **do**
    k←0
    while k≤m-1 **and** P[m-1-k]=T[i-k] **do**
     k←k+1
    if k=m
     return i-m+1
    else i←i+ Table[T[i]]
return -1

Пример поиска подстроки BARBER в тексте из латинских букв и пробелов (_)

Таблица сдвигов

Символ c	A	B	C	D	E	F	…	R	…	Z	_
Сдвиг t(c)	4	2	6	6	1	6	6	3	6	6	6

```
J I M _ S A W _ M E _ I N _ A _ B A R B E R S H O P
BAR B E R                 BAR B E R
        B A R B E R         B A R B E R
          B A R B E R               B A R B E R
```