Летняя школа по компьютерным наукам – 2016

НИУ ВШЭ, Центр «Стратегия»

Параллель В, занятие 4: Хеши

Автор: Михаил Густокашин

Хеши

Хеш-функцией называется функция, ставящая в соответствие каждому элементу большого множества элемент маленького множества. Например, в качестве хеш-функции для числа можно использовать взятие числа по небольшому модулю.

С помощью хешей можно значительно ускорить операции сравнения сложных объектов. Например, можно подсчитать хеши от строк таким образом, чтобы значения хеша могли сравниваться за O(1). Для большинства строк хеши будут различны и сравнение будет происходить мгновенно. В случае совпадения хешей можно сравнить строки полностью.

Один из наиболее распространенных способов подсчета хеша от строки состоит в том, чтобы интерпретировать строку как число в некоторой системе счисления и брать его по модулю. При добавлении очередного символа (фактически, цифры), достаточно умножать накопленную часть строки на основание системы счисления и прибавлять код очередного символа.

Хеши на префиксе

Если запомнить хеши для каждого префикса (начала) строки, то можно научиться сравнивать любые подстроки за O(1). Рассмотрим пример с задачей подсчета Z-функции: для каждой позиции строки нужно определить максимальную длину подстроки, которая совпадает с префиксом строки. Найти максимальную длину подстроки можно с помощью бинарного поиска.

Полный код решения для задачи подсчета Z-функции приведен ниже:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

enum { magic = 10 }; // константа 10 очень удобна для отладки, реально нужно большое простое число

vector <int> make_hash(string &s, vector <int> &pows)
{
	int len = s.length() + 1;
	vector <int> hash(len);
	hash[0] = 0; // нумерацию символов в строке лучше начинать с единицы, чтобы не было особых случаев с обработкой подстроки-префикса
	for (int i = 1; i < len; i++)
		hash[i] = hash[i - 1] * magic + s[i - 1] - 'a' + 1; // вычитание кода символа можно убрать – это для удобства отладки. Здесь не делается подсчета по модулю – используется переполнение (так лучше не делать!)
```

```
	return hash;
}

vector <int> make_pows(int len, int magic) // подсчитываем все степени
{
	vector <int> pows(len);
	pows[0] = 1;
	for (int i = 1; i < len; i++)
		pows[i] = pows[i - 1] * magic;
	return pows;
}

bool iseq(vector <int> &hash, vector <int> &pows, int from, int len)
{
	return hash[from + len - 1] == hash[len] + hash[from - 1] * pows[len];
}

int bin_search(vector <int> &hash, vector <int> &pows, int from)
{
	int l = 0, r = hash.size() - from;
	while (l < r) {
		int m = (l + r + 1) / 2;
		if (iseq(hash, pows, from, m))
			l = m;
		else
			r = m - 1;
	}
	return r;
}

int main()
{
	ios::sync_with_stdio(false);
	int n;
	string s;
	vector <int> hash, pows;
	cin >> n;
	getline(cin, s);
	getline(cin, s);
	pows = make_pows(s.length() + 1, magic);
	hash = make_hash(s, pows);
	for (int i = 1; i <= n; ++i)
		cout << bin_search(hash, pows, i) << ' ';
	return 0;

}
```

Посмотрим на примере, как происходит сравнение двух подстрок.

Например, для строки aabaa набор хешей будет выглядеть следующим образом:

h[0] = 0

h[1] = 1

h[2] = 11

h[3] = 112

h[4] = 1121

h[5] = 11211

Если, например, сравнить подстроку длины два, которая начинается с третьего символа (это подстрока ba) с префиксом строки той же длины, то выражение hash[from + len - 1] == hash[len] + hash[from - 1] * pows[len]

1121 = 11 + 1 * 10

Чтобы узнать хеш произвольной подстроки, начинающейся с позиции from и длины len, необходимо подсчитать следующую формулу:

hash[from + len – 1] – hash[from – 1] * pows[len]

Для уже рассмотренной подстроки длины два, которая начинается с третьего символа, вычисление по этой формуле даст результат

hash[3 + 2 – 1] – hash[3 – 1] * pows[2] = hash[4] – hash[2] * pows[2] = 1121 – 11 * 100 = 1121 – 1100 = 21

Однако, во избежание проблем связанных с взятием по модулю, следует избегать вычитания, поэтому при сравнении лучше перенести некоторые слагаемые таким образом, чтобы использовались только операции сложения.

Пользуясь формулой для подсчета хеша произвольной подстроки (и переносом слагаемых), можно научиться сравнивать две произвольные подстроки, а не только произвольную подстроку с префиксом соответствующей длины.

Двумерные хеши

По аналогии с одномерными хешами можно сделать двумерные хеши, которые позволят считать хеш от произвольного подпрямоугольника в таблице. Здесь нам потребуется аналогия с подсчетом частичных сумм в углах таблицы.

При вычислении хеша следует пользоваться формулой:

h[i, j] = h[i, j - 1] * p[j] + h[i - 1, j] * q[i] + s[i, j] - h[i - 1, j - 1] * p[j] * q[i]

Где p и q – две различные константы, а массивы – их предподсчитанные степени.

Сравнение двух подпрямоугольников осуществляется аналогично одномерному случаю. Необходимо сделать так, чтобы степени p и q совпадали.