Время работы алгоритма для тех или иных входных данных измеряется в количестве элементарных операций, или шагов, которые необходимо выполнить.

Время выполнения различных строк может различаться, но мы предположим, что одна и та же i-ая строка выполняется за время Vi, где Vi –временная константа. Введем для процедуры kernel – **метод решеток** – временную стоимость каждого блока кода и количество повторений.

|  |  |  |
| --- | --- | --- |
| **Алгоритм** | **Стоимость** | **Повторы** |
| for (unsigned s = 0; s < count; s++) | **V1** | **a** |
| { |  |  |
| … |  |  |
| while (true) | **V2** | **a\*b** |
| { |  |  |
| … |  |  |
| if (it >= eArray.end()) |  |  |
| { |  |  |
| for (size\_t i = 0; i < n; i++) m[i] <<= 1u; | **V3** | **a\*b\*с** |
| continue; |  |  |
| } |  |  |
| … |  |  |
| while ((it = thrust::find(it, eArray.end(), true)) < eArray.end()) | **V4** | **a\*b\*d** |
| { |  |  |
| … |  |  |
| } |  |  |
| … |  |  |
| for (size\_t k = 0; k < n; k++) | **V5** | **a\*b\*с** |
| { |  |  |
| … |  |  |
| } |  |  |
| } |  |  |
| } |  |  |

Таким образом, время выполнения всего алгоритма – это суммарное время выполнения всех блоков кода, входящих в алгоритм. Для нахождения времени выполнения алгоритма решеток необходимо просуммировать всё полученное время, помноженное на число повторов данного участка кода. Получим следующую формулу:

Формулу можно представить в виде:

Введем для процедуры kernel – **фронтальный спуск** – временную стоимость каждого блока кода и количество повторений.

|  |  |  |
| --- | --- | --- |
| **Алгоритм** | **Стоимость** | **Повторы** |
| for (unsigned s = 0; s < count; s++) | **V1** | **a** |
| { |  |  |
| while (true) | **V2** | **a\*b** |
| { |  |  |
| auto it = thrust::find(eArray.begin(), eArray.end(), true); |  |  |
| if (it >= eArray.end()) |  |  |
| { |  |  |
| for (size\_t i = 0; i < n; i++) m[i] <<= 1u; | **V3** | **a\*b\*c** |
| continue; |  |  |
| } |  |  |
| while ((it = thrust::find(it, eArray.end(), true)) < eArray.end()) | **V4** | **a\*b\*d** |
| { |  |  |
| } |  |  |
| } |  |  |
| while (true) | **V5** | **a\*b** |
| { |  |  |
| for (size\_t k = 0; k < n; k++) | **V6** | **a\*b\*v** |
| { |  |  |
| while (true) | **V7** | **a\*b\*e** |
| { |  |  |
| while ((it = thrust::find(it, eArray.end(), true)) < eArray.end()) | **V8** | **a\*b\*e\*f** |
| { |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |

Введем для процедуры kernel – **метод Хука-Дживса** – временную стоимость каждого блока кода и количество повторений.

|  |  |  |
| --- | --- | --- |
| **Алгоритм** | **Стоимость** | **Повторы** |
| for (unsigned s = 0; s < count; s++) | **V1** | **a** |
| { |  |  |
| while (true) | **V2** | **a\*b** |
| { |  |  |
| if (it >= eArray.end()) |  |  |
| { |  |  |
| for (size\_t i = 0; i < n; i++) m[i] <<= 1u; | **V3** | **a\*b\*c** |
| continue; |  |  |
| } |  |  |
| while ((it = thrust::find(it, eArray.end(), true)) < eArray.end()) | **V4** | **a\*b\*d** |
| { |  |  |
| } |  |  |
| } |  |  |
| while (true) | **V5** | **a\*b** |
| { |  |  |
| for (size\_t k = 0; k < n; k++) | **V6** | **a\*b\*d** |
| { |  |  |
| while (true) | **V7** | **a\*b\*f** |
| { |  |  |
| while ((it = thrust::find(it, eArray.end(), true)) < eArray.end()) | **V8** | **a\*b\*f\*e** |
| { |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |