Алгоритм шинглов: визуализация

Случай сравнения двух программ. Очевидным образом обобщается на N исходных кодов.
Достаточно малый размер кода выбран для более детального описания всех этапов.
Два кода различны лишь именем переменной *source/arr* и вынесением арифметического преобразования в функцию *midline* в первом коде.

Code #1	Code #2

`for (int i = 1; i < n; i++) {`
`  x = source[i].x, y = source[i].y;`
`  S += midline(yl, y) * (x - xl);`
`}`
	`for (int i = 1; i < n; i++) {`
	`  x = arr[i].x, y = arr[i].y;`
	`  S += (yl + y) / 2 * (x - xl);`
	`}`

Исходный код проходит нормализацию и лексический анализ: удаляются пробелы и иные незначащие символы, выделяются лексемы (токены). На следующем шаге имеем дело с массивом токенов.

Count=49	18: "source"	34: "S"
1: "for"	19: "["	35: "+="
2: "("	20: "i"	36: "midline"
3: "int"	21: "]"	37: "("
4: "i"	22: "."	38: "yl"
5: "="	23: "x"	39: ","
6: "1"	24: ","	40: "y"
7: ";"	25: "y"	41: ")"
8: "i"	26: "="	42: "*"
9: "<"	27: "source"	43: "("
10: "n"	28: "["	44: "x"
11: ";"	29: "i"	45: "-"
12: "i"	30: "]"	46: "xl"
13: "++"	31: "."	47: ")"
14: ")"	32: "y"	48: ";"
15: "{"	33: ";"	49: "}"
16: "x"		
17: "="		

Count=50	18: "arr"	35: "+="
1: "for"	19: "["	36: "("
2: "("	20: "i"	37: "yl"
3: "int"	21: "]"	38: "+"
4: "i"	22: "."	39: "y"
5: "="	23: "x"	40: ")"
6: "1"	24: ","	41: "/"
7: ";"	25: "y"	42: "2"
8: "i"	26: "="	43: "*"
9: "*"	27: "arr"	44: "("
10: "n"	28: "["	45: "x"
11: ";"	29: "i"	46: "-"
12: "i"	30: "]"	47: "xl"
13: "++"	31: "."	48: ")"
14: ")"	32: "y"	49: ";"
15: "{"	33: ";"	50: "}"
16: "x"	34: "S"	
17: "="	35: "+="	

На данном этапе из массива лексем формируются ***шинглы*** -

последовательности из K лексем. Заметим, что таких шинглов будет в точности N - K + 1, где K - количество лексем в массиве.
Примем K = 20.

for(inti=1;i*n;i++){x=source[i	for(inti=1;i*n;i++){x=arr[i
(inti=1;i*n;i++){x=source[i]	(inti=1;i*n;i++){x=arr[i]
inti=1;i*n;i++){x=source[i].	inti=1;i*n;i++){x=arr[i].
i=1;i*n;i++){x=source[i].x	i=1;i*n;i++){x=arr[i].x
=1;i*n;i++){x=source[i].x,	=1;i*n;i++){x=arr[i].x,
1;i*n;i++){x=source[i].x,y	1;i*n;i++){x=arr[i].x,y
;i*n;i++){x=source[i].x,y=	;i*n;i++){x=arr[i].x,y=
i*n;i++){x=source[i].x,y=source	i*n;i++){x=arr[i].x,y=arr
*n;i++){x=source[i].x,y=source[	*n;i++){x=arr[i].x,y=arr[
n;i++){x=source[i].x,y=source[i	n;i++){x=arr[i].x,y=arr[i
;i++){x=source[i].x,y=source[i]	;i++){x=arr[i].x,y=arr[i]
i++){x=source[i].x,y=source[i].	i++){x=arr[i].x,y=arr[i].
++){x=source[i].x,y=source[i].y	++){x=arr[i].x,y=arr[i].y
){x=source[i].x,y=source[i].y;	){x=arr[i].x,y=arr[i].y;
{x=source[i].x,y=source[i].y;S	{x=arr[i].x,y=arr[i].y;S
x=source[i].x,y=source[i].y;S+=	x=arr[i].x,y=arr[i].y;S+=
=source[i].x,y=source[i].y;S+=midline	=arr[i].x,y=arr[i].y;S+=(
source[i].x,y=source[i].y;S+=midline(	arr[i].x,y=arr[i].y;S+=(yl
[i].x,y=source[i].y;S+=midline(yl	[i].x,y=arr[i].y;S+=(yl+
i].x,y=source[i].y;S+=midline(yl,	i].x,y=arr[i].y;S+=(yl+y
].x,y=source[i].y;S+=midline(yl,y	].x,y=arr[i].y;S+=(yl+y)
.x,y=source[i].y;S+=midline(yl,y)	.x,y=arr[i].y;S+=(yl+y)/
x,y=source[i].y;S+=midline(yl,y)*	x,y=arr[i].y;S+=(yl+y)/2
,y=source[i].y;S+=midline(yl,y)*(	,y=arr[i].y;S+=(yl+y)/2*
y=source[i].y;S+=midline(yl,y)*(x	y=arr[i].y;S+=(yl+y)/2*(
=source[i].y;S+=midline(yl,y)*(x-	=arr[i].y;S+=(yl+y)/2*(x

```
source[i].y;S+=midline(yl,y)*(x-xl
[i].y;S+=midline(yl,y)*(x-xl)
i].y;S+=midline(yl,y)*(x-xl);
].y;S+=midline(yl,y)*(x-xl);}
```

```
arr[i].y;S+=(yl+y)/2*(x-
[i].y;S+=(yl+y)/2*(x-xl
i].y;S+=(yl+y)/2*(x-xl)
].y;S+=(yl+y)/2*(x-xl);
.y;S+=(yl+y)/2*(x-xl);}
```

Все имена переменных необходимо заменить одним словом (в данном случае - *var*).
На самом деле данный этап с целью оптимизации стоит проводить до построения шинглов - в данном случае он вынесен сюда для наглядности предыдущего этапа.
Также имеет смысл заменить все ключевые слова, синтаксические символы, операторы и т.п. на определенные слова (например, последовательность из двух букв) для экономии места. В данном описании опустим это преобразование.

```
for(intvar=1;var<var;var++){var=var[var
(intvar=1;var<var;var++){var=var[var]
intvar=1;var<var;var++){var=var[var].
var=1;var<var;var++){var=var[var].var
=1;var<var;var++){var=var[var].var,
1;var<var;var++){var=var[var].var,var
;var<var;var++){var=var[var].var,var=
var<var;var++){var=var[var].var,var=var
<var;var++){var=var[var].var,var=var[
var;var++){var=var[var].var,var=var[var
;var++){var=var[var].var,var=var[var]
var++){var=var[var].var,var=var[var].
++){var=var[var].var,var=var[var].var
){var=var[var].var,var=var[var].var;
{var=var[var].var,var=var[var].var;var
```

```
for(intvar=1;var<var;var++){var=var[var
(intvar=1;var<var;var++){var=var[var]
intvar=1;var<var;var++){var=var[var].
var=1;var<var;var++){var=var[var].var
=1;var<var;var++){var=var[var].var,
1;var<var;var++){var=var[var].var,var
;var<var;var++){var=var[var].var,var=
var<var;var++){var=var[var].var,var=var
<var;var++){var=var[var].var,var=var[
var;var++){var=var[var].var,var=var[var
;var++){var=var[var].var,var=var[var]
var++){var=var[var].var,var=var[var].
++){var=var[var].var,var=var[var].var
){var=var[var].var,var=var[var].var;
{var=var[var].var,var=var[var].var;var
```

var=var[var].var,var=var[var].var;var+= =var[var].var,var=var[var].var;var+=var var[var].var,var=var[var].var;var+=var([var].var,var=var[var].var;var+=var(var var].var,var=var[var].var;var+=var(var,].var,var=var[var].var;var+=var(var,var .var,var=var[var].var;var+=var(var,var) var,var=var[var].var;var+=var(var,var)* ,var=var[var].var;var+=var(var,var)*(var=var[var].var;var+=var(var,var)*(var =var[var].var;var+=var(var,var)*(var- var[var].var;var+=var(var,var)*(var-var [var].var;var+=var(var,var)*(var-var) var].var;var+=var(var,var)*(var-var);].var;var+=var(var,var)*(var-var);}	var=var[var].var,var=var[var].var;var+= =var[var].var,var=var[var].var;var+=(var[var].var,var=var[var].var;var+=(var [var].var,var=var[var].var;var+=(var+ var].var,var=var[var].var;var+=(var+var].var,var=var[var].var;var+=(var+var) .var,var=var[var].var;var+=(var+var)/ var,var=var[var].var;var+=(var+var)/2 ,var=var[var].var;var+=(var+var)/2* var=var[var].var;var+=(var+var)/2*(=var[var].var;var+=(var+var)/2*(var var[var].var;var+=(var+var)/2*(var- [var].var;var+=(var+var)/2*(var-var var].var;var+=(var+var)/2*(var-var)].var;var+=(var+var)/2*(var-var); .var;var+=(var+var)/2*(var-var);}

Далее необходимо произвести хеширование получившихся шинглов. В настоящем описании применим хеширование с помощью алгоритма SHA-1.

59165ba2c8eb0e70a023a3ece030d147821865a4 d21d33d616c6870210cd55053c721d61f7e9ce2e 9bd309a446eb0f68f502220affe4fb3833f4338d a89d81fe7a5e917f1daee6e787fd5b80c9431c1e 18b62ba51c464b230e81c5388b255d33fa9a0000 1aae8f010aeac39a33d1f61fe4dcbc32ee1ee1e3 dee5757671428dee8207ea89c98b1c86cc342109 88daa4adb8ddaff01ff20012e644384078fed4df 68cddc4a76cb237043f2723d30339f691215a571 7433ed85fdc05490ba62758ad389a19797a0b969	59165ba2c8eb0e70a023a3ece030d147821865a4 d21d33d616c6870210cd55053c721d61f7e9ce2e 9bd309a446eb0f68f502220affe4fb3833f4338d a89d81fe7a5e917f1daee6e787fd5b80c9431c1e 18b62ba51c464b230e81c5388b255d33fa9a0000 1aae8f010aeac39a33d1f61fe4dcbc32ee1ee1e3 dee5757671428dee8207ea89c98b1c86cc342109 88daa4adb8ddaff01ff20012e644384078fed4df 68cddc4a76cb237043f2723d30339f691215a571 7433ed85fdc05490ba62758ad389a19797a0b969

ede17c9515bff0eaf2ad942fb96e3d09f2200d21	ede17c9515bff0eaf2ad942fb96e3d09f2200d21
5747a769a70e7c5f242957625d5a16a5706799cc	5747a769a70e7c5f242957625d5a16a5706799cc
380249fe0852645b722a728b0930b606f03045ce	380249fe0852645b722a728b0930b606f03045ce
988f029b39820ceee2d95316c68dc3d4b88f60bf	988f029b39820ceee2d95316c68dc3d4b88f60bf
da5fcb27c85688d810a62a60e544181a46f80d4f	da5fcb27c85688d810a62a60e544181a46f80d4f
a73534c18caf150d993db36e3ac03890627cd52e	a73534c18caf150d993db36e3ac03890627cd52e
f99484e0cb985fe83b7873c2bd952e8a7e113a10	f99484e0cb985fe83b7873c2bd952e8a7e113a10
2e6fb92f83662747de846143944f7567d10ea046	51244914b2477a71eb8202f1262336226db29304
699415092ae38c4a5c3a239237947e59479ed7be	f6325f1731f6dfb1cb24a76474097cf4cd367aec
b1e5d07a2e8300f7594fe75dab3e3baa88995187	d05c19a816c68c1a62118972fa0ff9d5181562f9
aab3f8186a5280372fba2cb78631d11dba91de36	ecfe96a72f2f941813ba4aee355e77d10a6b914f
3d78c81a7a07c9cca6a4276be3ad928d60681b40	a3a73b5e6d445c3ae3358760df2accfd5898add7
a47446275a89211064630a18e011bb33f54f44f1	396267a9565c01b5583ac73f347eaea8694372c3
ad587f41f9d229ad84e5d5ccd46c6be19a9acc71	342f0e380bad4ccba404d1fed8dc9fd6a2f6faec
065063699af68bd49b6b9b20b5d74d9e8a30f22b	0fad104949d89ea1c2c03f78f3f81abb01789670
0c7102f69a6cb32a0767a00ab2c27ed2b1cc2317	34569da45cbc0264aea0ed05dfeb885d5f2fccc1
b3d2ae7f2a95ea82d6c4ee47551688e2694d933b	fbefb3d19885dc18e7f205d5f1f86c15f8728021
b774fcb7bcf37f4d77258b98378a5dc95f791803	d4f6f318433282cfbbc975e4e77c25207d034c4d
7bb4582ed40d2e56367d38c1c7cec093a907d3d2	ec91c20ecfe1d74a4004c8d43c6b383cc729a5e3
ac7213b3afd8c00f1ae352e032a614fe59b2e7ee	f03787720050e761759a51c56d6d189920ade6aa
	ac1a58b67a3513c5cd7ae135c040dc31cd724469

Результат сравнения можно рассчитать по формуле $\frac{A \cap B}{A \cup B}$ *100%, где A - множество хешей первой программы, B - второй.

Из нашего примера получаем 17 совпадающих хешей, 44 хеша в объединении. Таким образом, процент совпадения равен 39%.
Таким образом, можно отметить несколько ключевых особенностей алгоритма шинглов:

1. Уязвимость перед вставкой “шума”, вынесением части кода в дополнительную функцию.
2. Независимость от расположения отдельных блоков кода (например, функций) в коде программы, от изменения идентификаторов.

3. Достоверность при проверке достаточно малых исходных кодов (В действительности разобранные в данном описании коды могли быть написаны независимо разными людьми, потому такой процент совпадения не может однозначно идентифицировать совпадение. Тем не менее, такой процент может показать проверяющему, что совпадения имеются, и более точный вердикт будет вынесен при зрительном изучении.)
4. Асимптотика T = **O**(N - K), M = **O**(N - K).