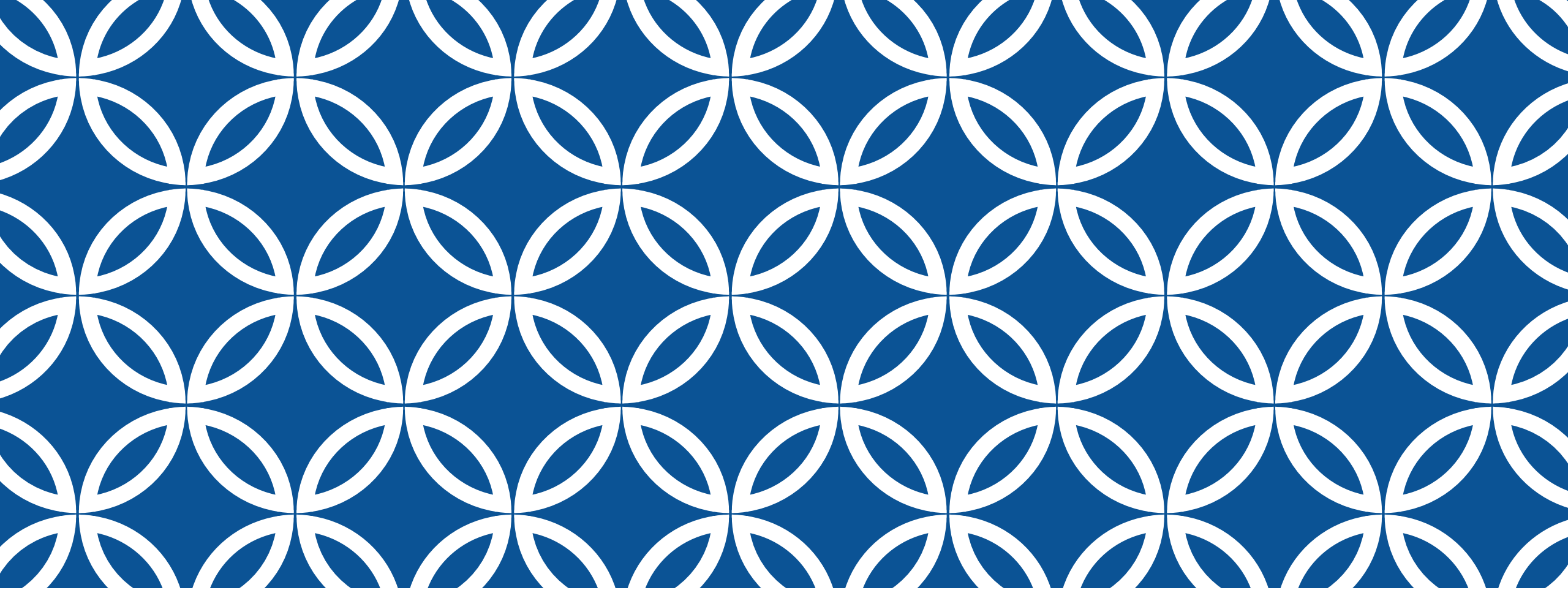


MAŠĪNKODA INSTRUMENTĒŠANA BLOKU IZPILDES STATISTISKAJAI ANALĪZEI UN PAREDZĒŠANAI

Eduards Vāvere

DARBA 1. UN 2. DAĻA

1. Konteksts
   1. Binārā analīze
      1. Kas ir binārā analīze?
      2. Kādi ir tās pielietojumi industrijā?
   2. Binārās analīzes rīki, process un soļi
2. Binārās analīzes efektivitāte un tās uzlabošana
   1. Kā salīdzināt efektivitāti?
   2. Esošās metodes
   3. Trūkumi esošajās metodēs

DARBA 3. DAĻA

1. Oriģināla metode binārās analīzes efektivitātes uzlabošanai
   1. Starp-bloku varbūtisko pāreju grafs – teorētiskais pamatojums
   2. Prakstiskā realizācija un ātrdarbības analīze
   3. Prakstisks pielietojums – maza programma un liela programma
   4. Integrācijas iespējamība esošos rīkos
   5. Ieguvumu un nepilnību kopsavilkums
2. Nākošo bloku paredzēšana un turpmāku pētijumu iespējas

TERMINI

Apgrieztā inženierija – Saražota produkta izpēte ar mērķi atgūt oriģinālo projektējumu vai tā elementus, no produkta kompozīcijas vai uzvedības.

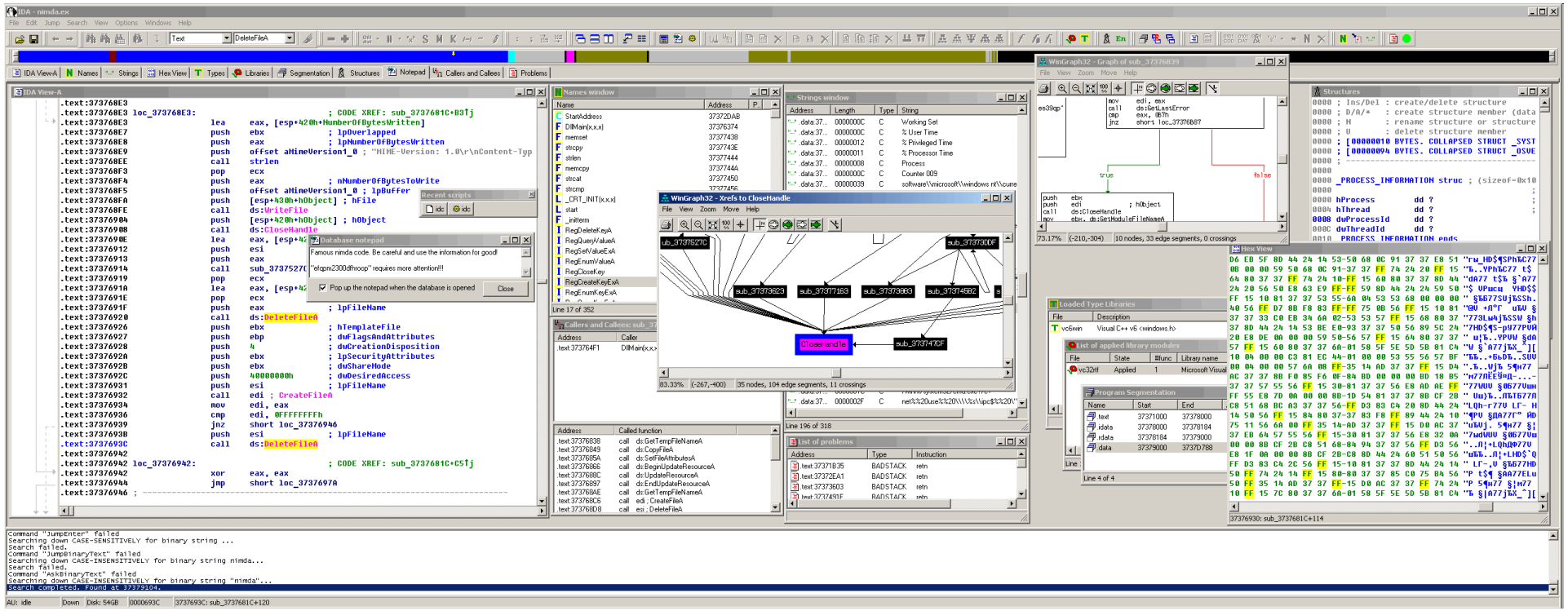
Binārā analīze – Apgrieztās inženierija virziens, kurš nodarbojas ar mašīnkoda izpēti.

Obfuscēt (Obfuskācija, obfuskātors) – Ar īpašu algoritmu palīdzību slēpt programmas projektējumu padarot mašīnkodu grūtāk analizējamu.

Deobfuscēt (**Deobfuskācija, deobfuskātors**) **–** Atgriezt atpakaļ oriģinālu vai labi zināmu obfuskātoru veiktās mašīnkoda transformācijas.

```
 .def ___main; .scl 2; .type 32; .endef
 .section .rdata,"dr"
LC0:
 .ascii "type a word \0"
LC1:
 .ascii "%c \0"
 .text
 .globl _main
 .def _main; .scl 2; .type 32; .endef
_main:
 pushl %ebp
 movl %esp, %ebp
 andl $-16, %esp
 addl $-128, %esp
 call ___main
 movb $97, 127(%esp)
 movl $0, 120(%esp)
 movl $LC0, (%esp)
 call _puts
 jmp L2
```

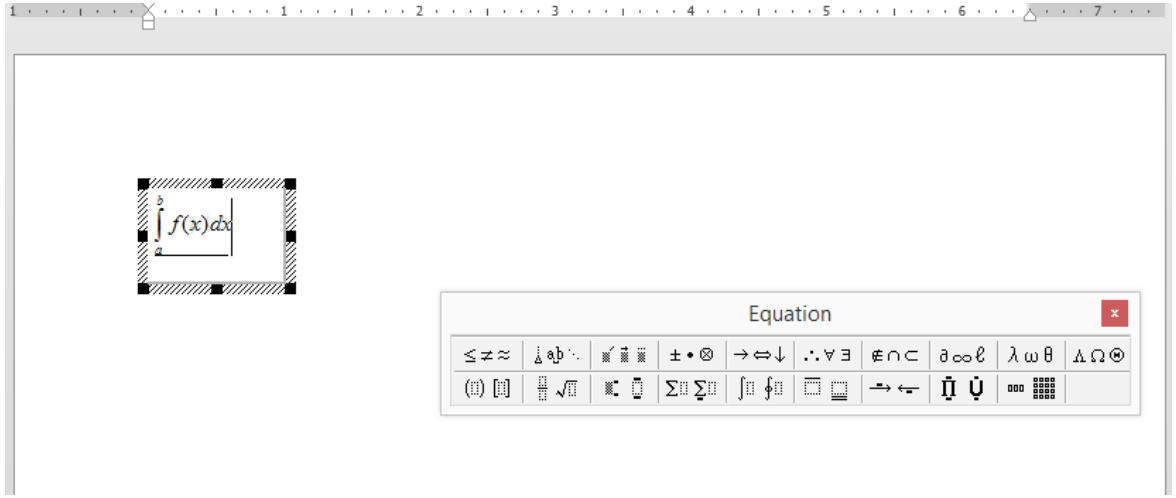
1.1. att. **Izvada fragments no C/C++ kompilatora GCC "disassembly" komandas [21]**



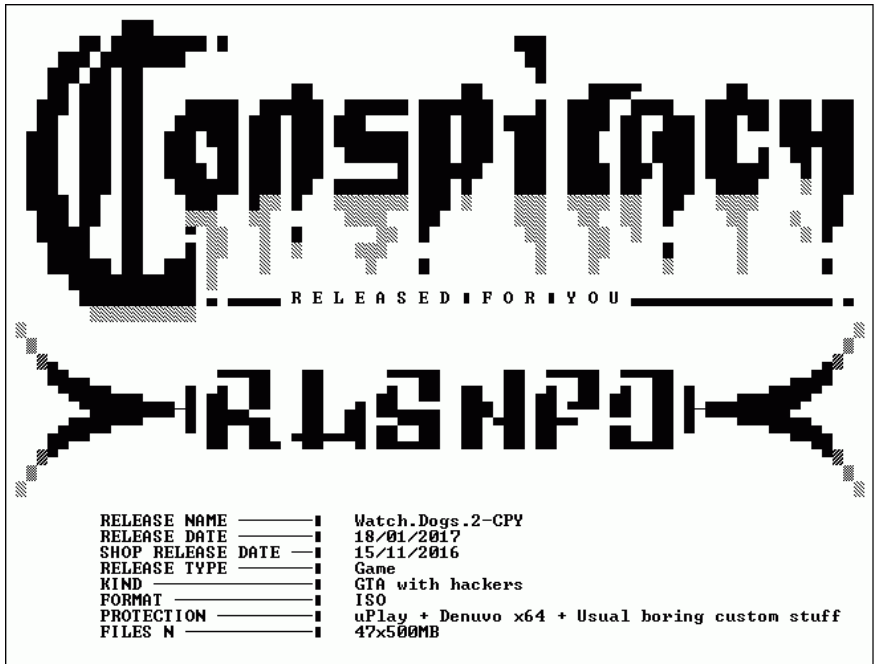
1.2. att. **Apgrieztās inženierijas rīks "IDA Pro" - moderns disasemblers [2]**

Binārās analīzes

PIELIETOJUMI



1.6. att. **Microsoft Office vienādojumu rīks, kuram tika pazaudēts pirmkods [13]**

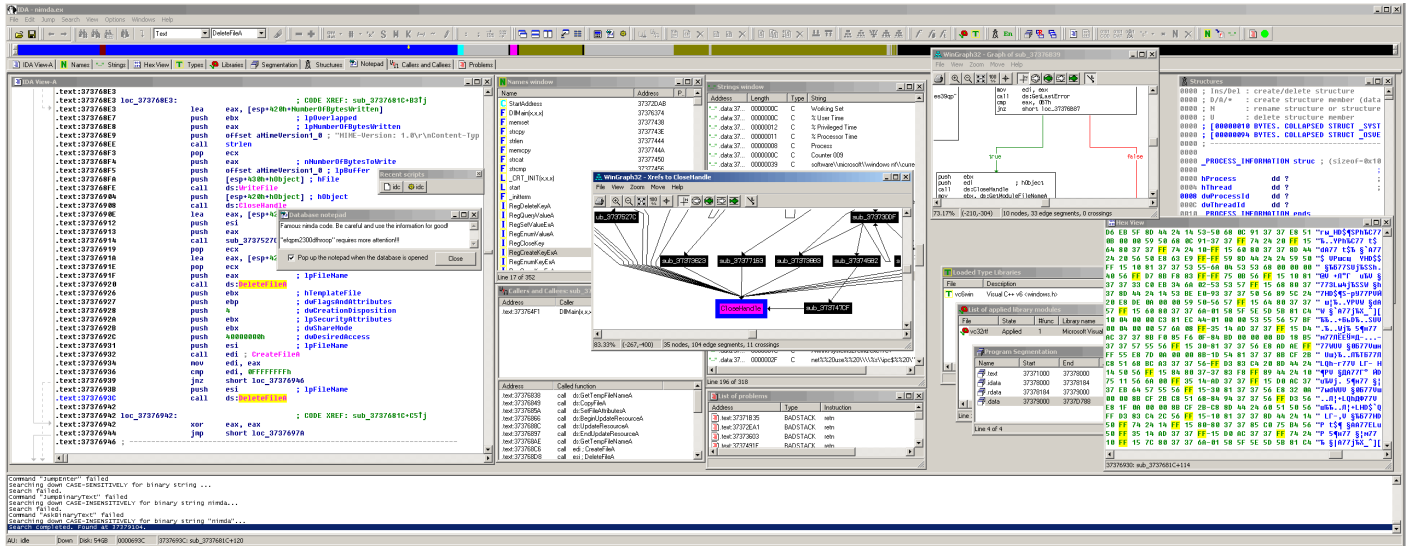


1.4. att. **Fragments no "CONSPIRACY" (CPY) laužu grupas paziņojuma publicējot kārtējo uzlauzto "Denuvo" spēli [16]**

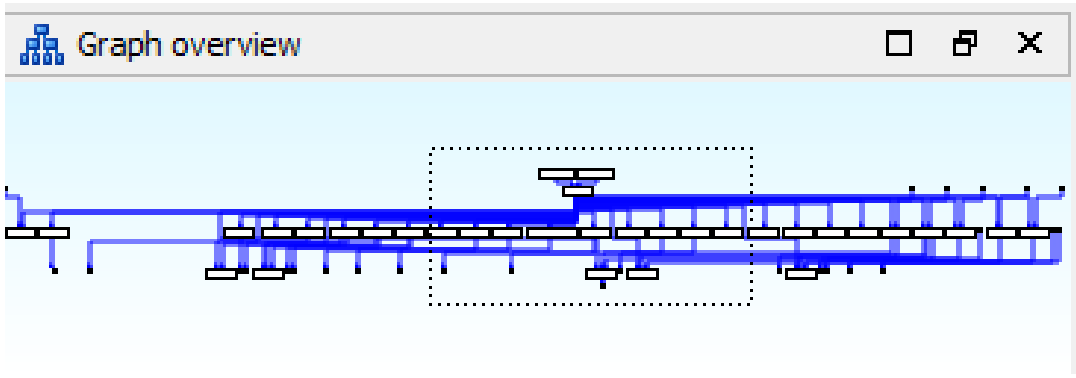
1.3. att. **"Denuvo" ir bieži izmantotā aizsardzība liela budžeta spēlēm [15]**

Binārās analīzes

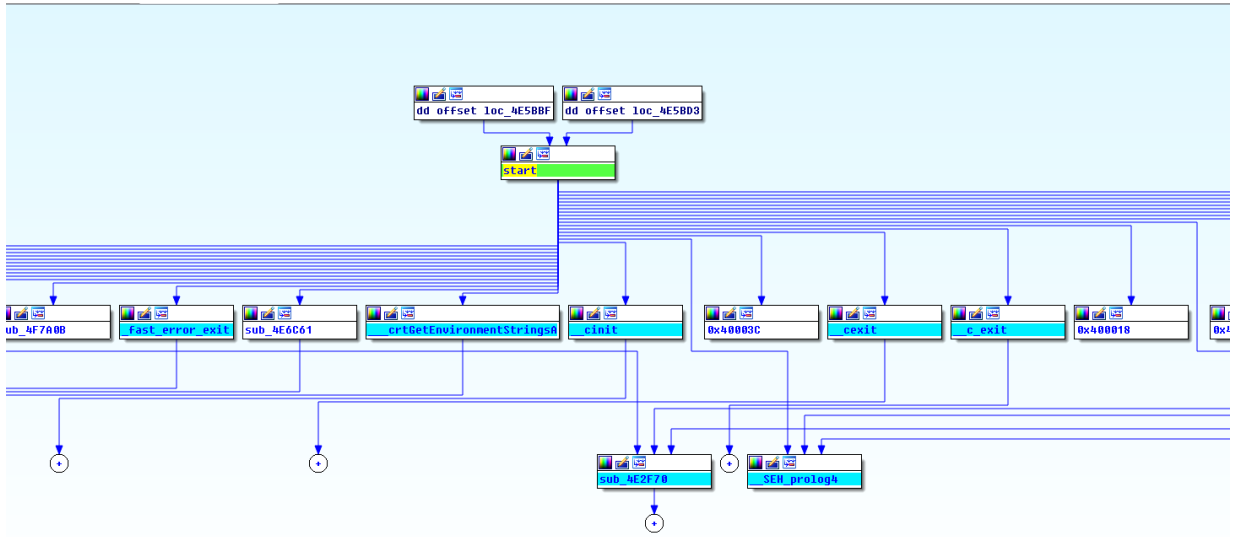
RĪKI, PROCESS UN SOĻI



1.2. att. **Apgrieztās inženierijas rīks “IDA Pro” - moderns disasemblers [2]**



1.9. att. **1.8 attēlā redzamā grafa pārskats**

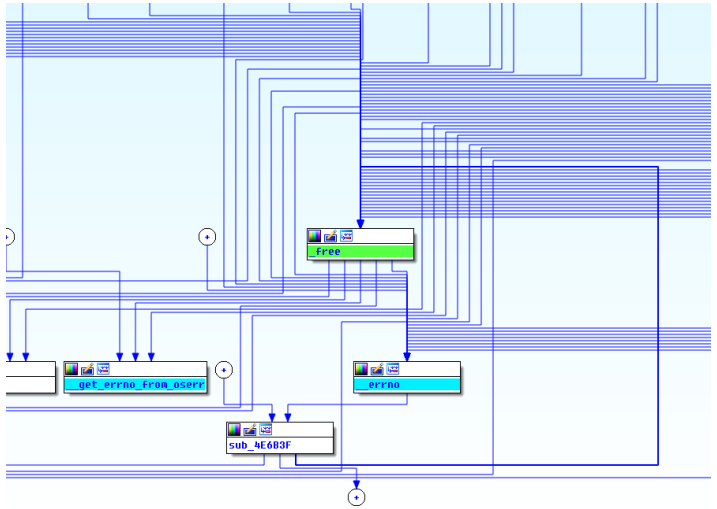


1.8. att. **Grafs no programmas ieejas punkta “start”. Skats, kurš tiek atvērts pēc Notepad++ .EXE faila ielādes IDA Pro**

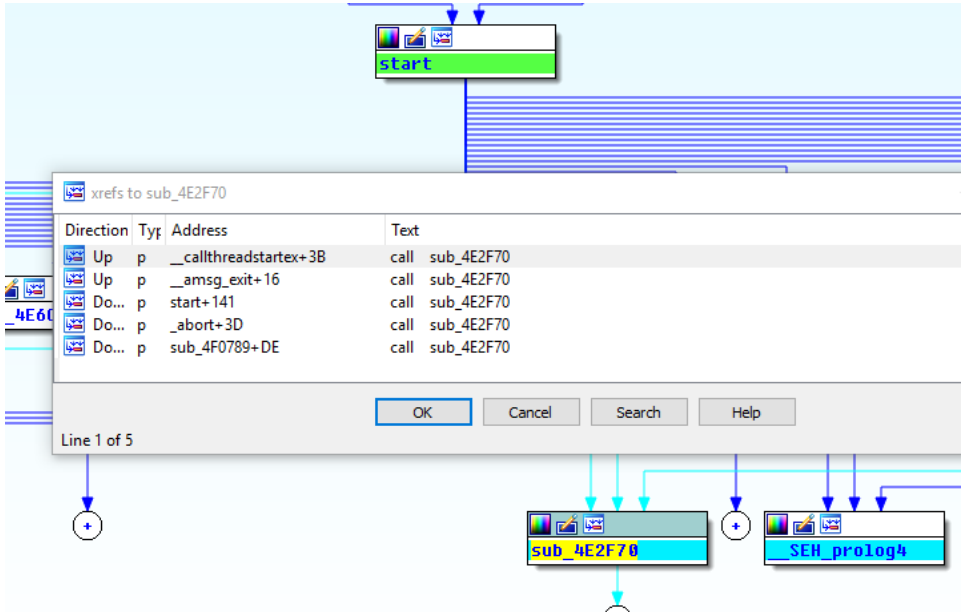
1.12. att. **IDA Pro ģenerētais pseidokods Notepad++ “main” funkcijai (līdzīgs C++)**

```
int v2; // ebx@2
int v3; // eax@17
int v4; // eax@19
int v5; // eax@19
int v7; // [sp-4h] [bp-4h]@1

sub_4F7A58();
_SEH_prolog4(&unk_560EA0, 20);
v1 = (unsigned __int16)sub_4E8921();
sub_4F7A0B(2);
if ( v400000 != 23117 || *(_DWORD *)(v40003C + 0x400000) != 17744 || *(_WORD *)(v40003C + 4194328) != 267 )
{
  v2 = 0;
}
else
{
  v2 = 0;
  if ( *(_DWORD *)(v40003C + 4194420) > 0xEu )
    LOBYTE(v2) = *(_DWORD *)(v40003C + 4194536) != 0;
}
*(_DWORD *)(a1 - 28) = v2;
if ( !sub_4E87A1() )
  fast_error_exit(28);
if ( !sub_4E6C61() )
  fast_error_exit(16);
sub_4F0506();
*(_DWORD *)(a1 - 4) = 0;
if ( sub_4ECD25() < 0 )
  fast_error_exit(27);
dword_583D28 = (int)GetCommandLineA();
dword_581F3C = (char *)__crtGetEnvironmentStringsA();
if ( _setargv() < 0 )
  _amsg_exit(8);
```



1.11. att. **Piemērs ļoti augstam grafa zarošanās faktoram**



1.10. att. **Referenču meklēšana. Ir iespējams navigēt, uz jebkuru no tām**

Binārās analīzes

METODES EFEKTIVITĀTES UZLABOŠANAI

00512160	ReleaseSemaphore	KERNEL32
00512164	SleepEx	KERNEL32
00512168	WaitForSingleObjectEx	KERNEL32
0051216C	CreateSemaphoreW	KERNEL32
00512170	Cancello	KERNEL32
00512174	InterlockedDecrement	KERNEL32
00512178	CreateFileW	KERNEL32
0051217C	ReadDirectoryChangesW	KERNEL32
00512180	MulDiv	KERNEL32
00512184	GetCurrentThreadId	KERNEL32
00512188	GetModuleHandleW	KERNEL32
0051218C	SetCurrentDirectoryW	KERNEL32
00512190	FreeLibrary	KERNEL32
00512194	CopyFileW	KERNEL32
00512198	GetProcAddress	KERNEL32
0051219C	GetCurrentProcess	KERNEL32
005121A0	GetCurrentProcessId	KERNEL32

2.1. att. **Fragments no IDA Pro Notepad++ importu analīzes. Ir iespējams redzēt visu, ko izmanto no "KERNEL32.DLL"**

.rdata:0054975C	00000012	C	not enough memory
.rdata:00549770	0000001F	C	resource unavailable try again
.rdata:00549790	00000012	C	cross device link
.rdata:005497A4	00000013	C	operation canceled
.rdata:005497B8	00000014	C	too many files open
.rdata:005497CC	00000012	C	permission_denied
.rdata:005497E0	0000000F	C	address_in_use
.rdata:005497F0	00000016	C	address_not_available
.rdata:00549808	0000001D	C	address_family_not_supported
.rdata:00549828	0000001F	C	connection_already_in_progress
.rdata:00549848	00000014	C	bad_file_descriptor
.rdata:0054985C	00000013	C	connection_aborted
.rdata:00549870	00000013	C	connection_refused
.rdata:00549884	00000011	C	connection_reset
.rdata:00549898	0000001D	C	destination_address_required

2.2. att. **Fragments no IDA Pro Notepad++ teksta fragmentu analīzes**

«Šī darba kontekstā, definēsim metodes efektīgumu pēc patērētā laika, kurš ir vajadzīgs, lai identificētu ar pētāmo problēmu saistītus mašīnkodā blokus.»

SITUĀCIJAS, KAD AR ESOŠĀM METODĒM VAR BŪT NEPIETIEKAMI

1. Trūkst informācijas:
   1. Ja programmā nav ārējo bibliotēku izsaukumu vai tie ir neatgriezeniski obfuscēti.
   2. Ja programmā nav tekstuālo konstanšu vai tās ir neatgriezeniski obfuscētas.
2. Pārāk daudz informācijas, lai saprastu, kur sākt analīzi.
3. Pietrūkst izpildes vēstures konteksta.
4. Pret populārām metodēm ir standarta obfuskācijas.

Izstrādātā metode tiek piedāvāta kā vēl viens rīks analītiķa arsenālā. Tiek pieņemts, ka tā tiks izmantota kopā ar citām komplementāram, un nav universāli labāka.

ORIĢINĀLI IZSTRĀDĀTĀ METODE UN TĀS ANALĪZE

STARP-BLOKU VARBŪTISKO PĀREJU GRAFS

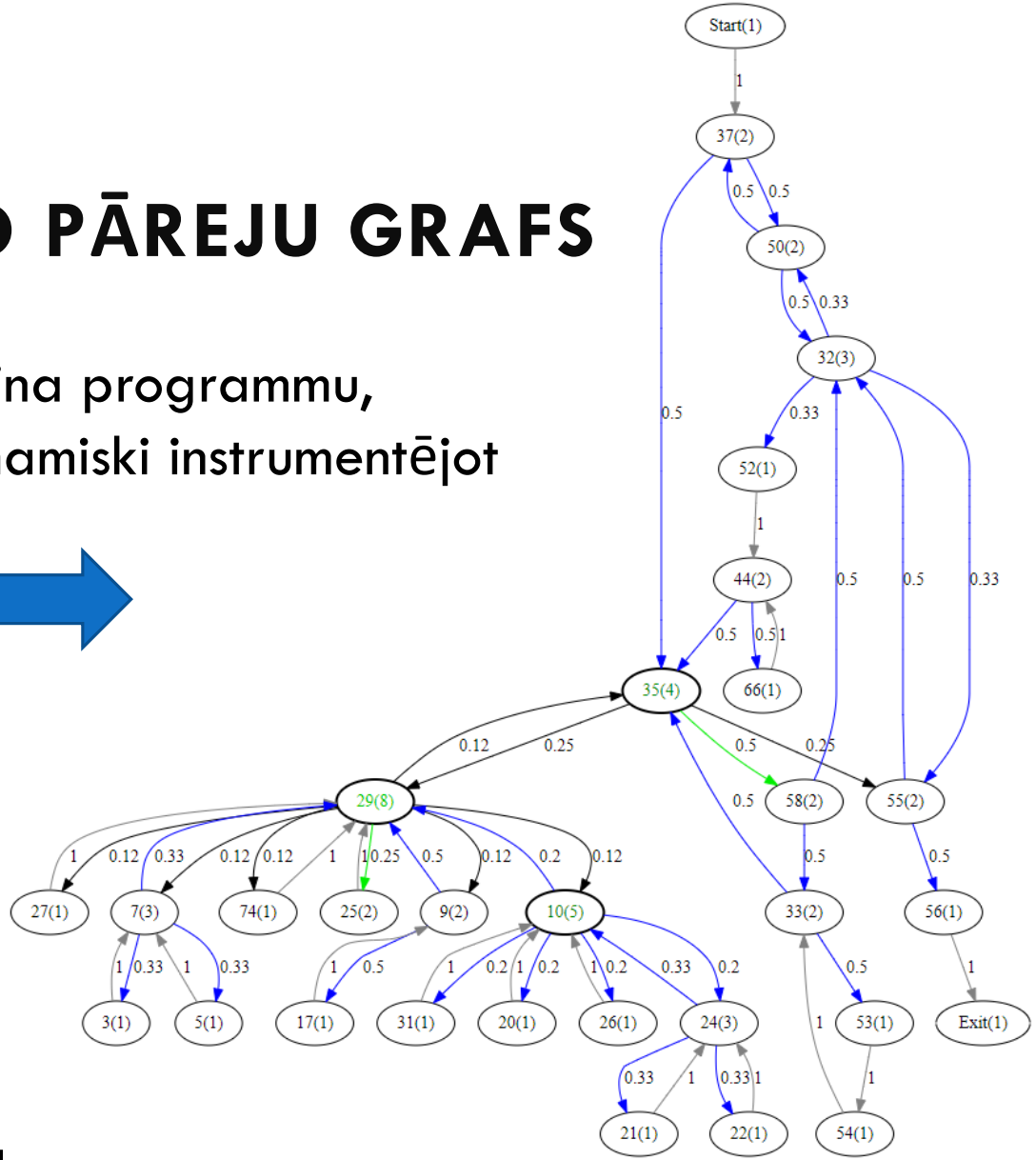
```
Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I press add
    And I have entered 70 into the calculator
    When I press equals
    Then the result should be 120 on the screen

Scenario Outline: Add three numbers
    Given I have entered <ValueA> into the calculator
    And I press add
    And I have entered <ValueB> into the calculator
    And I press add
    And I have entered <ValueC> into the calculator
    When I press equals
    Then the result should be <Result> on the screen

Examples:
| ValueA | ValueB | ValueC | Result |
| 50     | 70     | 20     | 140    |
| -10    | 20     | 20     | 30     |
```

3.1. att. **Piemērs Gherkina kodam, kurš apraksta kādu lietojuma scenāriju programmatūras automātiskajai testēšanai [19]**

Darbina programmu,
to dinamiski instrumentējot

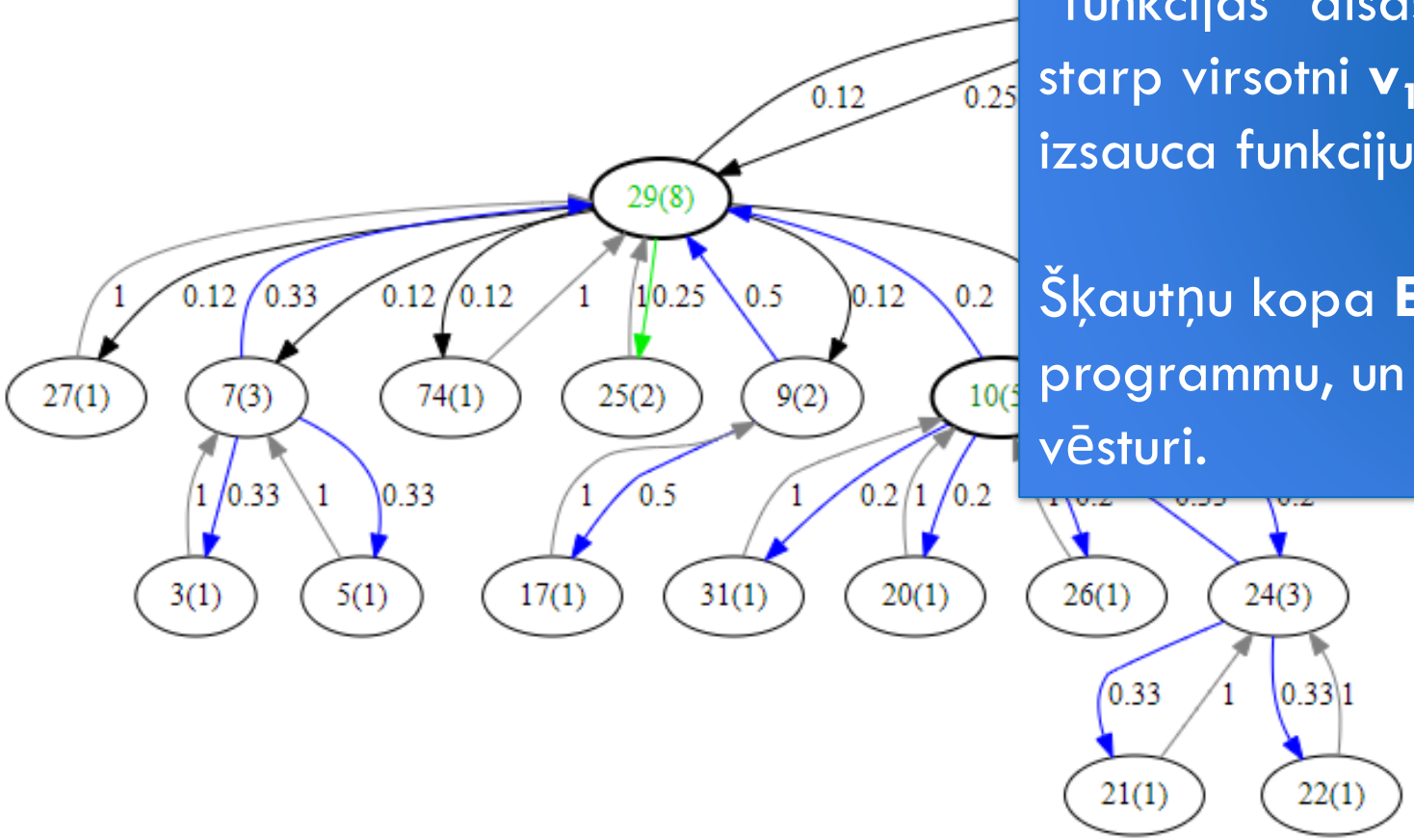


3.13. att. **Apstrādes programmas gala rezultāts ir grafs .dot formātā, kuru ir iespējams vizualizēt ar kādu GraphViz utilītu**

Sastādīsim grafu **G** ar virsotņu kopu **V**, un šķautņu kopu **E**.

Virsotņu kopa **V** ir rekonstruēti steka rāmji jeb "funkcijas" disasemblera izpratnē. Šķautne e_1 starp virsotni v_1 un v_2, ir norāde, ka funkcija v_1 izsauca funkciju v_2.

Šķautņu kopa **E** tiek sastādīta darbinot programmu, un pierakstot visu funkciju izpildes vēsturi.



3.21. att. **"boombox.exe" starp-bloku varbūtisko pāreju grafs, tikai ar programmas loģiku**

PRAKTISKĀ REALIZĀCIJA

Functions window

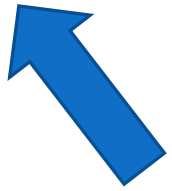
Function name	Segment	Start	Length	Lo
sub_401000	.text	00401000	00000018	
sub_401020	.text	00401020	00000056	00
sub_401080	.text	00401080	00000067	00
sub_4010F0	.text	004010F0	00000067	00
sub_401160	.text	00401160	0000000D	
sub_401170	.text	00401170	000000AE	00
sub_401220	.text	00401220	00000099	00
sub_4012C0	.text	004012C0	00000052	00
sub_401320	.text	00401320	0000001D	
sub_401340	.text	00401340	00000056	00
sub_4013A0	.text	004013A0	000000D0	00
__cfltcvt_init_0	.text	00401470	00000065	
sub_4014E0	.text	004014E0	00000065	
__cfltcvt_init_3	.text	00401550	00000065	
sub_4015C0	.text	004015C0	00000065	
__cfltcvt_init_6	.text	00401630	00000065	

3.2. att. **IDA Pro rekonstruēto funkciju logs**

```
module[  9]: 0x0000000000014028,  12
module[  9]: 0x0000000000022180,   2
module[  6]: 0x000000000000419c,   9
module[  6]: 0x00000000000045dc,  40
module[  6]: 0x0000000000004673,  21
module[  6]: 0x00000000000041a5,   9
module[  6]: 0x0000000000003fe0,  34
module[  6]: 0x0000000000004010,  11
module[  6]: 0x0000000000004025,  10
module[  6]: 0x000000000000402f,  29
module[  6]: 0x0000000000004820,   6
module[  7]: 0x0000000000022778,  23
module[  7]: 0x000000000002278f,   4
module[  7]: 0x0000000000022793,   8
module[  7]: 0x000000000002279d,   9
module[  7]: 0x000000000002279b,   2
module[  6]: 0x0000000000003ea0,  18
module[  6]: 0x0000000000003eb6,  25
module[  6]: 0x0000000000003ecf,  11
module[  6]: 0x0000000000003eda,  11
module[  6]: 0x0000000000003ee5,  26
module[  7]: 0x0000000000021280,   7
module[  6]: 0x0000000000003eff,  10
module[ 10]: 0x0000000000071270,  39
```

3.6. att. **“drcov” rīka izvada .log failā turpinājums ar apmeklētajām adresēm**

IDA Pro



Ievada dati grafa sastādīšanai

Dynamo RIO

The Dr. is in.

node_stats.txt

```
1  Avarage calls: 4, standard deviation: 5.444525428617
2  Node nr. Start (Start), called count: 1
3  Node nr. 2679 (sub_4CAEA0), called count: 30
4  Node nr. 78 (sub_402DC0), called count: 9
5  Node nr.
6  Node nr.
7  Node nr.
8  Node nr.
9  Node nr.
10 Node nr.
11 Node nr.
12 Node nr.
13 Node nr.
```

node_stats.txt

```
306 Node nr. 1387 (sub_45DFE0), called count: 1
307 Node nr. 1843 (sub_486B30), called count: 2
308 Node nr. 1755 (sub_47EAA0), called count: 1
309 Node nr. 1844 (sub_486E20), called count: 1
310 Node nr. Exit (Exit), called count: 1
311 Node nr. 2679 (sub_4CAEA0), is called 2 STDEV more than AVG.
312 Node nr. 2934 (_memmove_0), is called 1 STDEV more than AVG.
313 Node nr. 2858 (sub_4D92B0), is called 1 STDEV more than AVG.
314 Node nr. 1893 (sub_48CF50), is called 1 STDEV more than AVG.
315 Node nr. 3262 (sub_4EB5D8), is called 1 STDEV more than AVG.
316 Node nr. 2422 (sub_4BE010), is called 1 STDEV more than AVG.
317 Node nr. 3080 (sub_4E4FCA), is called 1 STDEV more than AVG.
318 Node nr. 3389 (sub_4F4DAE), is called 2 STDEV more than AVG.
319 Node nr. 2542 (sub_4C3A00), is called 2 STDEV more than AVG.
```

3.11. att. **Izsaukumu skaitam tiek aprēķināts arī vidējais un standartnovirze**

ĀTRDARBĪBAS ANALĪZE

Kopā algoritma soļi:

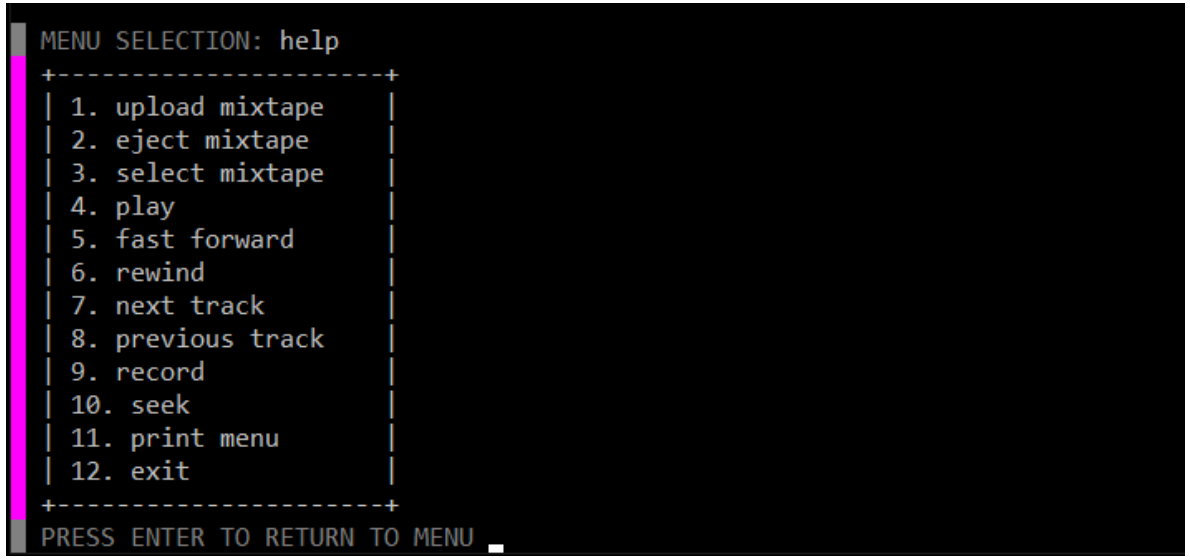
1. No disasemblera iegūt funkciju sarakstu N.
2. Izmantojot dinamisko instrumentētāju iegūt bez-zarošanās pamata bloku izpildes vēsturi M.
3. Par katru bez-zarošanās pamata bloku m_i no M, noskaidrot, kuras funkcijas n_i no N adrešu diapazona tas atrodas. Katrai funkcijai n_i ir sākuma adrese un beigu adrese. Piemēram m_1 = 0012. n_1 = 0000:0016. Tas nozīmē, ka m_1 tiek translēts par n_1.
4. Ja vienā funkcijā ir vairāki bez-
   jāapvieno vienā funkcijā. Tas n
   n_2".
5. Pievieno "Start" un "Exit" funk
   izejas vizualizācijai.
6. Par visām funkcijām sarakstā ti
   izsaukumu (atkārtojumu) skaits
   par 1 vai 2 standartnovirzēm no
7. Par katru funkciju tiek pierakstī
   n_1 n_2 n_1 n_3. Sanāk aiz n_1 seko 2
   seko funkcija "Exit". Uzreiz tie
   tas ir 66% (0.66) n_2, un 33% (0.
8. Izvads par katru virsotni (funkci

Ātrdarbības analīze soļiem:

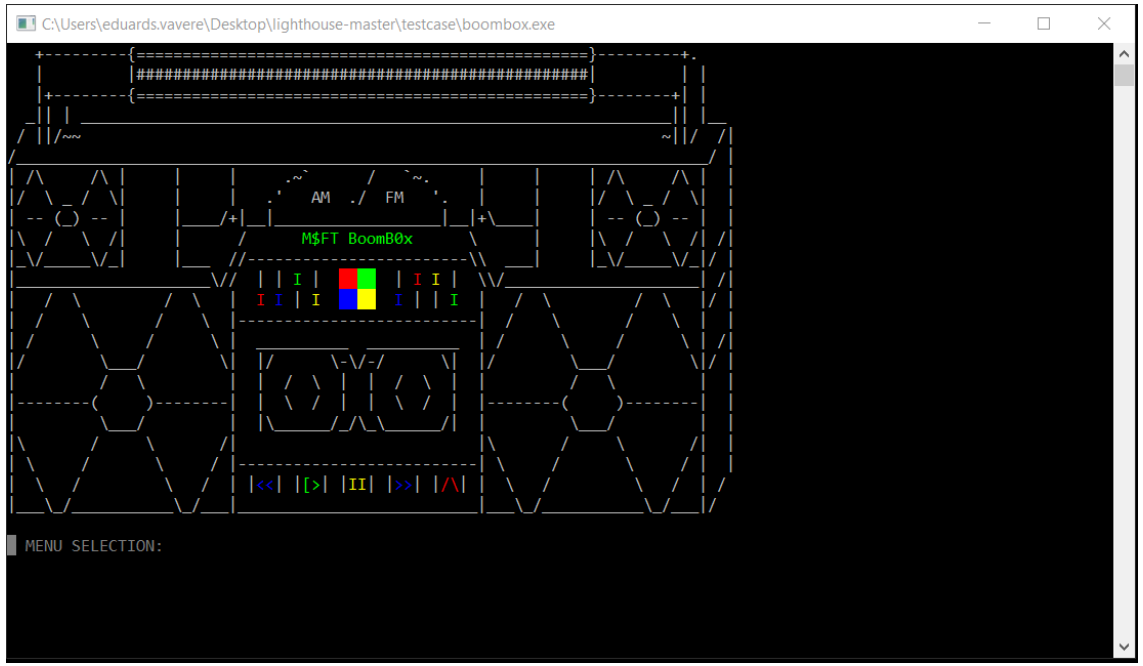
1. O(1)
2. O(1)
3. O(N * M)
4. O(M)
5. O(1)
6. O(3*N)=O(k*N)=O(n) (k=3, jo tie ir 3 "foreach" cikli pēc kārtas)
7. O(N)
8. O(N + M)

Kopumā ir O(N * M), kur ir N funkciju saraksts, un M ir trasēto bez-zarošanās pamata bloku izpildes vēsture.

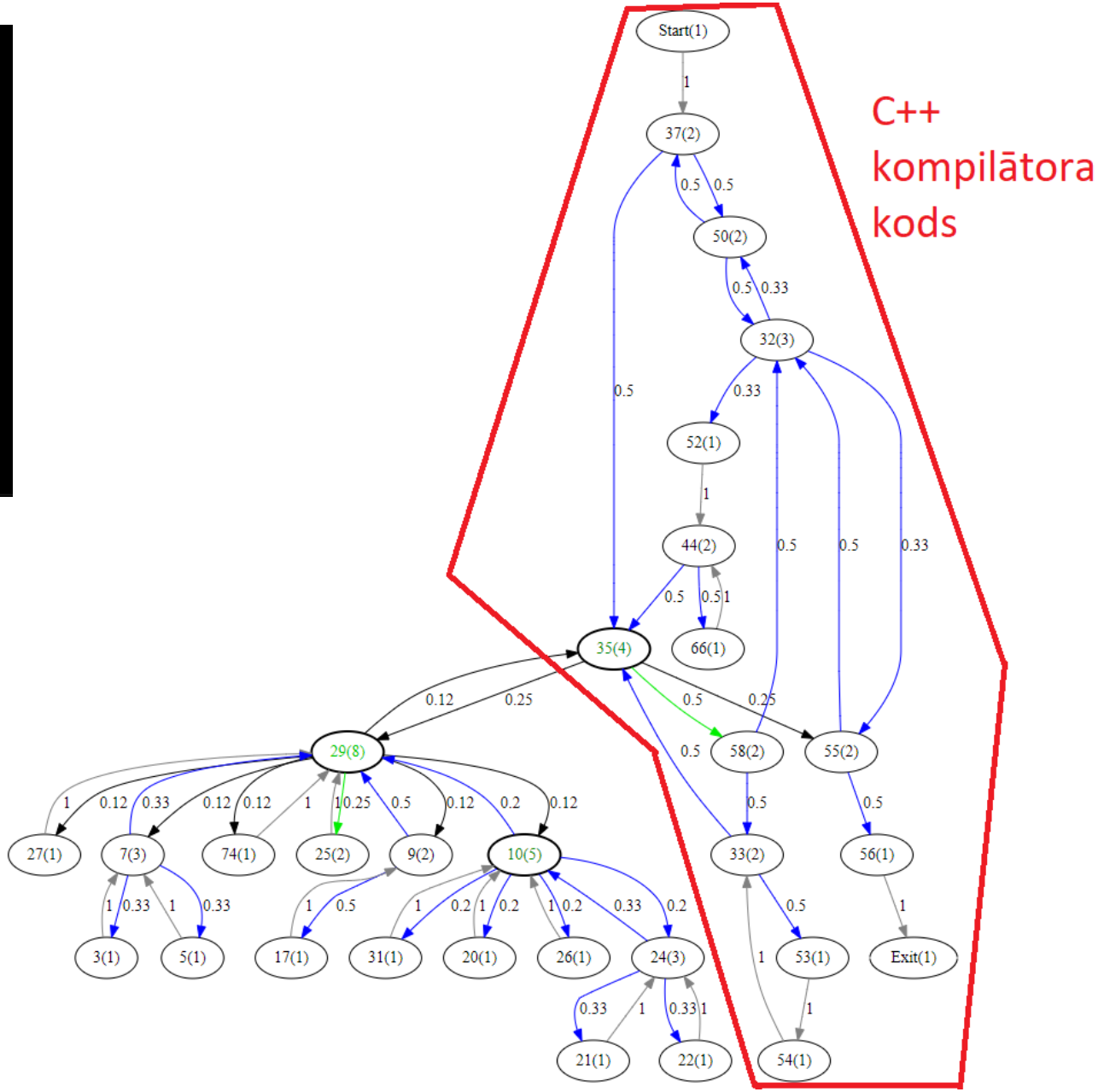
PRAKTISKS PIELIETOJUMS – MAZA PROGRAMMA



3.15. att. **“boombox.exe” izvēlne**



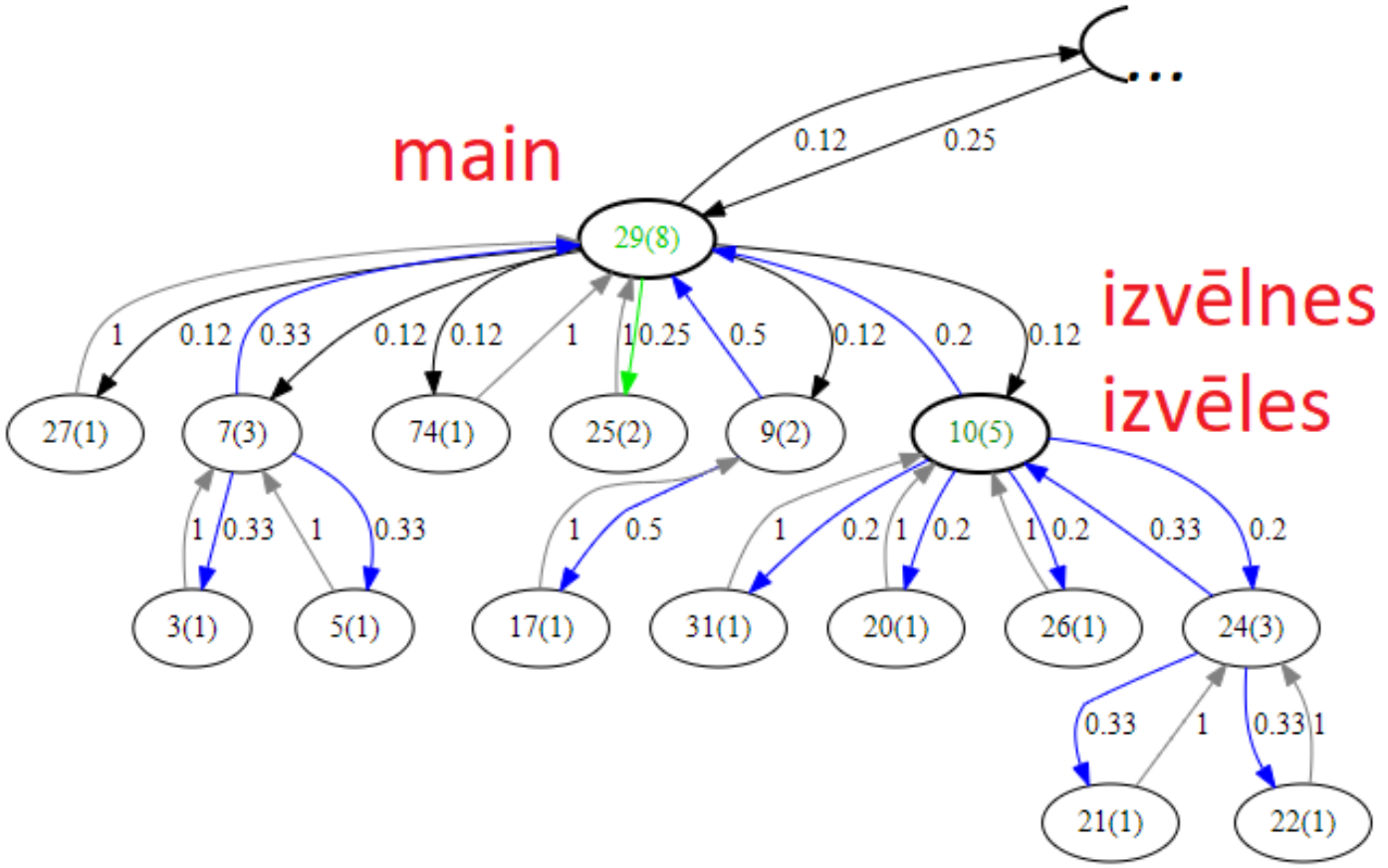
3.14. att. **“boombox.exe” testa programma**



3.20. att. **“boombox.exe” starp-bloku varbūtisko pāreju grafā iezīmēta C++ kompilatora izpildes laika infrastruktūra**

```
MENU SELECTION: help
+----------------------+
| 1. upload mixtape    |
| 2. eject mixtape     |
| 3. select mixtape    |
| 4. play              |
| 5. fast forward      |
| 6. rewind            |
| 7. next track        |
| 8. previous track    |
| 9. record            |
| 10. seek             |
| 11. print menu       |
| 12. exit             |
+----------------------+
PRESS ENTER TO RETURN TO MENU
```

3.15. att. **“boombox.exe” izvēlne**



3.22. att. **“boombox.exe” starp-bloku varbūtisko pāreju grafā var saskatīt “main” galveno kontroles bloku**

```
25  memset(&Dst, 0, 0x5Cui64);
26  v3 = 1;
27  hConsoleOutput = GetStdHandle(0xFFFFFFF5);
28  GetConsoleScreenBufferInfo(hConsoleOutput, &ConsoleScreenBufferInfo);
29  word_1400075D8 = ConsoleScreenBufferInfo.wAttributes;
30  while ( 1 )
31  {
32    while ( 1 )
33    {
34      if ( v3 )
35        sub_140003320();
36      *(_QWORD *)Buf = 0i64;
37      v23 = 0i64;
38      v24 = 0;
39      v3 = 1;
40      SetConsoleTextAttribute(hConsoleOutput, 0x88u);
41      printf("|");
42      SetConsoleTextAttribute(hConsoleOutput, 8u);
43      printf(" %s", "MENU SELECTION: ");
44      v4 = _iob_func();
45      fflush(v4 + 1);
46      SetConsoleTextAttribute(hConsoleOutput, word_1400075D8);
47      v5 = _iob_func();
48      fgets(Buf, 20, v5);
49      if ( Buf[0] != 13 && Buf[0] != 10 )
50        break;
51      v3 = 0;
52    }
53    if ( Buf[0] != 49 || Buf[1] != 10 || Buf[2] )
54      break;
55    sub_1400012B0(&v19);
56LABEL_69:
57    SetConsoleTextAttribute(hConsoleOutput, 0x88u);
58    printf("|");
59    SetConsoleTextAttribute(hConsoleOutput, 8u);
60    printf(" %s", "PRESS ENTER TO RETURN TO MENU ");
61    v17 = _iob_func();
62    fflush(v17 + 1);
63    SetConsoleTextAttribute(hConsoleOutput, word_1400075D8);
64    for ( i = getchar(); i != 10; i = getchar() )
65    {
66      if ( i == -1 )
67        break;
68    }
69  }
```

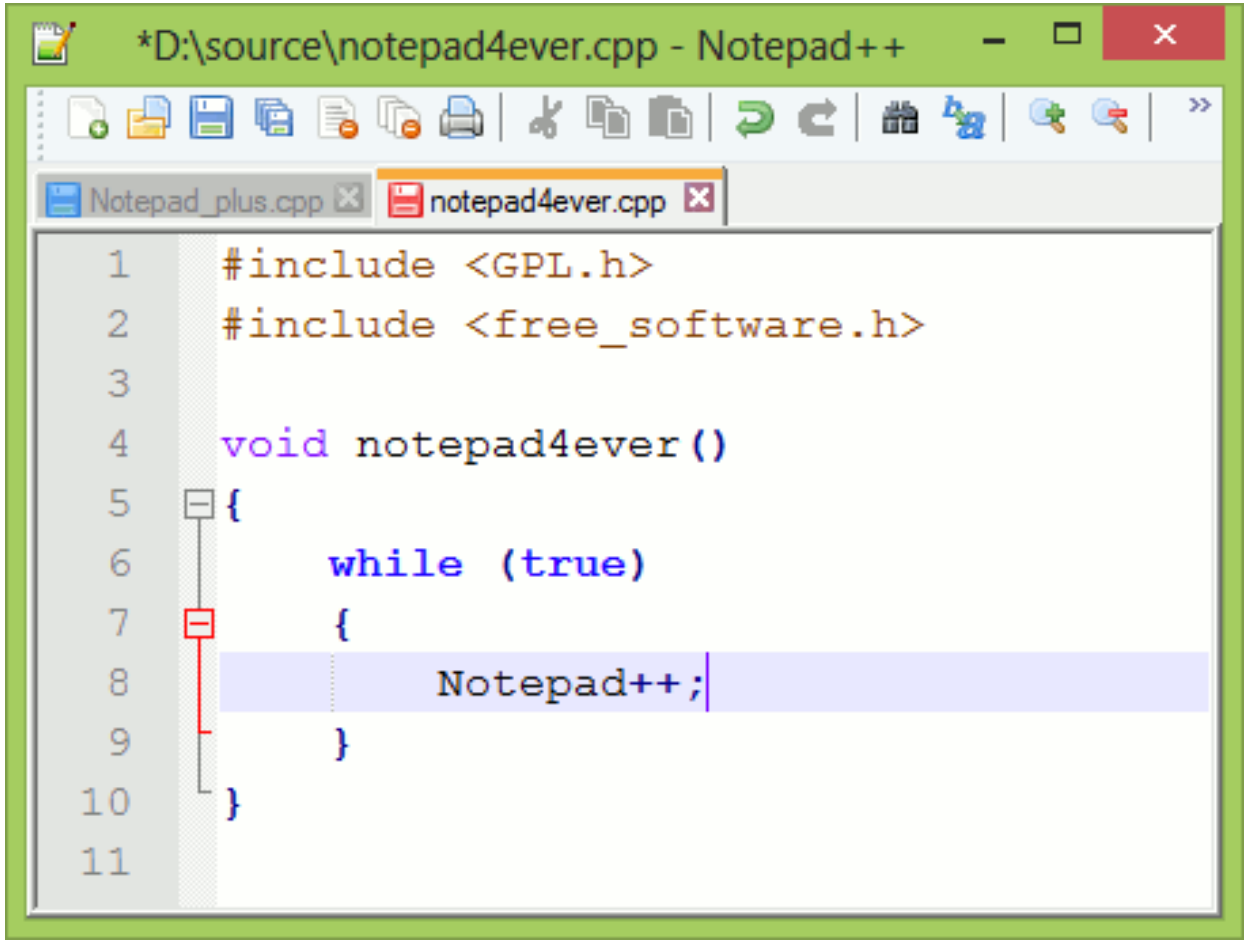
3.23. att. **“boombox.exe” IDA Pro ģenerētais pseidokoda fragments 29. virsotnei (“main”)**

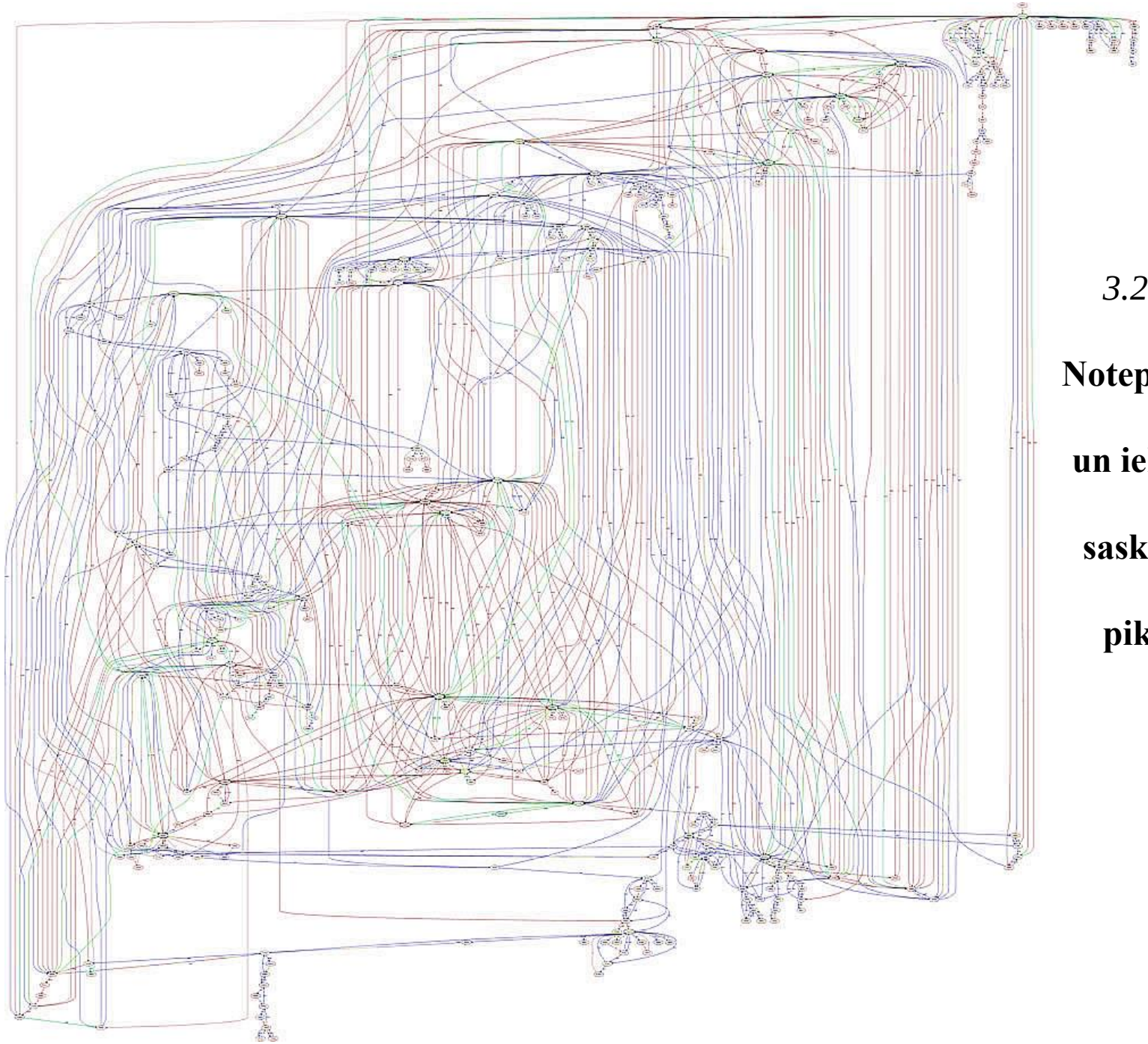
PRAKTISKS PIELIETOJUMS – LIELA PROGRAMMA

500 000+ RINDIŅAS AR KODU

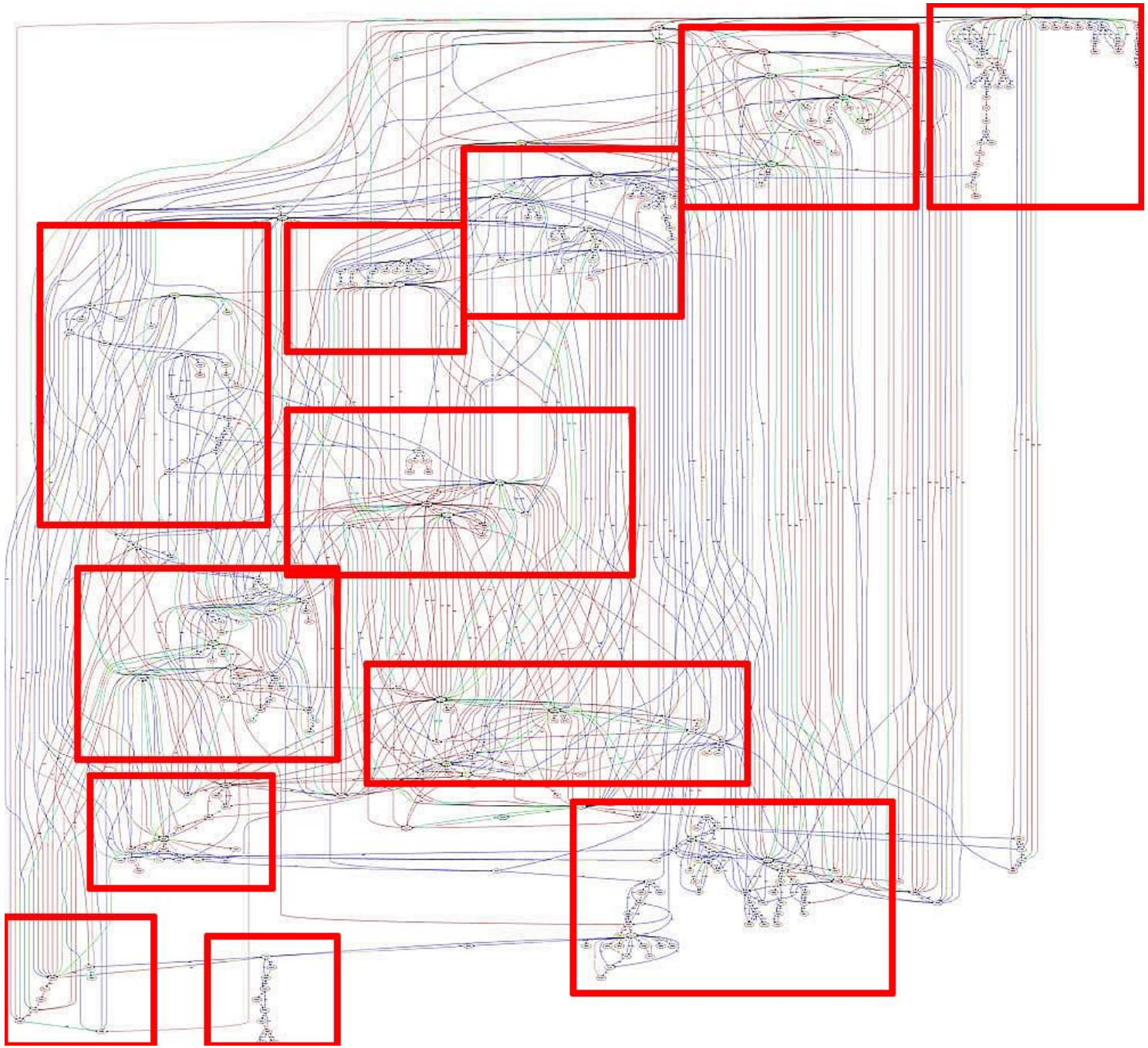
Lietojuma scenārijs:

1. Atvērt programmu
2. Ievadīt tekstu
3. Saglabāt failu un aizvērt programmu



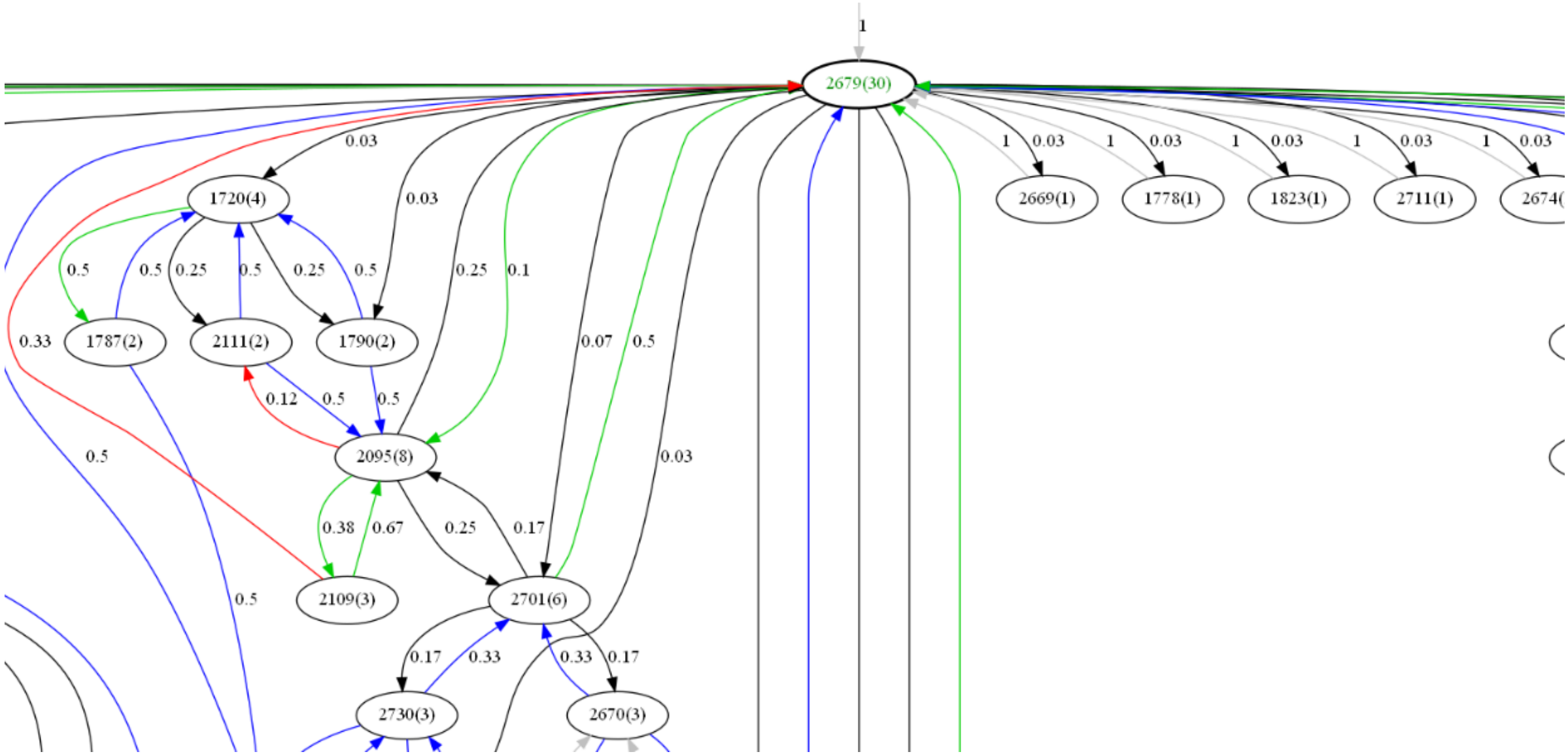


3.25. att. **Starp-bloku varbūtisko pāreju grafs Notepad++ lietojuma scenārijam (bildes kontrasts, un iekrāsojums ir palielināti, lai to vieglāk varētu saskatīt ļoti attālinātu, pilns 14MB 12000x11000 pikseļu izmērs ir publicēts 3.1.3. apakšnodaļā norādītājā GitHub lapā)**

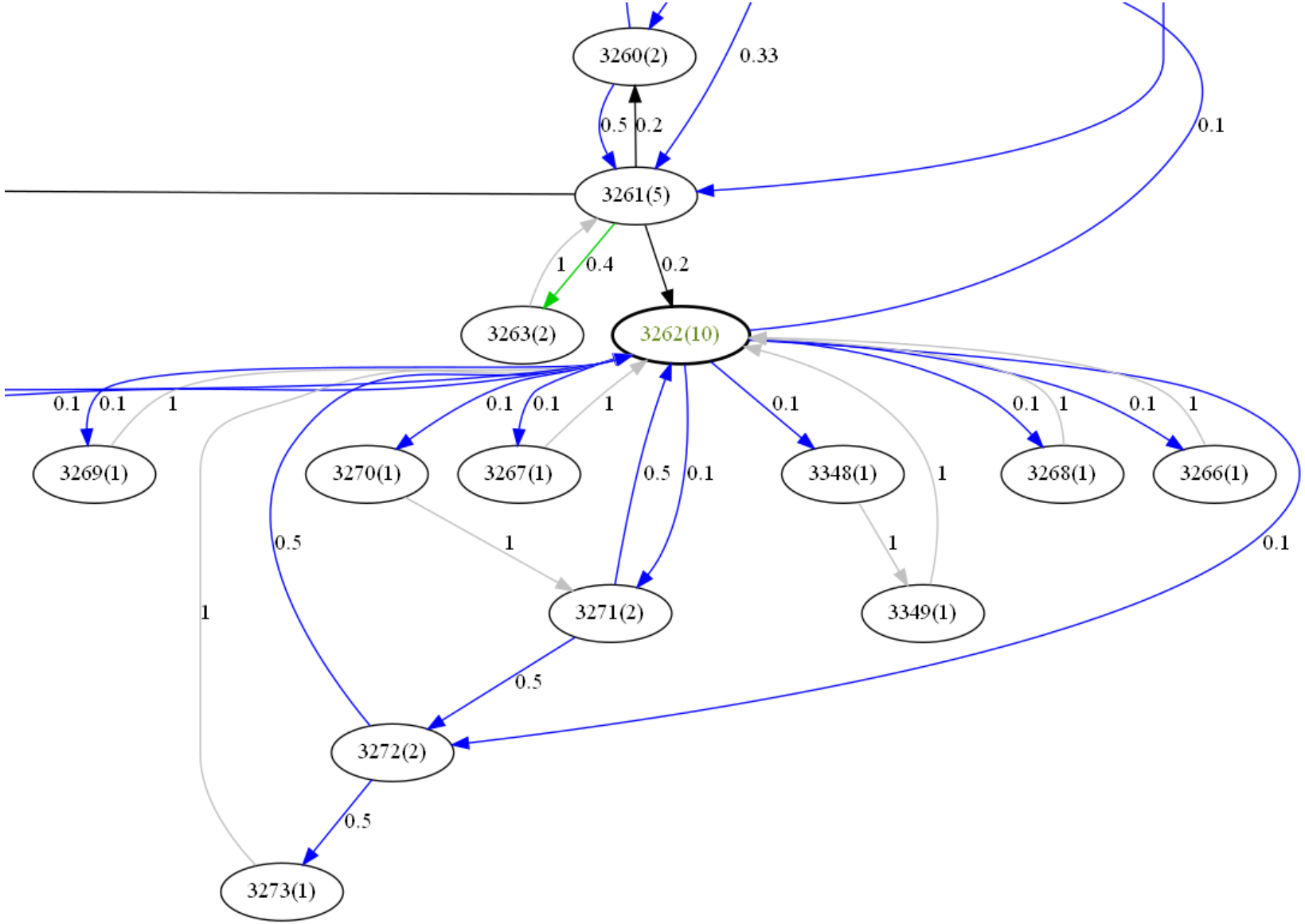


3.26. att. **Starp-bloku varbūtisko pāreju grafs ar vizuāli identificētiem programmas moduļiem**

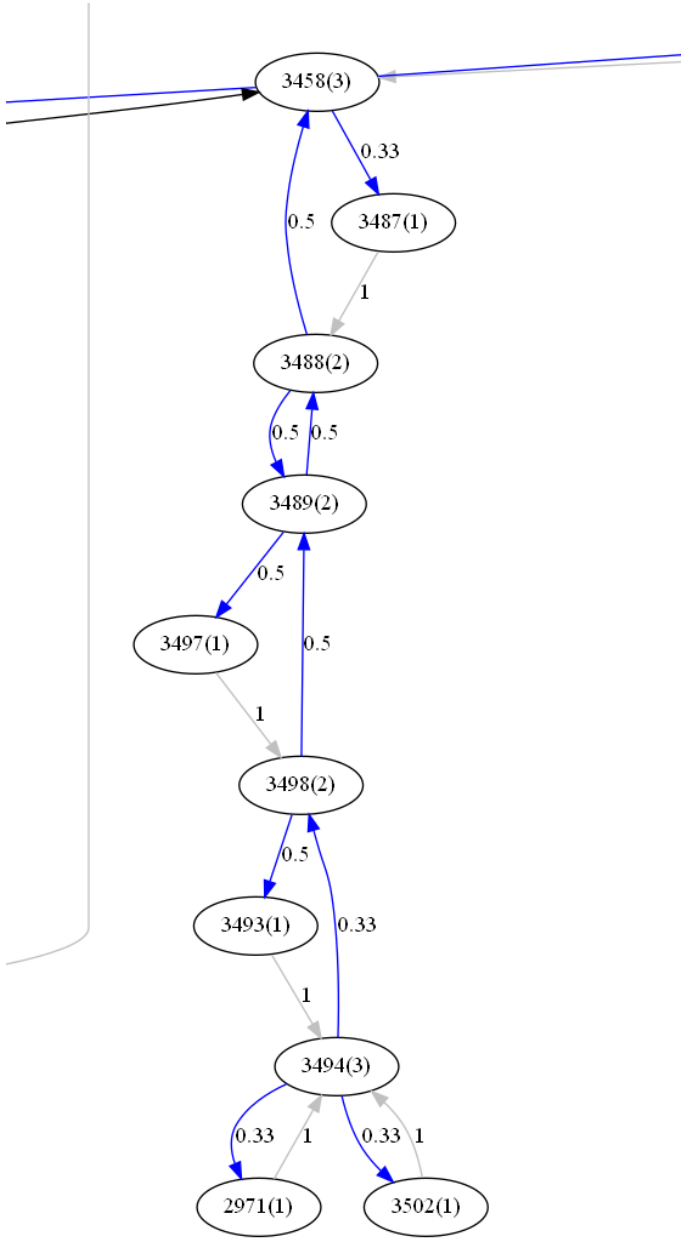
3.27. att. **Starp-bloku varbūtisko pāreju grafā analīzei izvēlētie reģioni**



3.28. att. **Starp-bloku varbūtisko pāreju grafā 1. reģions**

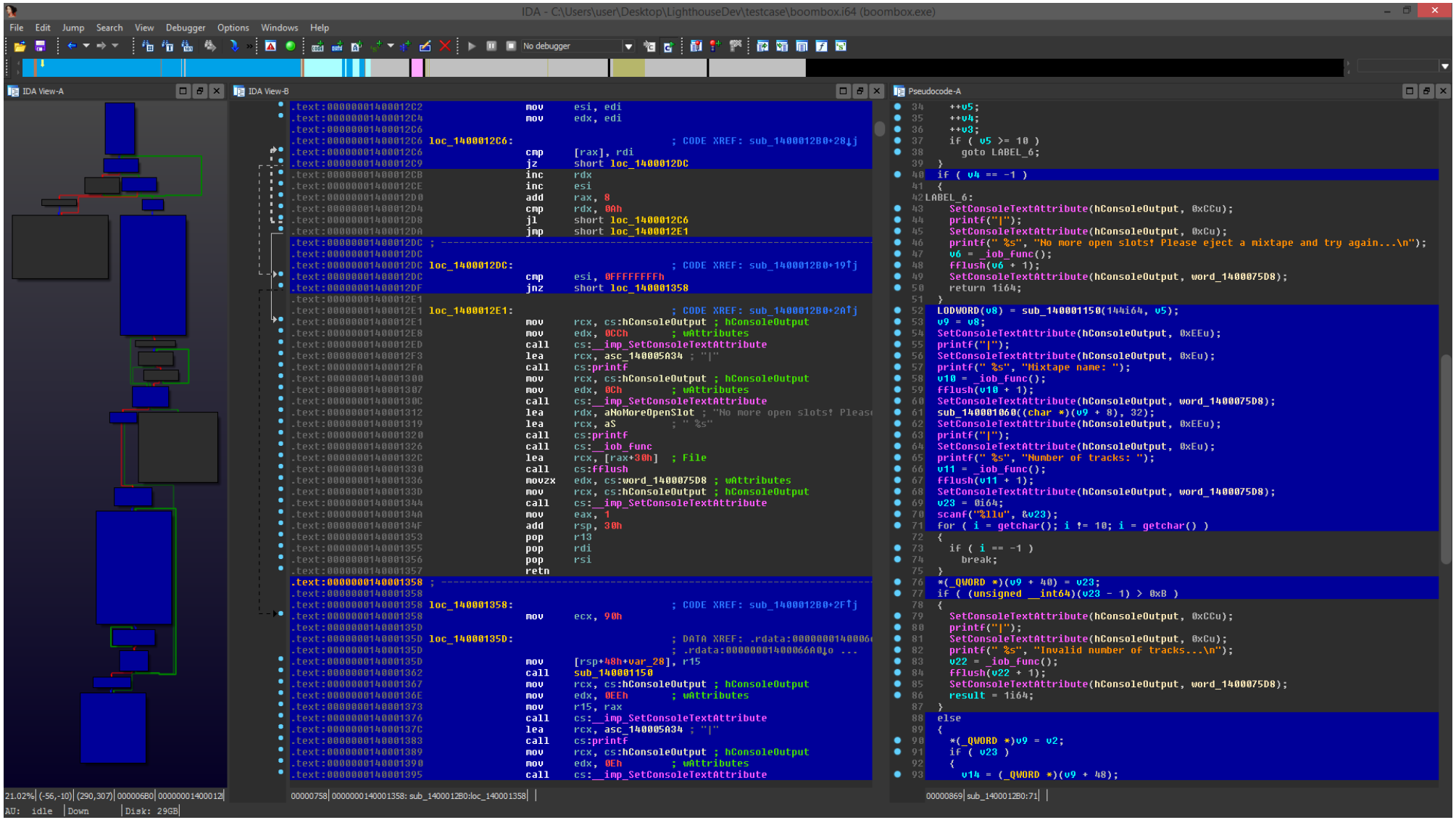


3.31. att. **2. analīzei izvēlētais kopējā grafa reģions**



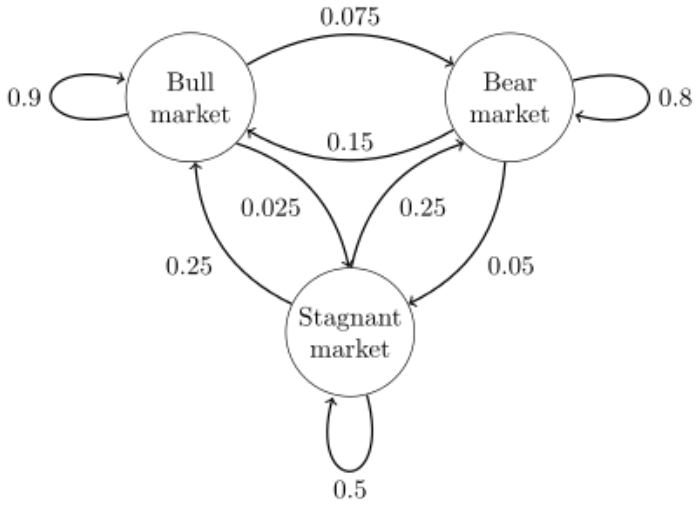
3.33. att. **3. analīzei izvēlētais reģions**

INTEGRĀCIJAS IESPĒJAMĪBA ESOŠOS RĪKOS

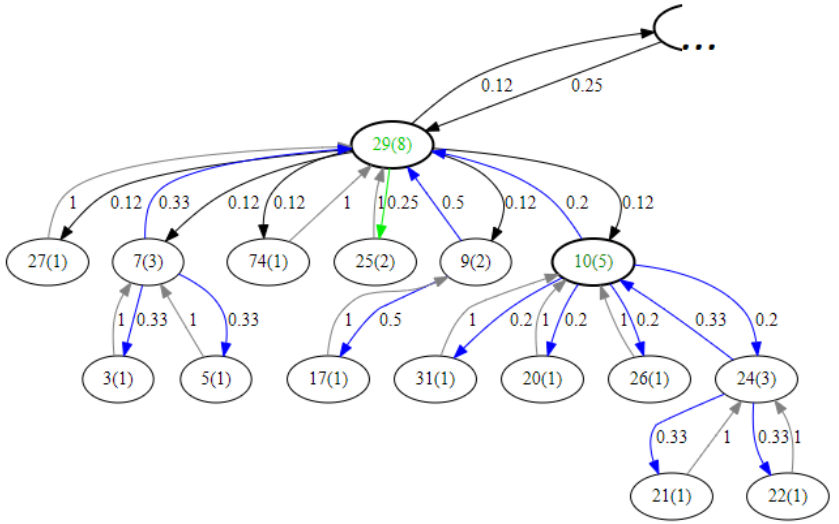


2.5. att. **IDA Pro paplašinājums “lighthouse”, kurš parāda izpildītā koda pārklājumu. Ar zilu ir iekrāsots apmeklētais kods izpildot programmu [17]**

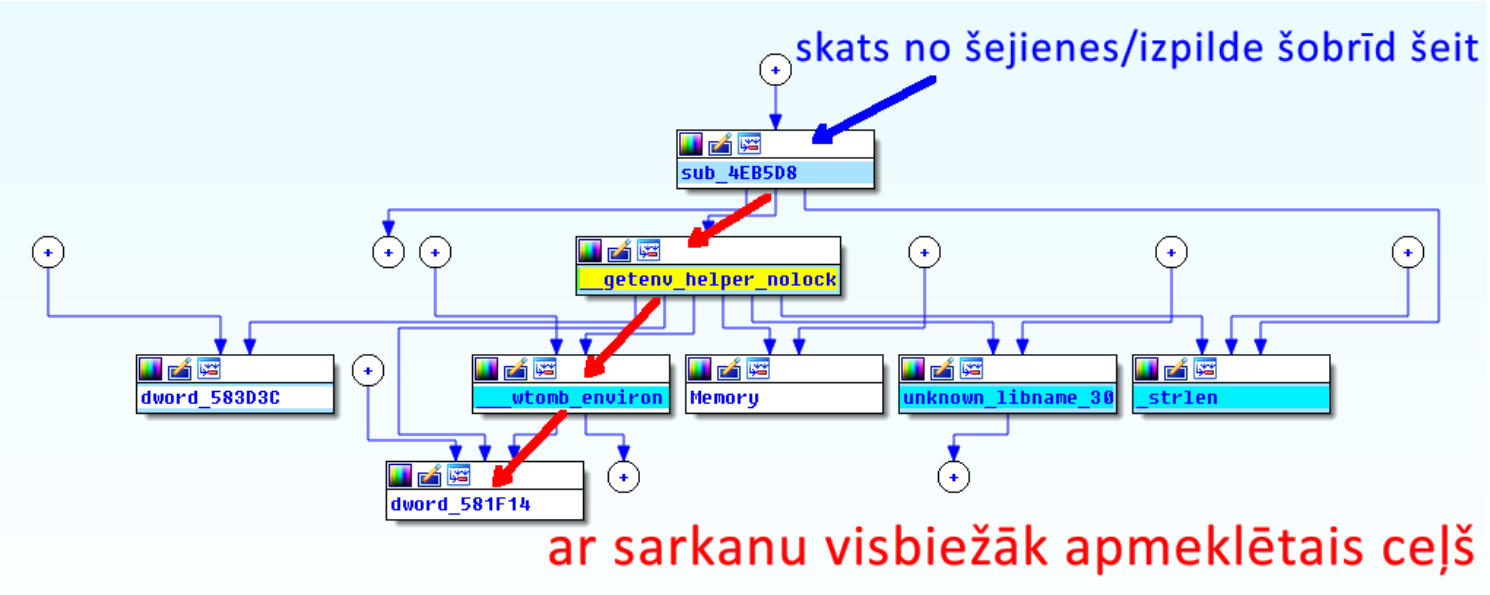
NĀKOŠO BLOKU PAREDZĒŠANA



3.37. att. **Ilustrēta Markova ķēde ar 3 stāvokļiem [18]**



3.38. att. **Starp-bloku varbūtisko pāreju grafs ir Markova ķēde**



3.40. att. **IDA Pro statiskās analīzes funkciju izsaukumu grafu būtu iespējams papildināt ar izpildes vēstures informāciju**

REZULTĀTI UN SECINĀJUMI

REZULTĀTI

1. Piedāvāta oriģināla metode binārās analīzes efektivitātes uzlabošanai – starp bloku varbūtisko pāreju grafs.
    1. Balstās uz jau eksistējošām bloku trasēšanas un funkciju grafiskās vizualizēšanas metodēm, bet ar pievienotu izpildes vēstures statistisko analīzi.
2. Šai metodei tika aprakstīts teorētiskais pamatojums, un piedāvāti vairāki teorētiski gadījumi, kad tā var sniegt ieguvumus.
3. Piedāvātā metode tika realizēta praktiskā programmā, aprakstot programmas darbības soļus un algoritmu.
    1. Tai tika veikta praktiskās ātrdarbības un asimptotiskā analīze. Metode tika testēta uz divām programmām: mazas (ap 1000 rindiņu koda), un ļoti lielas (virs 500 000) rindiņām koda
4. Tika veikta izpēte, kā aprakstīto metodi iebūvēt industrijas rīkos, un kādi ir tās galvenie ieguvumi un nepilnības. Tika norādīts uz turpmāku pētījumu iespējām.

SECINĀJUMI

1. Startējot programmu ar iepriekš sastādītiem scenārijiem, var trasēt tās funkciju izpildes vēsturi.
2. Ar šīs metodes palīdzību ir iespējams atrast noderīgus analīzes sākumpunktus.
3. Veicot metodes ieguvumu un nepilnību analīzi (3.1.8 apakšnodaļa), tika secināts, ka tā nav universāli pielietojama visos gadījumos, bet atsevišķos gadījumos var būt efektīva.
4. Šo metodi ir iespējams integrēt esošās analīzes vidēs (piem. IDA Pro), kā neatkarīgu paplašinājumu vai eksistējošos (piem. "Lighthouse") paplašinājumos.
5. Starp-bloku varbūtisko pāreju grafu var izmantot, lai paredzētu programmas turpmāko darbību (programmu modelējot kā Markova procesu).
    1. Tas var būt noderīgi, ja nav deterministiski iespējams noteikt ārēju izsaukumu (piem. tīkla) vai lietotāja ievada rezultātus. Tomēr, ir jāveic papildus pētījumu, lai pateiktu vai programmas kā stohastiska procesa modelēšana ir noderīga binārajā analīzē. Šiem pētījumiem var izmantot literatūru no procesoru arhitektūras par zarošanās paredzēšanu ("probabilistic branch prediction").

PALDIES PAR UZMANĪBU!