NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY
[IJIERT] ISSN: 2394-3696 Website: ijiert.org
VOLUME 9, ISSUE 12, Dec. -2022

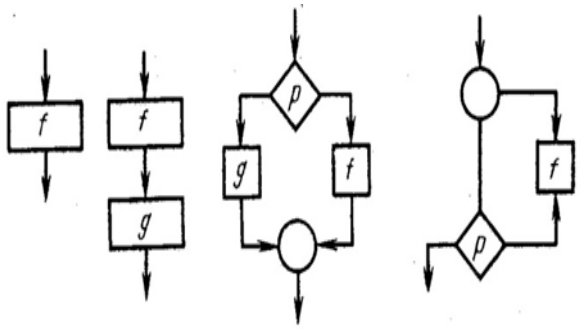
CONSTRUCTING THE MINIMUM COVERAGEOF THE I GRAPH

M. M. Arripov
Candidate of Technical Sciences, Associate Professor of the
Department of Informatics of the Kokand State Pedagogical Institute

ABSTRACT
Describes the structural methods of testing programs such as branch testing, program verification, symbolic testing and the generation of structural tests. An algorithm for the minimum coverage of the program graph based on a packaged adjacency matrix and a specific example of the minimum coverage of the program graph are introduced.

Keywords - testing, program graph, minimum program graph coverage, packed adjacency matrix, DD paths, vertex, branches, g graph, h graph, algorithm complexity, Python, IF, WHILE.

Among the many methods proposed to increase the reliability of programs, the method of structural programming has become of great importance. A program is called a structure in relation to some basic set of program control structures if it is composed of these control structures suitably interconnected. A generally recognized set of basic program control structures is shown in Figure 1.



a) b) c)

Рис.1. Набор базовых структур программы: a – линейная программа;
b – разветвленные типа IF; c – цикл типа WHILE

These basic structures are called D elements. Accordingly, a program composed of these basic structural elements is called a D-structured program. It has been proven that D elements are a complete (sufficient) basis for compiling any programs or describing any algorithms.

Some program can be described as a graph G=(V,E) consisting of a plurality of vertices V and a set of arcs E. Let each vertex v_i , which is part of V (v_i Є V) be somelinear section (sequence of commands without branching) of the program. Let each arc (v_i v_j) of the graph be a transfer of control from the linear section i linear section j. Let's call a graph g ca structured graph or a d graph if itdescribes a structural program. Although structural programming increases the reliability of the program, there is no need for tests because you need to prove the correctness of the program. It is usually required that the tests provide at least one execution of each instruction or each forks in the program.

On a modeland a program in graph form, this corresponds to finding many paths covering all the vertices or all the arcs of the graph. Often the set of tests that lead to the execution of all the operators or all the branches in the program is insufficient. On the other hand, testing all paths in the program structure is impractical or even impossible because of their a lot of it. The trade-off solution is path testing, which provides testing of critical interactions between parts of the program. On the graph describing the program,

such interactionswithtwiya can beat the simulated by introducing the required pairs, i.e., pairs of vertices that must interact in at least one test.

The graph is represented as a packaged adjacency matrix (UMS). The packed adjacency matrix A = { and$_{ij}$} of a graph with v vertices is the v x l matrix (l is the maximum degree of output of the t-th vertex). The degree of input $d_{tox}(v_i)$ and the output d_{of}(v i) of some vertex of the graph means, respectively, the number of arcs entering and exiting the vertices. Each row i of the UMS is filled in random order with vertex numbers that are adjacent with vertex i.

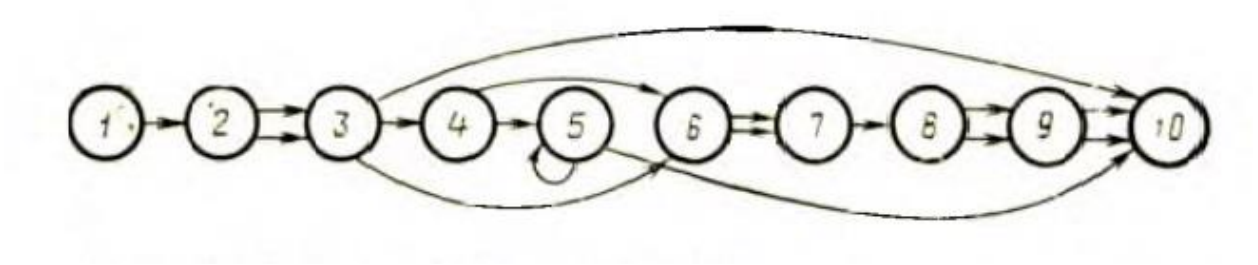
The graph view as a UMC has the following advantages over other existing representations: for large graphs, the number of UMC columns is significantly smaller than the number of columns in the corresponding adjacency matrix; it is relatively simple to simulate the process of moving along the graph to build paths; the graph processing time is reduced. The criterion for testing is the criterion of branches, where under the branch of the program thereis someconsistency of operators executed strictly one after another. Thus, the branch is a linear section of the program. To construct a minimum coverage, the graph is divided into DD paths using the UMS of the original graph. A set of vertices whose output $d(v_i)>1$, the input and output vertices are denoted as D vertices. Then the DD path is a simple path between two D vertices, such that there are no D vertices within it. Cycles and loops are then defined and the arcs that close them are excluded.

The proposed algorithm for constructing a minimum coverage (BMS) of the graph consists of the following stages. An example of a program graph is shown in Fig. 2.

Step 1. The i-th vertex is viewed and the adjacent vertex j is determined, the number of which is the maximum among the numbers of adjacent vertices, where i Є { l , n −1;} n is the number of vertices of the graph.

Этап 2. Просматривается дуга (v i, v j). Если d вх (v i) > 1 и d вых(v j) >1 , то дуга g(v i,v j) исключается. Если $d_{вых}$ (v i) > 1 и $d_{вх}$(v j) = 1, то дуга h(v_i, v_j)

Notes.



Rice. 2. Example of a programgraph s

Step 3: Substitute i = j and repeat steps 1-2 until j is equated with the number of the final (output) vertex. Fixes the path as a sequence of values j.

Step 4. If there are no arcs of type g in the constructed path, then the last arc of type h is excluded.

Step 5: Repeat steps 1-2 until there are no arcs of type g and h in the constructed path.

An example of building a minimum coverage of a program graph. Let be the graph of the program shown in Fig. 1. Graph arcs mean a sequence of computational program operators, ingraph ruffs mean branching and unification operators. After excluding the closing arc cycles (they are tested separately), the graph is drawn. Figure 1 is described by the following UMC:

1	2	0	0
2	3	3	0
3	4	6	10
4	5	6	0
5	10	0	0
6	7	7	0
7	8	0	0
8	9	9	0
9	10	10	0
10	0	0	0

The first stages of the ISTOC algorithm give the following results:

Этап 1. Устанавливается i = 1, j = 2. {1, 2}

Step 2. The arc (v_i, v_j) is not excluded or marked.

Этап 1. Устанавливается i = 2, j = 3. {1, 2, 3}

Step 2. One of the arcs (v_2, v_3) is excluded

Этап 1. Устанавливается i = 3, j = 10. p_1 = {1, 2, 3, 10}

Этап 2. Дуга (v_3, v_{I0}) исключается.

Этап 1. Устанавливается i = 3, j = 6.

Этап 2. Дуги (v 6, v 7), (v 8, v 9), (v 9, v I0) исключаются, дуга h(v 3 , v6)

Notes.

The procedures of steps 1 through 2 are repeated until the path to the final vertex of graph v_{10} corresponding to the result of the calculations is determined. In this case, the first path p_1 = { l, 2, 3, 10} is determined after three steps. The following steps, repeated until there are no arcs of type g and h left in the constructed path, allows you to determine the following paths:

p 2 = {1,2, 3, 6, 7, 8, 9, 10},

p_3 = {1. 2, 3, 4, 6, 7, 8, 9, 10},

p 4 = {1. 2, 3,4, 6. 7, 8, 9, 10},

p_5 = {1, 2, 3, 4, 5, 10}.

According to the developed algorithm of the minimum coverage of the program graph, a program in Python is compiled, which is shown in Table 1.

The program consists of the following parts:

- enter the original adjacency matrix;
- determination of the degree of yield of vertices;
- determine the degree of entry of vertices;
- create a specific path;
- Excluding a g or h doop in a route of a specific path.

To create one path in the worst case requires n operation, and to build the minimum number of operations requires m operations, where m is the minimum number of paths that cover all the branches of the graph of the program. Therefore, the complexity of the developed algorithm is equal to

$$O(|v| x |m|) => O(|(v|)$$

Table 1 Resource requirements by component

Program Minimum Graph Coverage Program	Extension of the program

# Enter the original adjacency matrix 	
import numpy as np
a=np.array([[0,1,0,0,0,0,0,0,0,0],
[0,0,1,0,0,0,0,0,0,0],
[0,0,0,1,0,0,0,0,0,0],
[0,0,0,0,1,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,1],
[0,0,0,0,0,0,1,0,0,0],
[0,0,0,0,0,0,0,1,0,0],
[0,0,0,0,0,0,0,0,1,0],
[0,0,0,0,0,0,0,0,0,1],
[0,0,0,0,0,0,0,0,0,0]])
print(a)
vx=[0,0,0,0,0,0,0,0,0,0]
vix=[0,0,0,0,0,0,0,0,0,0]
 	
# Determining the yield of vertices 	
for i in range(0,10):
for j in range(0,10):
vix[i]=vix[i]+a[i,j]
print(vix)
 	
# Determining the degree of input xxxxxxxx 	
for i in range(0,10):
for j in range(0,10):
vx[i]=vx[i]+a[j,i]
print(vx)
 	
# creating paths xxxxxxxxxxxxxxx 	
p=[0,0,0,0,0,0,0,0,0,0,0,0]
# creating paths xxxxxxxxxxxxxxx 	
ii=[0,0,0,0,0,0,0,0,0,0,0]
jj=[0,0,0,0,0,0,0,0,0,0,0]
ia=[0,0,0,0,0,0,0,0,0,0,0]
and=[0,0,0,0,0,0,0,0,0,0,0]
w=[0,0,0,0,0,0,0,0,0,0,0]
z=[0,0,0,0,0,0,0,0,0,0,0]
a[i,j]=0
k=0
for i in range(0,10):
for j in range(9,0,-1):
if a[i,j]>0:
ii[k]=i
jj[k]=j
k=k+1
	#xxxxxxxxxxx education is a setof pathways
	d=0
	q=0
	k=0
	p[0]=1
	for i in range(10,0,-1):
	for j in range(10,0,-1):
	ia[j]=ii[j] # трансформация
	ja[j]=jj[j] # трансформация
	for i in range(0,10):
	d=ja[i]
	#print('d,d)
	p[k]=9
	if d==9:
	break
	for j in range(0,10):
	if ia[j]==d:
	p[k]=d
	print('put ',p[k])
	print('put ',p)
	k=k+1
	if ia[j]==d or d==9:
	break
	if ia[j]==d or d==9:
	break
	p[0]=1
	
	# exclusion of arcs of type "g" and "h" xxxxxx
	g=0
	h=0
	for i in range(0,10):
	s=p[i]
	t=p[i+1]
	print('s t ',s,t)
	if(vix[s]>1 and vx[t]>1):
	a[s,t]=a[s,t]-1
	print('искл. ',a[s,t])
	p[i]=j
	g=1
	elif(vix[s]>1 and vx[t]==1)and g==0:
	a[s,t]=a[s,t]-1
	print('xxx ',s,t)
	if t==9:
	break
	if t==9:
	break
	#print('put ',p)
	print(a)

References

1. Iyulu K.A., Aripov M.M. Automation of the generation of ways for testingprograms written x on fortran. Programming, 1986, No7.
2. Ijudu K.A., Aripov M.M. Testing of the program based on
3. Control Systems and Machines, 1985, No. 6.
4. Ijudu K.A., Aripov M.M. Automation of structural testing of programs. Republican Conference. Reliability and quality of software. Abstracts. Lviv, January 29-31, 1985
5. Kulikov S. A. Software testing. Minsk, 2020.
6. Fundamentals creating the algebra science and algoritms / M.M.Aripov,
7. R.N.Normatov, I.M.Siddikov, U.Oripova // Solid state technology. 2020.
8. Vol.63.No. 5. 6103-6111.
9. Simon C., Ntafos S., Louis Hakimi. On structured digrafs and program testing. IEEE Trans. On Computers, vol. C-30, № 1, January 1981.
10. Shukhratovich, Shirinov Feruzjon. "The Field of Computer Graphics and Its Importance, Role and Place in The Information Society." Texas Journal of Multidisciplinary Studies 4 (2022): 86-88.
11. Tokhirovna, Khakimova Yoqutkhon. "Stages Of Implementation Of Distance Learning In Higher Education." Texas Journal of Philology, Culture and History 1 (2021): 38-39.
12. Хонбобоев, Хакимжон Октамович, Мубина Хакимжоновна Икромова, and Мухаммад-Анасхон Хакимжонович Икромов. "Ta'limda axborot texnologiyalarni qollashning oziga xos xususiyatlari." Молодой ученый 3-1 (2016): 21-22.
13. Хонбобоев, Хакимжон Октамович, Фозилжон Усибхонович Полатов, and Мухаммад-Анасхон Хакимжонович Икромов. "Tasviriy san'atni oqitishda interfaol metodlardan foydalanish." Молодой ученый 3-1 (2016): 22-23.
14. Khaidarova, S. "Sql-expressions That Manage Transactions." JournalNX: 307-310.
15. Siddiqov, I. M. "THE IMPORTANCE OF USING THE ACT IN THE PROCESS OF DEVELOPMENT OF PRESCHOOL CHILDREN." Экономика и социум 5-1 (2021): 458-461.
16. Muydinovich, Rasulov Inom. "The Role of Digital Technologies in Growing Secondary School Students to the Profession." Eurasian Scientific Herald 6 (2022): 137-142.
17. Muydinovich, Rasulov Inom. "The Role of Digital Technologies in Growing Secondary School Students to the Profession." Eurasian Scientific Herald 6 (2022): 137-142.
18. Normatov, R. N., M. M. Aripov, and I. M. Siddikov. "Some issues of analysis structural complex systems." International Journal on Orange Technologies 3.2 (2021): 59-62.
19. Juraev, M. M. (2022). Prospects for the development of professional training of students of professional educational institutions using electronic educational resources in the environment of digital transformation. Academicia Globe: Inderscience Research, 3(10), 158-162.
20. Toshpulatov, Raximjon I. "MODERN METHODS AND TENDENCIES IN TEACHING INFORMATION TECHNOLOGY." International Journal of Pedagogics 2.09 (2022): 43-46.
21. Yuldoshev, Utkir, and Uktamjon Zhumankuziev. "Determination of the leading pedagogical laws and fundamental principles of the formation of the information culture of school-age children." Society and Innovation 2.5/S (2021): 68-76.
22. Marufovich, Aripov Masud, and Shirinov Feruzjon Shuxratovich. "DEVELOPING THE COMPETENCE OF FUTURE INFORMATICS TEACHERS TO WORK WITH GRAPHICAL INFORMATION." ONLINE SCIENTIFIC JOURNAL OF EDUCATION AND DEVELOPMENT ANALYSIS 2.1 (2022): 183-187.

23. Siddikov, I. M., and Sheraliev O. Sh. "ABOUT ONE INNOVATION METHOD OF LOCALIZATION OF INDEPENDENT DIGITAL DEVICES." E-Conference Globe. 2021.
24. Shirinov F., Mamasoliyev A. A GENERAL DESCRIPTION OF THE HARDWARE AND SOFTWARE ENVIRONMENT USED TO ORGANIZE COMPUTER-BASED LEARNING PROCESSES //Euro-Asia Conferences. – 2021. – Т. 3. – №. 1. – С. 63-65.
25. Normatov, R. N., M. M. Aripov, and I. M. Siddikov. "Analysis Method of Structural-complex System Indicators by Decomposition Into Subsystems." JournalNX 7.04 (2021): 68-71.
26. Marufovich, Aripov Masud, and Shirinov Feruzjon Shuxratovich. "DEVELOPING THE COMPETENCE OF FUTURE INFORMATICS TEACHERS TO WORK WITH GRAPHICAL INFORMATION." ONLINE SCIENTIFIC JOURNAL OF EDUCATION AND DEVELOPMENT ANALYSIS 2.1 (2022): 183-187.