The Ministry of Transport and Communications of Ukraine

The State Administration of Communication

ODESSA NATIONAL ACADEMY of TELECOMMUNICATIONS named after A.S. POPOV

Chair of Communication Networks

Nikityuk L.A., Shulakova E.S.

Synthesis and Analysis Elements of Telecommunication Networks

Methodical instructions and the complex task
for students of all educational forms on discipline
Telecommunication Networks

APPROVED BY
Faculty senate of
Telecommunication Networks
Protocol number 6
on March 30, 2010

Odessa, 2010

UDC 621.391 Plan SMP 2010

Reviewers: G.S. Gayvoronskaya,
O.O. Varaksin

The compilers: Nikityuk L.A., Shulakova E.S.

Synthesis and analysis elements of telecommunication networks: methodical instructions and the complex task for students of all educational forms / Nikityuk L.A., Shulakova E.S. – Odessa: ONAT named after A.S. Popov, 2010. – 50 p.

The key positions and methodical instructions for the complex task performing on discipline «Telecommunication networks» are outlined. Applied problems of synthesis and the analysis of telecommunication networks and the requirement for their solution are formulated. The variants of individual tasks are given.

ACCEPTED
After Meeting Chair of
Telecommunication networks
and recommended for printing
Protocol № 3
on October 8, 2010

CONTENTS

1 General conditions 4
2 Key positions 5
2.1 General concept about problems of telecommunication networks synthesis and analysis 5
2.2 Model representation of telecommunication networks as object of synthesis and analysis 7
2.3 Elements of optimization theory on graphs and networks 10
2.3.1 Synthesis of the minimal cost network 10
2.3.2 Determination of a graph median 14
2.3.3 Determination of the graph center 14
2.3.4 Determination of the minimal length cycle 15
2.3.5 Finding the shortest path in the connecting network 16
2.3.6 Determination of the given transient path sets 19
2.3.7 Problems about flow 21
3 The complex task 25
3.1 Construction of telecommunication network models 25
3.2 Synthesis of subscriber access network 25
3.3 Synthesis of the internodal network 27
3.4 Construction of the routing matrixes 27
3.5 Estimation of network capacity between points pair 29
4 The list of literature 29
Appendix A 30
Appendix B 47

1 GENERAL CONDITIONS

The complex task includes material containing the general principles about construction of telecommunication networks as synthesis and analysis objects.

Practice of maintenance and modernization designing (reconstruction and development) of telecommunication networks brings various problems solution of which assumes their formalization in terms of mathematical models of networks synthesis and analysis and selection of adequate methods for the solution among all multitude of existing methods.

The purpose of this complex task is acquaintance with elements of a mathematical apparatus of networks synthesis and analysis and obtaining practical skills in the solving specific problems appearing at their designing and maintenance.

To perform the complex task it is necessary to study chapters «Key positions» and «The complex task» of the present methodical instructions and to choose the individual variant of the initial data defined by a number composed from two last figures of the student's card (see *Appendix A*).

The calculations, executed according to the task, together with problems setting, the general explanations of their solution principles and illustrating material are necessary to form as an explanatory note with the title page (the form of title page is introduced in *Appendix B*).

The explanatory note should contain introduction and the chapters corresponding to a number of tasks. One should introduce such positions as telecommunication networks dedication and its general characteristic, structural organization principles, the basic components and networks segments, general characteristic of synthesis problems and communication networks analysis and etc. in introduction.

2 KEY POSITIONS

2.1 The general concept about problems of telecommunication networks synthesis and analysis

All problems appearing by telecommunication networks construction and maintenance can be divided into two classes: problems of synthesis and problems of analysis.

«Synthesis» according to the translation from Greek means «union, composition».

The problem of network synthesis arises at a new network constructing, and at existing networks reconstructing and developing. This problem has technical and economic nature because more often the solution, optimal for a number of economic parameters such as minimum capital investment, is found.

At network synthesis location of network points is usually assumed as given one. The communication lines configuration (topology) can vary at economic parameters optimization. It allows to use expenses for communication lines as a target criterion of the optimum network synthesis. The restrictions on exceptional separate geographical routes in the organization of the communication between points, for example, if they cross the water or mountain obstacles can be made on the lines configuration.

The problems of choosing optimal network topology, optimal number and switching nodes location etc. can refer to *the particular problems of synthesis*.

Analysis problems are topical for existing (the synthesized) network. These problems include the problems of a finding optimal ways for the information messages transmission, set ways with a given transient determination, network capacity estimations, probability of the path connection between points determination etc.

Also the questions of calculating characteristics and parameters as networks on the whole, and also its separate elements are considered in the class of analysis problems. The quality of service on the network, reliability and survivability parameters refer to such characteristics.

To solve a specific problem of telecommunication network synthesis or analysis, it is necessary **to formalize** this problem, i.e. to write down it in the form of scheme: what is given, what is necessary to define and under which restrictions.

Formalization can be executed in verbal form (such form is called verbal model of a problem) or in the form of the mathematical model presenting a problem in terms of this or that theory (for example, graph theory, theories of optimal decisions, etc.)

Formalization implementing requires not only the given problem understanding, but also choosing the matching model of the object (telecommunication network). The modeling (simplified) representation of synthesis or analysis object allows to identify and to represent the most significant elements of object and connection between them from the point of view of existing problem, without distracting on details.

For modeling representation of telecommunication networks graph models are mostly used. The mathematical model representing dependence between required parameters and independent variables in problem in the form of mathematical functions can be built on the basis of object model and its parameters (quantity of points and network lines, distances between points, nodes capacity and network lines, cost parameters, etc.).

Mathematical models of optimization where the purpose of problem solution is written in the form of so-called **criterion function** for which it is necessary to find **an extremum (**minimum or maximum) are used more often in the problems of telecommunication networks synthesis and analysis. The imposed limits, specifying in which borders required parameters values to be changed, can be made on the incoming parameters.

D e f i n i t i o n. *The problems*, *where the extremum* (*the minimum or maximum*) *of some criterion function*, *representing an optimization of the problem solution*, *is found*, *are called* ***extreme***.

Specific feature of extreme problems of telecommunication networks synthesis and analysis is their big dimensions. The formulation of these problems in terms of graph and network models allowed us to obtain a series of effective, from the point of view of computing complexity overcoming, methods and algorithms for their solving oriented to computer applying. A series of such algorithms are considered below.

The algorithm is understood as the procedure, providing the optimal solution of a problem, performance of which can be entrusted to the computer. They differ such algorithms as **the exact** and approximate, so-called **heuristic**.

Exact algorithms always guarantee finding the optimal solution (criterion function global optimum). For example, the algorithm of exhaustive search of all possible solutions with choosing the best of them, is exact algorithm.

However, the exact algorithms, as a rule, are quite time-consuming from the computation point of view. Therefore, in practice, they often use more simple algorithms providing quick solutions obtaining with acceptable accuracy for practice. Such algorithms are constructed using rational, from the point of view of human logic, *rules of calculations performing*. These rules are called **heuristics** and, as practice shows, allow to obtain the solution, close to the optimal one. For example, the problem of determining the closed contour with a minimal length, which provides by-path of all network points, can be solved by complete searching of all possible contours with choosing among them the minimal length contour, i.e. exact algorithm. It is known, that for a network containing n points, the quantity of possible contours is $n!$, obtaining of which for the network of $n>30$ size, represents considerable difficulties. However, heuristics usage: «*on each step one moves only to the nearest point*» provides reception of the acceptable solution for a time necessary for only one contour constructing.

Heuristic algorithms are used also in those cases when it is impossible to build an exact algorithm because of the mathematical model complexity of the problem (its nonlinearity, discreteness and etc.).

2.2 Model representation of a telecommunication networks as object of synthesis and analysis

The communication network (telecommunication network) as object of synthesis and analysis is represented as a set of network points and lines connecting them. As a mathematical model of such object *the graph* is used.

D e f i n i t i o n *Some set of points and arrows connecting them is called* ***a graph***.

Graph *points* are called **vertices**, and *arrows* – **arcs**. Mathematically the graph is marked out as $G(N, V)$, where N - final set of vertices, with n power, and V - final set of arcs with m power.

Vertices can be marked with lower – case letters (i, j, k, l, s), or digits (1, 2, 3, 4, 5), and arcs - accordingly by pairs: $\{(i, j), (j, k), (k, l)...\}$, or {(1,2), (2,3), (3,4)...} where the first index defines the beginning, and second – the arc end.

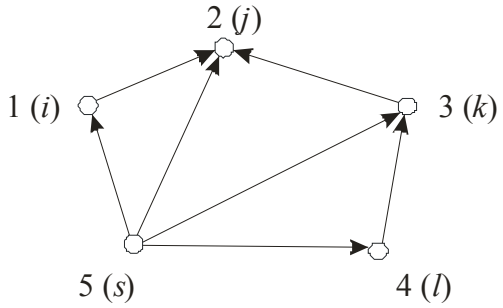


Figure 2.1 – Graph model of network

The graph, where the arcs direction is set, is called **oriented**, otherwise – **undirected**. Non-directional arcs are called **as edges**.

Between two vertexes connected by an arc (edge) there is **a adjacency** relation (for oriented graph vertices i and j are adjacent, only if the arc begins in i and is directed to j).

Between an vertex and arcs (edges) connected with it there is a **incidence** relation.

The graph, each arc (edge) of which is assigned in the certain numerical characteristics, are called **weights**, is represented as **weighted** graph. If it is necessary, weights can be assigned to graph vertices.

Weighed graph is usually called **a network** (in this case it refers to the network model, but not the network itself as object). Distance, capacity, cost and etc. can serve as a weight network characteristics.

Besides the geometrical image in the form of points and lines, the graph can be presented in a digital form. Exactly this form is used at entering graph model into the computer.

One of the most widespread discrete representations of the graph is **the adjacent matrix**. It is a matrix $A = [a_{ij}]$, of $(n \times n)$ size elements which can possess such values:

a_{ij} = 1 if there is an arc (edge) between vertices *i* and *j* in graph G;
a_{ij} = 0, - in the other case.

The adjacent matrix of the graph, showed on fig. 2.1, looks like:

$$A = \begin{vmatrix} 0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 1\,1\,1\,1\,0 \end{vmatrix}$$

For storing in a computer memory adjacent matrix, as it is seen, we need n^2 cells.

In the non-directional graph the adjacent matrix is symmetric relatively to the main diagonal, and, hence, in the computer memory only one of its triangles can be stored that will allow to save memory, but it complicates its processing in the computer.

If to renumber arcs (edges) of graph *G* in a random order and to put these numbers in accordance with the lines numbers of some matrix $B = [b_{ij}]$, and columns numbers will be the same matching to the graph vertices numbers, then it is possible to represent the **incidence** relation of graph *G* elements in such matrix. B_{ij} matrix elements can posses {0,1} values.

Let's renumber arcs for the considered graph: (*i*, *j*) – 1; (*j*, *k*) – 2; (*k*, *l*) – 3; (*l*, *s*) – 4; (*s*, *i*) – 5; (*s*, *j*) – 6; (*s*, *k*) – 7.

Then the incidence matrix will look like:

$$B = \begin{vmatrix} 1\,1\,0\,0\,0 \\ 0\,1\,1\,0\,0 \\ 0\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1 \\ 1\,0\,0\,0\,1 \\ 0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,1 \end{vmatrix}$$

The weighed graph (network) can have a discrete form presented as **a weighted matrix** $W = [w_{ij}]$, where w_{ij} – arc (edge) weight if it exists in graph G. Nonexistent arcs (edges) weights consider to be equal to ∞ or 0 depending on problem statements where they are considered.

If the graph is disperse (has less arcs (edges) quantity) then it is possible more compact representation of graph G – by the list of arcs (edges). This list can be realized with two one-dimensional arrays (R_1 and R_2) with dimension m, in the first of which initial vertices of arcs (edges), and in the second – final one are written down, or with a two-dimensional array R with dimension $(2 \times m)$. For example,

$$R_1 = (1, 3, 4, 5, 5, 2, 5);$$
$$R_2 = (2, 2, 3, 4, 1, 5, 3)$$

$$R = \begin{vmatrix} 1\ 3\ 4\ 5\ 5\ 2\ 5 \\ 2\ 2\ 3\ 4\ 1\ 5\ 3 \end{vmatrix}$$

In organizing graph representation in the form of a discrete array with floating boundaries, i.e. in the case, when one needs to forsee the possibility of adding or removing graph vertices, it is expedient to use **adjacency structure**. The last one represents the list of adjacent vertices for each graph vertex. The **adjacency structure** for the graph, represented on figure 2.1, looks like:

1: 2
2: 5
3: 2
4: 3
5: 1, 2, 3, 4

2.3 Elements of optimization theory on graphs and networks

2.3.1 Synthesis of the minimal cost network

The situation, where some set of points is necessary to connect so, that each pair of points to be connected (directly or via other points), and the total weight

characteristic of connections was minimal, generates a problem **of synthesis of the minimum cost network**.

For example, there is a sequence of points, where points of telecommunication network can be placed. Distances between pairs of points and cost of the foundation for communication line of one kilometer are known. It is necessary to define the set of communication lines that provides connectivity of all network points and its minimal cost.

From graph and networks theory it is known, that solution of the given task is the network with topology of «*tree*» type.

D e f i n i t i o n. *The connected graph* (*connecting network*) *is called* ***a tree*** *if there are no cycles in it*.

They say that the graph contains **cycles** if it is possible to find the closed contours in it. Cycles absence defines the feature of the tree type graph which consists in that between any pair of its vertices there is only one path connecting them, i.e. connectivity parameter h=1. The quantity of edges in a tree is always one unit less than the quantity of its vertices.

D e f i n i t i o n. *The tree which includes all vertices is called* ***covering***.

Mathematically the problem of network synthesis of the minimum cost is formulated this way.

Let non-directional graph G (N, V) be given, where the set of N vertices corresponds to the set of network points, general number of which is equal to n, and set of V edges - $\{l_{ij}\}$ distances between pairs of points. The organizations cost c_{ij} for communication line of 1 kilometer between points i and j is known.

It is necessary to find some covering $G'(N, V')$ tree for which the criterion function minimum is attained:

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} l_{ij} \to \min$$

For this task solving there is a sequence of effective algorithms. We will bring one of them, which is known as Prim. The algorithm is implemented by

assignment of marks to vertices, which are input into graph $G'(N, V')$, and consequential introduction in it, the shortest edges, total quantity of which should not exceed (n-1) and provide connectivity between all n vertices of a covering tree.

The step-by-step form of Prim algorithm has the following form:

Step 0. The initial network $G'(N, V')$ in a stowed position contains n vertices and does not contain edges. One random vertex i is chosen and marked as «*selected*». The other (n-*1*) vertices are marked as «*non-selected*».

Step 1. The edge (i, j) belonging to G (N, V) with the minimal weight, which vertex i belongs to a subset of « *selected* » vertices, and vertex j - to a subset of «*non-selected* » vertices is found.

Step 2. The edge (i, j) is included into the initial network $G'(N, V')$, and the vertex j is excluded from a subset of « *non-selected* » vertices and is included into a subset of «*selected*» vertices. If the subset of « *non-selected* » vertices is empty it means the end of algorithm operation. Otherwise – going back to the step 1.

Let's illustrate the Prim algorithm operation on the instance. Let there be seven network points, the distances between which are implemented into matrix $L = [l_{ij}]$, namely:

$$L = \begin{vmatrix} 0 & 10 & 5 & 12 & 9 & 3 & 9 \\ 10 & 0 & 7 & 2 & 8 & 4 & 6 \\ 5 & 7 & 0 & 3 & 1 & 5 & 11 \\ 12 & 2 & 3 & 0 & 10 & 15 & 10 \\ 9 & 8 & 1 & 10 & 0 & 12 & 9 \\ 3 & 4 & 5 & 15 & 12 & 0 & 17 \\ 9 & 6 & 11 & 10 & 9 & 17 & 0 \end{vmatrix}$$

On a step 0 the initial graph $G'(N, V')$ contains 6 vertices and does not contain edges. Let's choose vertex 3 and mark it as «*selected*» (see figure 2.2).

On a step 1 let's choose the edge (l_{35}) as an edge with a minimal weight, which vertex i=3 belongs to a subset of t «*selected*» vertices (still it contains only one vertex 3), and the vertex j=5 - to a subset of «*non-selected*» vertices (now these are all other vertices). On a step 2 edge l_{35} is input into the initial graph G', and the vertex 5 is included into a subset of « *selected* » vertices (figure 2.3).

As a subset of « *non-selected*» vertices is not empty, we repeat step 1. For this purpose we find the edge of the minimum weight, overhauling combinations of each pair *of* «*selected* » and « *non-selected* » vertices. Thus there was an edge l_{34} (figure 2.4). It is input into the graph G', and the vertex 4 becomes «*selected*».

The following edges to be chosen are: l_{26} (figure 2.6); l_{31} (figure 2.7); l_{27} (figure 2.8). Here the algorithm operation comes to an end, because all vertices are

marked as «*selected*» (i.e. the subset of «*non-selected*» vertices are an empty subset).

The graph $G'(N, V')$, representing covering tree, as it includes all vertices, contains the number of edges one unit lower than the numbers of vertices (n=7, v=6) and provides connectivity of each pair of vertices is obtained.

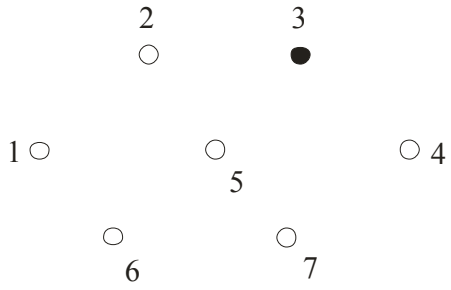


Figure 2.2

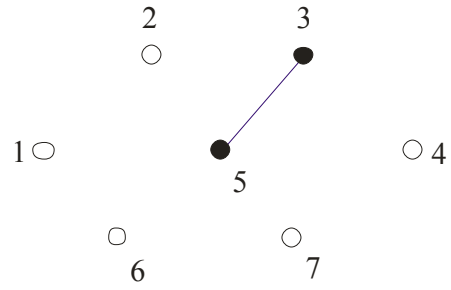


Figure 2.3

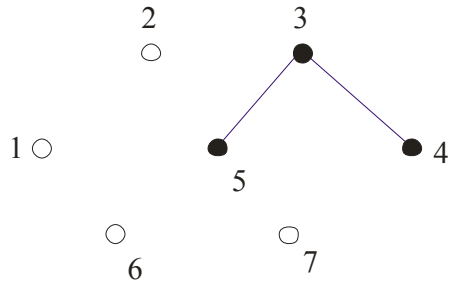


Figure 2.4

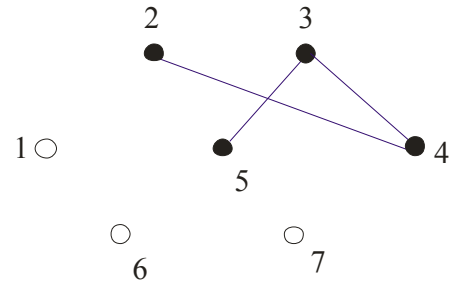


Figure 2.5

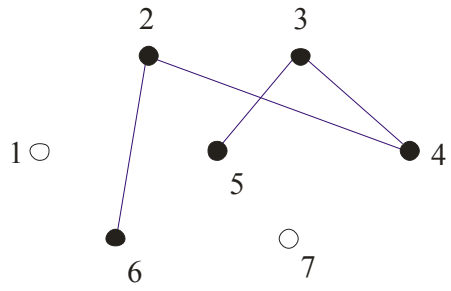


Figure 2.6

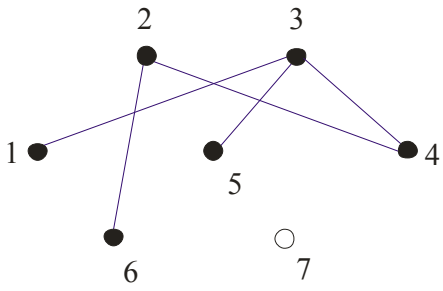


Figure 2.7

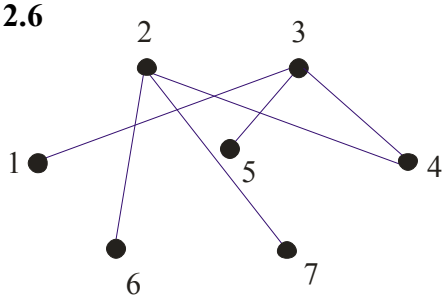


Figure 2.8

2.3.2 Determination of a graph median

Let's consider the following problem. Let the graph G (N, V) represent some cable network, connecting n subscriber points. The weight of each edge (i, j) belonging to V corresponds to the length l_{ij} or to the cable costs connecting points i and j. It is necessary to define some vertex m belonging to N where it is expedient to place a switching node (for example, regional automatic telephone exchange) from the point of view of minimization in total length of the cable connecting subscriber points with switching node.

Solution of the given task is determination of the graph G (N, V) median.

D e f i n i t i o n. *The vertex m, belonging to N, is* ***a graph G (N, V) median*** *if it satisfies a condition*:

$$\sum_{j=1}^{n} l_{mj} \le \sum l_{kj}; k \ne m$$

Value $R_m = \sum_{j=1}^{n} l_{mj}$ is called ***median length*** of graph G and represents the minimal total length of the edges connecting vertex m with other graph nodes.

The algorithm of graph G median definition includes following steps.

Step 1. To find the sum of elements for every line in the initial matrix of weights $L = [l_{ij}]$, corresponding to edges lengths:

$$R_i = \sum_{j=1}^{n} l_{ij} \; ; \; \forall i \in N$$

Step 2. To find minimal R_m. The vertex m also is a graph G median among set of values $\{R_i\}$.

2.3.3 Determination of the graph center

Let's assume that the location of a subscriber network where they introduce the stationary radio access to the reference node of a base network is given. It is

necessary to define among subscriber network points the location of base station (BS), which connects with subscriber points (SP) through radio channels. It is desirable to have minimal distance from BS to any SP that will provide a stable radio communication with smaller power of BS transmitter. It is clear that such criterion is impossible to satisfy therefore it is necessary to minimize the distance to the most remote point. It is thus expedient the BS whenever possibly to occupy the central position under the relation to all RP.

The problem of finding such point can be turn to a problem of **the graph centre finding.**

D e f i n i t i o n. *Let G* (*N*, *V*) *be a graph*, *where N – set of vertices*, *and V – set of distances between all vertices*.

The vertex s is called ***the graph G* (*N*, *V*) *centre*** *if it satisfies to a condition* $\max l_{sj}\ m \le \max l_{ij}$ for any i; $1 \le j \le n$.

The algorithm of finding graph centre (vertex *s*) follows from the definition:

Step 1. In each line of an initial weighted matrix $L = [l_{ij}]$ – find element with the maximum value.

Step 2. Among set of the maximum values of lines elements find the least $l_{sj}\ \{\in l_{ij}\}$. The vertex *s* is the graph centre.

Thus, having minimized the distance from a point *s* to the most remote vertex, we provided to all other vertices guaranteed smaller distance.

2.3.4 Determination of the minimal length cycle

This problem is known as «The problem about the travelling salesman». Let the graph $G(N,V)$ is given, which vertices correspond to cities in the service zone of the travelling salesman, and arcs represent connections between pairs of cities. The route of the travelling salesman is called the contour which includes each node of a graph *G*.

D e f i n i t i o n. *Contour including each vertex of graph G* (*N*, *V*) *only once is called* ***Hamiltonian contour*** (*or Hamiltonian cycle*)

The term Hamiltonian cycle is given by name of the Irish mathematician William Hamilton who began studying these problems for the first time in 1859.

The travelling-salesman problem is called the problem of search of the traveling salesman route of the least length. The optimum solution of this problem is the Hamiltonian cycle of the least length. The problem can be solved with the following exact method.

Let's renumber n cities as integers from 1 to n. The base city will posses number n. We will notice that the travelling-salesman round tour corresponds to the shift of integers 1, 2..., (n-1). Thus, the base city at number n constantly takes last position and does not participate in shift processes. Each shift can take conformity some number defining length traveling-salesman route as the sum of ribs lengths of a cycle, connecting all n graph nodes.

Having formed all shifts from (n-1) numbers and gained routes lengths, which quantity is defined as (n-1)!, it is easy to find the route of the least length.

The approximate *heuristic* algorithm can be obtained with the use of heuristics, for example «on each next step the nearest city is included into the cycle».

Definition of Hamiltonian cycle of the least length is immediate while defining optimum ring topology of telecommunication networks segments.

2.3.5 Finding the shortest way in a connecting network

The problem about finding the shortest path length in a connecting network refers to the fundamental combinatory optimization problems. It is possible to reduce a wide range of the practical problems originating in various areas of a national economy, and first of all in communication.

D e f i n i t i o n. ***The path*** *is the sequence of vertices* $\mu_{ir} = (i, j, \ldots, r)$ *or sequence of arches* (*ribs*) $\mu_{ir} = \{(i, j), \ldots (to, r)\}$, *connecting pair of vertices i and r of graph G.*

The sum of the weights attributed to arches (ribs) in path μ_{ir} defines **a path length.**

The way from an vertex i into an vertex r possessing minimum possible length is called **the shortest way**.

The network is called **connecting** if there is at least one way connecting them for each pair of vertices.

Being based on network model, the problem about finding the shortest way can be formulated by the following way.

Let connecting network G be given where every arch (rib) has the positive weight that is proportional to its length. It is required to find a way μ_{st} between the set vertices s and t, having minimum possible length, i.e.

$$L = \sum_{(i,j)\in\mu_{st}} l_{ij} \rightarrow min \text{ (on } M\text{)},$$

where M – a set of all possible ways from s to t.

One of the most effective algorithms solving the given task is Dijkstra algorithm having the author's name.

This algorithm feature is the fact that during its performance the shortest ways are simultaneously under construction from the set vertex s into all other network vertices. It can be explained by the fact that any vertex $i \in N$ can appear intermediate in the shortest way from s in t. At the end of algorithm operation the vertex s appears connecting with all other network G vertices as well as with an vertex t, the shortest ways, and the arches (rib) having entered into them, form some sub-network without cycles, i.e. a tree with a root in an vertex s.

Algorithm operation is introduced by means of arrangement at vertices aspect (L_{sj}, i) marks, where L_{sj} - length of the shortest way from an initial vertex s into some vertex j, and i preceding j vertex in this path.

Marks are divided into *time* and *constant*. Time marks can change as a result of algorithm operation, and the constants – do not change.

Dijkstra algorithm in the step-by-step form is given below.

Step 0. For vertex s $L_{ss} = 0$ *is considered*, and for other vertices $L_{sj} = \infty$. All vertices have time marks of an aspect (L_{sj}, s).

Step 1. Among vertices with time marks we choose vertex r, for which $L_{sr}=min\ (L_{sj})$. The apex mark r becomes *constant*.

Step 2. If all vertices of a network obtained *constant* marks it means the end of algorithm operation. Otherwise - transition to a step 3.

Step 3. We enumerate *time* marks for vertices contiguous to vertex obtained a constant mark on a step 1, according to expression $L_{sj}=\min\ (L_{sj}, L_{sr}+L_{rj})$.

Transition to a step 1.

Path trace μ_{st} is made in the inverse direction following from an vertex t in s, being guided by vertices i in *constant* marks.

Let's illustrate the Dijkstra's algorithm operation on an instance.

Let's find the shortest way from the vertex s to the vertex t in a network represented on fig. 2.9. The weights put down near ribs define their length.

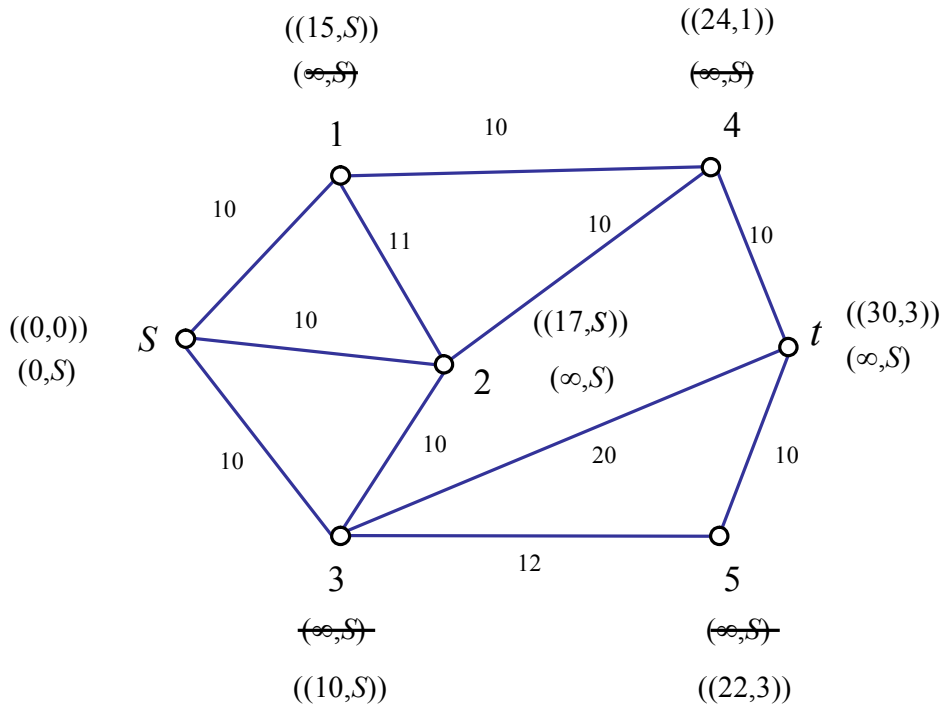


Figure 2.9 – Illustration work of algorithm of Dijkstra

Step 0. Mark P for vertex s looks like: $P_s=(0,0)$. For other vertices $P_i=(\infty, s)$. All marks are *time ones*.

Step 1. Among *time* marks the vertex s has the least length parameter as $L_{ss}=0$. Its mark becomes *a constant* one (we will note it with double brackets)

Step 2. We convert *time* marks for vertices contiguous to vertex *s*. For the vertex 1 the parameter length

$$L_{s1}=L_{ss}+l_{s1}=0+15=15$$

The obtained value is less than available one($L_{si}=\infty$) and consequently a new value *of a time* mark will be $P_1 = (15, s)$.

For the vertex 2 a new mark has the value $P_2 = (17, s)$. For the vertex 3 – $P_3 = (10, s)$.

Passing to a step 1, we choose the vertex 3 as it has the least length parameter $L_{s3}=10$, among all vertices with *time* marks. Its mark becomes *a constant one*. As not all vertices have obtained *constant* marks we pass to a step 2 and we carry out the re-count of marks for vertice, contiguous to an vertex 3:

$P_2 = (17, s)$ can de left without changing as the new length parameter is equal to former value.

$P_5 = (22, 3)$.

$P_t = (30, 3)$.

The vertex 1 on a step 1 obtains *a constant* mark as its length parameter is minimum.

New value of a mark on a step 2 is obtained by the vertex 4, namely $P_4 = (24, 1)$.

Coming back to a step 1, we set *a constant* mark for vertex 5. To change the value of marks on a step 3 is impossible. We fix the following vertex with *a constant* mark it is the vertex 4.

To change *a time* mark for the vertex *t* is impossible – and it automatically becomes *a constant one*.

Here algorithm operation comes to an end.

We define the way trace μ_{st} moving in the opposite direction from *t* to *s* through the vertices specified in marks: $t \to 3 \to s$.

2.3.6 Determination of the given transient path sets

Among the restrictions imposed while setting connecting sections of transfer in communication networks, restriction on number of transit points or transit sections in them can be considered.

Transit points are regarded as switching knots appearing along the messages travelling from some subscriber's point *i* in *j* where there is messages streams reallocating. Transit sections accordingly represent communication line connecting transit points.

Restriction on transience along the messages transmitting is caused by demands to a network service quality (for example, to the time of message transit in a network, to the time of message processing in switching knots etc.).

In terms of graph models the problem is formulated as follows.

Let some initial graph G be given (N, V), in accordance to a set N power n which vertices put switching knots of a communication network, and to set V - network junction circuits.

It is necessary to define a set of ways $M = \{\mu_{si}\}$ from the given vertex s in other vertices $i \in N$, $i \neq s$, $i = 1 \ldots n$, graph G for which the transient parameter T does not exceed some specified value T_0, i.e.

$$T \leq T_0, \forall \mu_{si}, i \neq s, i = 1 \ldots n.$$

One of the most convenient, easily implemented on the computer methods of paths definition meeting this demand is constructing so-called «**multilevel tree**» paths from some defined vertex s to other graph vertex.

There is the initial graph and corresponding to it «multilevel tree» with parameter $T_0 = 2$ on fig. 2.10. Here T is regarded as the quantity of transit vertices.

The algorithm of «multilevel tree» constructing includes the following steps.

Step 0. To form a subset of a zero tier including a unique element-top s into it. Using a contiguity matrix to write out columns numbers in a line with number S which elements are equal to $a_{sj} = 1$. Thus, there is the first tier vertices subset, formed by an vertex s.

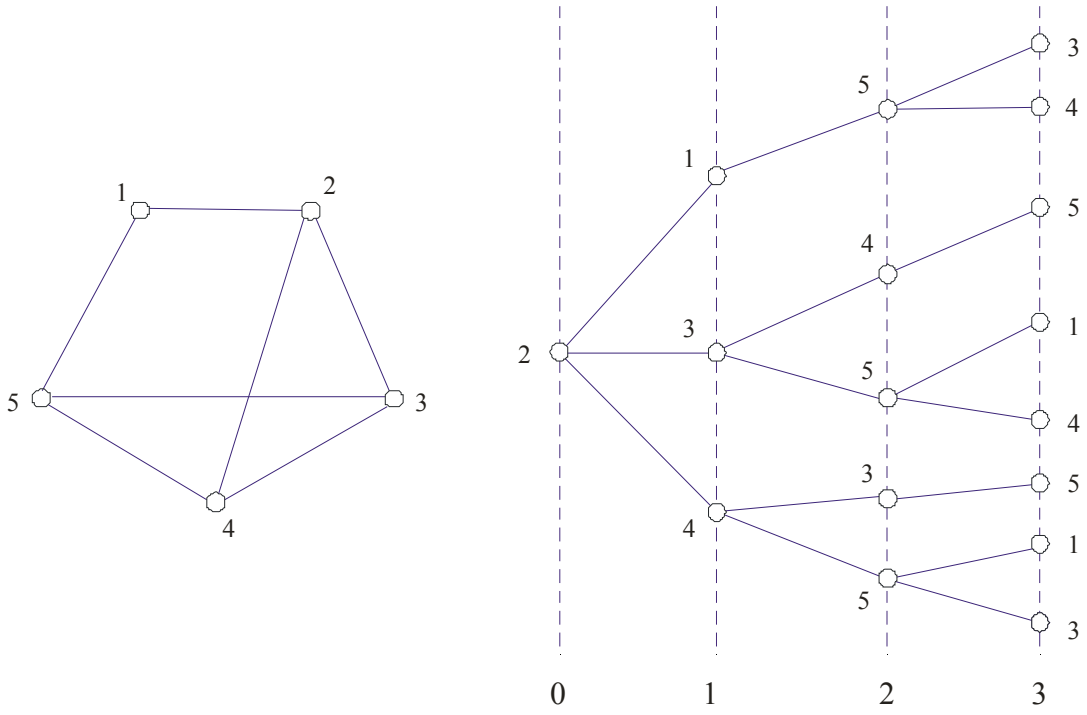


Figure 2.10 – Illustration work of algorithm «multilevel tree» construction

Step 1. To form a subset of the following tier vertices. For this purpose:

a) the previous tier vertices are chosen in succession for each of which the line with number with the same name is chosen in a contiguity matrix;

b) columns numbers defined by nonzero elements are written out for every line;

c) in each of the formed subsets there expelled the apexes numbers (columns numbers) concerning which apexes subsets in the previous tiers were formed. All not deleted elements (columns numbers) form the following tier subsets.

Step 2. If the tier number is equal to (T_0+1) – it means the end. Otherwise - to pass to a step 1.

2.3.7 Problems about flow

The problems about streams in networks draw special attention thanks to specificity of its structure. Let's introduce some definitions.

D e f i n i t i o n 1. *The number* x_{ij} *is called* ***a stream on an arch*** (*to a rib* (i, j), *if* $x_{ij} \le b_{ij}$, *where* b_{ij} *- bandwidth of this arch.*

D e f i n i t i o n 2. *Stream* P_{st} *from some vertex s named* ***a source*** *to some vertex t named* ***a flow*** *in a network is a set of non-negative numbers* x_{ij} (*arches streams*) *if these numbers satisfy the following restrictions*:

$$\sum x_{ij} - \sum x_{jr} = \begin{cases} -P_{st}, \text{if } j = s \text{ (source)}; \\ 0, \text{if } j \neq s, t; \\ P_{st}, \text{if } j = \text{t, (flow)} \end{cases} \quad (1)$$

$$P_{st} \ge 0;\ 0 \le x_{ij} \le b_{ij} \text{ for all } (i, j) \in V. \quad (2)$$

Here the first sum is taken on the arches leading to the vertex j, and the second sum – on the arches leading from j.

Restriction (1) characterizes that fact that to each apex (except a source and a flow) comes so much stream, as from it gets out of it and is called *a condition of conservation of a stream*. Restriction (2) means that the stream on an arch is restricted by its bandwidth.

D e f i n i t i o n 3. ***Cross-section*** (***cut***) ***of a network*** *is called irredundant set of arches* (*ribs*) *while excluding it from a network its connectivity is broken.*

D e f i n i t i o n 4. ***Bandwidth*** *of cross-section is called the sum of arches bandwidth oriented in a direction from a source to a flow or ribs making this cross-section*.

D e f i n i t i o n 5. *The cross-section dividing s and t and possessing the least bandwidth is called* ***the minimum cross-section***.

The minimum cross-section dividing a source *s* and a flow *t* is analogue of «bottle neck» in any network and, hence, the maximum stream magnitude cannot exceed its bandwidth. There is a theorem proved by Ford and Falkerson conferming that **maximum stream magnitude is always equal to the minimum bandwidth among all cross-sections dividing *s* and *t***. The theorem of the maximum stream and the minimum cross-section is basic in the theory of streams in networks.

D e f i n i t i o n 6. *The network having one source and one flow is called* ***double-pole***.

D e f i n i t i o n 7. *The network where each pair of vertices can considered as a source and a flow is called* ***multipolar***.

D e f i n i t i o n 8. *If there are some sources and some flows in a network and a stream can go from any source to any flow such stream is called* ***one-component*** (for example networks of gas pipelines, oil pipelines, energy line etc.).

D e f i n i t i o n 9. *If in networks with several sources and flows the stream must go from some selected sources to some fixed flows such stream is called* ***multicomponent*** (for example, networks of the informational communication, mail conveyances post, etc.).

At any moving of some objects from one point to another there appear streams and if they are regarded taking into account restrictions on moving there is a natural task about finding the maximum magnitude of a stream which can exist in the conditions of the defined restrictions.

For an estimation of connecting network bandwidth it is enough to define the maximum stream magnitude that can pass it and besides calculation of its arc components finding, the transfer paths can appear to be not necessarily at all. According to *the theorem of the maximum stream and the minimum cut*, the specified problem can be shown to definition of minimum cross-section bandwidth. Most simple way of a finding such cross-section for a double-pole network is looking through all possible cross-sections dividing a set of vertices *N of a network* into two not correlated set: N_1, including a source and N_2, including a flow and selection among them cross-section which possesses the minimum bandwidth. Unique difficulty which appears here lies in definition of cross-sections set. The algorithm allowing to ,overcome the defined difficulty and possessing high efficiency from the point of view of implementation on the computer is given below. The principal idea accepted for its basis is essentially as follows.

Vertices of subsets N_1 and N_2 where the next cross-section divides a network are appropriated to some marks, for example,: $0 \Rightarrow N_1$; $1 \Rightarrow N_2$.

Let the source s belong to N_1, and the flow t belong to N_2. So, the vertex s will be marked with zero and t – with one. It is necessary to distribute marks between the others (n - 2) vertices for each possible cross-section of a network.

For this purpose it is necessary to be set by a rule generating various combinations of zeroes and ones which can be used to sort remained (n - 2) vertices in two subsets N_1 and N_2. On the basis of such a rule we can use a principle of a number representation of a natural sequence in a binary aspect. Capacity of binary number in this case is defined by value (n - 2) and, hence, the maximum quantity of binary numbers will make $2^{(n-2)}$. The same magnitude corresponds to the greatest possible quantity of the cross-sections dividing a network into two subsets N_1 and N_2 of not correlated vertices. Every position of binary number should be put to the vertex number (the order makes no difference) in conformity. We will remind that vertices s and t do not participate here as they have already obtained marks and according to them are distributed in subsets: $s \in N_1$, $t \in N_2$.

The algorithm is formulated as follows.

Step 0. Let's appropriate to a source (to an apex s) a mark «0», and to a flow (to an apex t) a mark «1». Let's consider that the value of the counter of cross-sections equals 0.

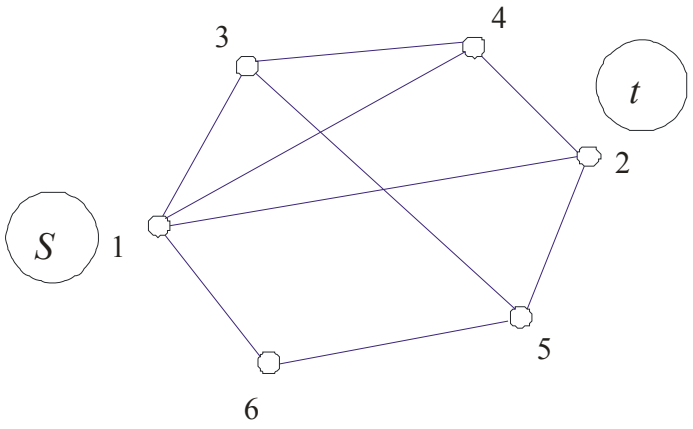
Step 1. Let's form binary representation of number C and sort vertices according to marks. Let's define bandwidth of cross-section for the obtained alternative of vertices distribution as the sum of bandwidth of the arches (ribs) making considered cross-section.

Step 2. Let's increase value of a variable C to one. If $C=2^{(n-2)}$, let's pass to a step 3, otherwise – to a step 1.

Step 3. Among the obtained values $\{M_i (N_1, N_2)\}$ let's choose the least one.

Let's consider an instance illustrating the algorithm operation.

Let it be required to size up bandwidth between a source s and a flow t in a non-directional network, the model and matrix of ribs bandwidth of which are shown on figure 2.11.



	1	2	3	4	5	6
1	0	10	5	3	0	1
2	10	0	0	2	4	0
3	5	0	0	3	6	0
4	3	2	3	0	0	0
5	0	4	6	0	0	4
6	1	0	0	0	4	0

Figure 2.11

Let's assume s=1 and it is marked as «0», t=2 and it is marked as «1».

Combination of vertices marks for all possible cross-sections and magnitude of their bandwidth are put into the table 2.1.

Table 2.1

№ Cross-sections	Binary combinations Apexes numbers				M_i (N_1N_2)
	3	4	5	6	
0	0	0	0	0	**16**
1	0	0	0	1	**21**
2	0	0	1	0	**22**
3	0	0	1	1	**19**
4	0	1	0	0	**20**
5	0	1	0	1	**25**
6	0	1	1	0	**30**
7	0	1	1	1	**23**
8	1	0	0	0	**30**
9	1	0	0	1	**35**
10	1	0	1	0	**24**
11	1	0	1	1	**21**
12	1	1	0	0	**21**
13	1	1	0	1	**33**
14	1	1	1	0	**23**
15	1	1	1	1	**19**

So, for example, zero by order cross-section will be characterised in accordance with symbolics, given in table 2.1. by the following apexes division into subsets:

$$N_1 = (1,3,4,5,6);\ N_2 = (2).$$ ribs:

this cross-section is made

$$(1,2),\ (4,2),\ (5,2).$$

The total bandwidth of these ribs is equal to 16. Next cross-section is characterized by division

$$N_1 = (1,3,4,5);\ N_2 = (2,6)$$

It includes the ribs: (1,2), (1,6), (4,2), (5,2), (5,6) they also define cross-section bandwidth accordingly equal 21.

As we see from the table 2.1 the least magnitude of the cross-section bandwidth refers to cross-section with number 0. It also defines bandwidth of the set network.

Remarks. It should be noted that the way of obtaining the set of the ribs included into some cross-section generates the ribs which can be absent in the physical count is given above. These ribs can be either expelled from consideration, or can be considered with zero bandwidth.

3 THE COMPLEX TASK

3.1 Construction of telecommunication network models

3.1.1 Study sub item 2.1 and 2.2 of the Key statements.

3.1.2 In Appendix A choose the alternative of initial data defined by number obtained from the two last figures of the student's card. Initial data are presented by a file, 1st and 2nd lines in which contain numbers of the points connected by communication lines, and 3rd - weight characteristics of these lines.

The initial telecommunication network contains 10 points and 19 lines providing their communication between points in both directions.

3.1.3 Build all forms of model representation of an initial telecommunication network (graph, and also digital forms of graph representation) and show them supplying with necessary comments in an explanatory note.

3.1.4 Answer the following key questions in writing:

- Which problems refer to the synthesis problems class and which - to the analysis class?
- What is model network representation is used for?
- List the forms of model telecommunication network representation as object of synthesis and analysis. Characterize each of them.
- What is called a graph? oriented graph? Non-directional graph?
- What do relations of contiguity and incidence of elements graph represent?
- What does distinctive feature of network model consist in?

3.2 Synthesis of subscriber access network

3.2.1 Study sub item 2.1, 2.3.1, 2.3.2, 2.3.3 of the Key statements.

3.2.2 Build a cable network of subscriber's access for which the minimum of expenses for the line structure is provided.

There given: the list of network points (n = 10) and a matrix of distances between them. As the capacity of distance matrixes take advantage of weight matrix obtained while performing the Task 3.1 considering values of ribs weight as distances in kilometres. Absent elements values of weight matrix should be considered as infinitely big distances, i.e. physical cable laying between some pairs of points is impossible.

Cost of one line structures kilometre along all directions is the same and equal to 10 c.u.

3.2.3 On the synthesized network for which the minimum of the line structure cost is provided, define the point where it is expedient to place basic networks knot (BK) of subscriber's access from the point of view of minimization of the general extent of subscriber's lines from BK to all subscriber's points (SP).

Remarks. For the synthesized network it is necessary to build a matrix of distances between all points of a subscriber's network which should be used as initial data for the solution of this problem.

3.2.4 For the organization of a subscriber's network with a stationary radio access it is necessary to provide selection of base station (BS) location, providing a resistant to radio communication at comparatively small power of the radio-transmitting set.

There given: the list of points (n=10) and a matrix of distances between them. As the last one take advantage of weight matrix obtained while performing the Task 3.1. Absence of matrix elements values should be considered as absence of directly radio visibility between them.

3.2.5 Show the problem set in the form of verbal and mathematical models, the results of their solutions matching to your initial data alternative and explanations to their solutions methods in the explanatory note.

3.2.6 Answer the following key questions in writing:

- What is the assignment of subscriber's access network?
- List requirements which the solving the problem of minimum cost network synthesis should satisfy.
- What graph is called covering tree?
- Formulate the idea of Tonic algorithm providing minimum cost network setting.
- Can Tonic algorithm be applied for minimum cost network setting? If it can, then what way?
- Can Tonic algorithm be applied at incomplete initial distances matrix? What does it mean?
- Is Tonic algorithm exact or heuristic?
- What apex is called a graph median? What are initial data for its definition?

- Formulate algorithm of graph median definition by a matrix of distances between all pairs of graph avertices.
- What vertex is called the graph centre?
- Formulate the algorithm of graph centre definition.

3.3 Synthesis of the internodal network

3.3.1 Study sub item 2.1, 2.3.4 of the Key statements.

3.3.2 Define the contour of the least length merging **internodal** telecommunication network points into the hoist ring.

There given: location of points (n=10), the circuit design of the linearly-cable conduit presented as weight matrix of graph ribs representing a network. Weight characteristics of ribs match to distances in kilometres of linearly-cable constructions between network points. Use the weight matrix obtained while performing the Task 3.1 as mentioned matrix.

Remarks. For the problem solution it is possible to take advantage of heuristic algorithm of «the problem about the travelling salesman» solving. As heuristics it is possible to use the rule shown in the item 2.3.4, or to formulate a heuristic rule independently.

3.3.3 Show the obtained solution and the explanation to it in an explanatory note.

3.3.4 Answer the following key questions in writing:
- What does «the problem about the travelling salesman » consist in?
- What is the exact solution of «the problem about the travelling salesman»?
- What contour is called Hamiltonian cycle?
- What is called as heuristics? What algorithms refer to the heuristic ones?
- Formulate advantages and disadvantages of exact and heuristic algorithms.

3.4 Construction of the routing matrixes

3.4.1 Study sub item 2.1, 2.3.5, 2.3.6 of the Key statements.

3.4.2 Build routing matrixes for each network knot (n=10) providing main route selecting and also two bypaths while connecting sections setting up from the considered knot to all the others.

For a main route (a way of the first sampling) it is necessary to use the shortest paths from the knot s (s=1.... n) to other network points. For bypaths (paths of the second and third selections) – the shortest paths by transits at maximum

admissible transience $T_0 = 2$. In the presence of several paths of equal transience preference one should favor to the shortest by length path.

For definition of the shortest by length paths as initial matrix of lengths of the lines connecting knot network points, take advantage of the weight matrix obtained while performing the Task 3.1 and for minimum transience path finding - contiguity matrixes.

3.4.3 Methodical instructions.

Routing matrixes are stored in every UK_i networks and intended for definition of a direction of the proceeding communication line (communication channel) at connecting section of information message transmitting from initial point to the point of destination. It is possible to forsee additional directions (bypaths) selection in case of the first selection of occupation direction on the definite UK_s in the presence of the switching equipment feasibilities. The order of directions selecting is defined by a routing matrix the quantity and line-numbering of which match to paths number p and to the assigned order of their occupation, and columns number - to points of destination numbers (figure 3.1).

	1	2	…	*j*	*i*	...	*n*
1				*j*	*I*		
2				*r*	*J*		
… … … … … … … … … … … … … … … … … … … ….							
p				1	*L*		

Figure 3.1

Element m_{ij} of routing matrix for UK_s is the knot number of contiguous UK_s in transit of matching selection from UK_s to the point of destination *J*.

3.4.4 Show the general problem assignment (verbal model), results of the solution and explanations matching to them in the explanatory note.

3.4.5 Answer the following key questions in writing:

- What class of problems does routing matrixes construction refer to?
- What is called the path in a network, a path length, way transience and the shortest way?
- Show practical situations where there is a problem of the shortest way definition.
- What is Dijkstra's algorithm of the shortest ways searching based on?
- What kinds of mark are used at Dijkstra's algorithm operation?

- Is it possible by means of Dijkstra's algorithm to find paths variety, the shortest by transience? Which initial data are required for this purpose?
- What method is it possible to obtain paths variety of set transience by?
- What does the idea of multilevel tree construction method consist in?

3.5 Estimation of network capacity between points pair

3.5.1 Study sub item 2.1, 2.3.7 of the Key statements.

3.5.2 Define bandwidth between network points pair which numbers match to the last two figures of the student's card number.

There given*:* the connecting network containing n=10 points and a matrix of communication lines bandwidth. As the last one take advantage of the weight matrix obtained while performing the Task 3.1 assuming that elements value of this matrix matches to lines bandwidth.

3.5.3 Show the general problem setting, results of its solution and the explanation to it in an explanatory note.

3.5.4 Answer the following key questions (in writing):

- What is a stream from a source to a flow?
- Give definition of a network cross-section.
- What cross-section is called minimum one?
- Formulate the theorem of the maximum stream and the minimum cross-section.
- What network is called a double-pole one? the multipolar? one-component? multicomponent?
- What difference is between multipolar and the multicomponent network?
- What does the basic idea of network bandwidth estimation consist in?

4 THE LIST of LITERATURE

1. Frank G., Frisch I. Networks, communication, streams. – M.: Communication, 1978.
2. Mainika E. Optimization algorithms on networks and graphs. – M.: Mir, 1981.

Appendix A

The variants of individual tasks

01

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	20	25	6	43	95	30	12	10	35	71	63	50	18	48	21	90	15	10

02

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	70	11	90	25	80	15	3	52	30	15	93	60	20	18	45	75	13	14	8

03

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	18	22	15	34	17	10	25	14	20	31	40	21	19	23	70	10	17	50

04

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	9	31	15	73	21	8	19	45	27	9	3	90	41	18	80	77	11	13

05

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	3	70	90	21	35	10	15	50	16	60	18	37	30	21	15	80	14	12	10

06

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	30	5	9	95	42	27	14	5	98	19	17	11	9	25	50	2	71	80

07

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	83	31	41	50	19	17	16	25	21	33	13	25	9	19	13	53	8	12

08

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	15	75	25	18	18	90	4	30	91	15	32	14	93	25	7	11	2	65

09

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	7	10	11	25	15	30	4	3	40	12	14	32	15	18	20	6	13	10	5

10

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	20	10	15	9	17	7	19	19	9	40	30	15	16	10	50	17	32	12	16

11

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	30	73	12	11	9	4	17	12	4	10	7	32	12	8	7	31	70	35	14

12

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	19	20	15	18	22	20	13	17	54	31	24	18	73	51	16	42	17	19	53

13

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	31	17	70	18	60	32	15	9	52	73	7	18	20	43	19	27	18	93

14

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	81	33	16	34	20	18	10	31	7	18	13	27	31	18	30	54	20	81	17

15

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	12	71	30	21	7	9	14	3	20	31	18	20	9	51	21	18	34	19

16

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	30	12	10	30	10	11	19	24	14	10	30	27	19	9	7	14	10	7	90

17

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	20	70	93	17	19	63	18	10	45	67	13	9	10	21	31	21	18	16

18

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	3	14	5	70	31	9	7	13	10	11	16	54	70	10	22	17	13	18	10

19

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	20	16	31	22	10	19	34	19	35	27	19	16	37	10	42	31	70	15

20

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	43	19	61	15	45	18	20	13	15	20	37	18	12	71	13	14	16	31

21

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	20	16	31	22	10	19	34	19	35	27	19	16	37	10	42	31	70	15

22

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	20	3	15	14	50	19	80	25	18	30	21	45	15	9	13	70	27	20	28

23

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	21	63	18	19	17	19	7	21	19	53	15	30	10	16	17	50	12	32	15

24

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	12	30	55	75	10	15	90	99	10	70	80	19	73	40	10	33	7	3

25

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	21	35	5	25	13	18	71	10	15	37	90	30	16	19	14	7	20	99

26

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	35	13	41	99	80	95	18	15	7	9	70	17	12	11	10	52	1	83

27

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	31	14	15	92	11	43	19	13	15	18	20	31	71	16	18	31	16	13

28

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	20	70	93	17	19	63	18	10	45	67	13	9	21	31	21	18	16	21

29

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	32	18	27	13	31	27	60	34	14	18	12	31	10	3	7	12	42	5

30

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	19	10	50	9	13	12	19	13	10	42	18	10	17	5	15	13	17	19

31

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	19	37	18	27	34	63	19	17	35	15	19	72	12	13	43	51	18	12	11

32

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	18	22	15	34	17	10	25	14	20	31	40	21	19	23	70	10	17	50

33

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	8	12	30	15	7	9	14	4	20	17	11	11	9	14	7	9	11	10

34

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	81	33	16	34	20	18	10	31	7	18	13	27	31	18	30	54	20	81	17

35

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	20	25	6	43	95	30	12	10	35	71	63	50	18	48	21	90	15	10

36

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	70	11	90	25	80	15	3	52	30	15	93	60	20	18	45	75	13	14	8

37

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	18	22	15	34	17	10	25	14	20	31	40	21	19	23	70	10	17	50

38

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	9	31	15	73	21	8	19	45	27	9	3	90	41	18	80	77	11	13

39

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	3	70	90	21	35	10	15	50	16	60	18	37	30	21	15	80	14	12	10

40

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	30	5	9	95	42	27	14	5	98	19	17	11	9	25	50	2	71	80

41

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	83	31	41	50	19	17	16	25	21	33	13	25	9	19	13	53	8	12

42

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	15	75	25	18	18	90	4	30	91	15	32	14	93	25	7	11	2	65

43

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	7	10	11	25	15	30	4	3	40	12	14	32	15	18	20	6	13	10	5

44

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	20	10	15	9	17	7	19	19	9	40	30	15	16	10	50	17	32	12	16

45

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	30	73	12	11	9	4	17	12	4	10	7	32	12	8	7	31	70	35	14

46

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	19	20	15	18	22	20	13	17	54	31	24	18	73	51	16	42	17	19	53

47

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	31	17	70	18	60	32	15	9	52	73	7	18	20	43	19	27	18	93

48

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	81	33	16	34	20	18	10	31	7	18	13	27	31	18	30	54	20	81	17

49

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	12	71	30	21	7	9	14	3	20	31	18	20	9	51	21	18	34	19

50

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	30	12	10	30	10	11	19	24	14	10	30	27	19	9	7	14	10	7	90

51

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	20	70	93	17	19	63	18	10	45	67	13	9	10	21	31	21	18	16

52

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	3	14	5	70	31	9	7	13	10	11	16	54	70	10	22	17	13	18	10

53

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	20	16	31	22	10	19	34	19	35	27	19	16	37	10	42	31	70	15

54

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	43	19	61	15	45	18	20	13	15	20	37	18	12	71	13	14	16	31

55

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	20	16	31	22	10	19	34	19	35	27	19	16	37	10	42	31	70	15

56

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	20	3	15	14	50	19	80	25	18	30	21	45	15	9	13	70	27	20	28

57

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	21	63	18	19	17	19	7	21	19	53	15	30	10	16	17	50	12	32	15

58

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	12	30	55	75	10	15	90	99	10	70	80	19	73	40	10	33	7	3

59

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	21	35	5	25	13	18	71	10	15	37	90	30	16	19	14	7	20	99

60

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	35	13	41	99	80	95	18	15	7	9	70	17	12	11	10	52	1	83

61

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	31	14	15	92	11	43	19	13	15	18	20	31	71	16	18	31	16	13

62

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	20	70	93	17	19	63	18	10	45	67	13	9	21	31	21	18	16	21

63

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	32	18	27	13	31	27	60	34	14	18	12	31	10	3	7	12	42	5

64

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	19	10	50	9	13	12	19	13	10	42	18	10	17	5	15	13	17	19

65

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	19	37	18	27	34	63	19	17	35	15	19	72	12	13	43	51	18	12	11

66

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	18	22	15	34	17	10	25	14	20	31	40	21	19	23	70	10	17	50

67

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	8	12	30	15	7	9	14	4	20	17	11	11	9	14	7	9	11	10

68

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	81	33	16	34	20	18	10	31	7	18	13	27	31	18	30	54	20	81	17

69

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	20	25	6	43	95	30	12	10	35	71	63	50	18	48	21	90	15	10

70

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	70	11	90	25	80	15	3	52	30	15	93	60	20	18	45	75	13	14	8

71

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	18	22	15	34	17	10	25	14	20	31	40	21	19	23	70	10	17	50

72

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	9	31	15	73	21	8	19	45	27	9	3	90	41	18	80	77	11	13

73

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	3	70	90	21	35	10	15	50	16	60	18	37	30	21	15	80	14	12	10

74

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	30	5	9	95	42	27	14	5	98	19	17	11	9	25	50	2	71	80

75

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	15	83	31	41	50	19	17	16	25	21	33	13	25	9	19	13	53	8	12

76

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	15	75	25	18	18	90	4	30	91	15	32	14	93	25	7	11	2	65

77

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	7	10	11	25	15	30	4	3	40	12	14	32	15	18	20	6	13	10	5

78

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	20	10	15	9	17	7	19	19	9	40	30	15	16	10	50	17	32	12	16

79

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	30	73	12	11	9	4	17	12	4	10	7	32	12	8	7	31	70	35	14

80

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	19	20	15	18	22	20	13	17	54	31	24	18	73	51	16	42	17	19	53

81

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	31	17	70	18	60	32	15	9	52	73	7	18	20	43	19	27	18	93

82

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	81	33	16	34	20	18	10	31	7	18	13	27	31	18	30	54	20	81	17

83

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	10	12	71	30	21	7	9	14	3	20	31	18	20	9	51	21	18	34	19

84

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	30	12	10	30	10	11	19	24	14	10	30	27	19	9	7	14	10	7	90

85

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	20	70	93	17	19	63	18	10	45	67	13	9	10	21	31	21	18	16

86

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	3	14	5	70	31	9	7	13	10	11	16	54	70	10	22	17	13	18	10

87

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	20	16	31	22	10	19	34	19	35	27	19	16	37	10	42	31	70	15

88

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	43	19	61	15	45	18	20	13	15	20	37	18	12	71	13	14	16	31

89

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	13	20	16	31	22	10	19	34	19	35	27	19	16	37	10	42	31	70	15

90

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	20	3	15	14	50	19	80	25	18	30	21	45	15	9	13	70	27	20	28

91

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	21	63	18	19	17	19	7	21	19	53	15	30	10	16	17	50	12	32	15

92

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	12	30	55	75	10	15	90	99	10	70	80	19	73	40	10	33	7	3

93

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	21	35	5	25	13	18	71	10	15	37	90	30	16	19	14	7	20	99

94

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	50	35	13	41	99	80	95	18	15	7	9	70	17	12	11	10	52	1	83

95

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	31	14	15	92	11	43	19	13	15	18	20	31	71	16	18	31	16	13

96

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	18	20	70	93	17	19	63	18	10	45	67	13	9	21	31	21	18	16	21

97

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	32	18	27	13	31	27	60	34	14	18	12	31	10	3	7	12	42	5

98

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	90	19	10	50	9	13	12	19	13	10	42	18	10	17	5	15	13	17	19

99

Initial vertex	1	1	1	1	1	2	2	3	3	4	5	5	6	6	6	7	8	8	9
Final vertex	2	4	5	7	8	3	4	6	7	6	6	8	7	8	9	10	9	10	10
Weight	19	37	18	27	34	63	19	17	35	15	19	72	12	13	43	51	18	12	11

Appendix B

The ministry of transport and Communications of Ukraine

The state administration of communication

ODESSA NATIONAL ACADEMY of TELECOMMUNICATIONS named after A.S.POPOV

Chair of Communication Networks

The complex task
on discipline ***Telecommunication Information Networks***

On the topic:

Synthesis and analysis elements of telecommunication networks

Done by:
student group _____ department _____

_____
(S.N.P.)

Checked by:_____
Date:_____
Mark:_____

Odessa, 201_

Text editor – *Radius E. A.*

Computer aided make-up – *Korneychuk E. S.*

Sent to printing 3.12.2010. Signatured for printing 27.12.10
Format 60/88/16. Order № 4363.
Number of 200 copies Total output volume 3,0 printed pages
ONAT, 2010