공학 연구논문집 제28권 1호 pp. 95~111, 울산대학교.
Journal of Engineering Research Vol. 28, No. 1, pp. 95~111, 1997
University of Ulsan.

네트워크 모형의 개체지향적 전산구현

고재문 · 서준용
산업공학과

<요 약>

본 논문에서는 개체지향 프로그래밍 개념을 이용하여 네트워크 모형의 전산 구현 통합 패키지를 개발하였다. 네트워크 모형을 표현하기 위하여 내부적으로 단일 연결리스트를 채택하였다.

풀-다운 메뉴 방식을 이용하여 파일, 자료 입출력, 그리기, 문제선택 메뉴를 구성하였다. 자료입력의 경우 마우스를 이용한 입력, 고밀도 네트워크에 대한 파일 입력, 저밀도 네트워크에 대한 파일 입력의 3가지 방식을 제시하였다. 현실감을 높이기 위해 모형을 그래픽으로 나타낼 수 있도록 하였다. 또한 사용자가 자신의 환경에 따라 선택이 가능하도록 고밀도 및 저밀도 자료형식으로 저장이 가능하도록 하였다.

응용 예를 위하여 최단경로나무문제, 최소걸침나무문제, 최대흐름문제의 해법을 첨가하였으며, 사용자는 자신의 문제에 대한 해법을 서브루틴으로 추가할 수 있도록 하였다.

개체지향개념을 이용하였기 때문에 네트워크 모형을 쉽게 확장할 수 있으며 프로그램을 재사용할 수도 있다. 따라서 사용자는 네트워크 모형을 전산구현하는데 드는 노력을 대폭 경감시킬 수 있다.

Computerization of the Network Model Using the Object-Oriented Programming

Koh, Jae-Moon · Seo, Joon-Yong
Dept. of Industrial Engineering

<Abstract>

An integrated computer-package for the network model is developed, where the

network model is implemented using object-oriented programming. Node object, arc object and network object are defined as user-defined objects. Singly linked list is adopted as the internal data structure for network representation.

All works are designed to be implemented through the pull-down menu. The main menu consists of FILE, INPUT TYPE, DRAWING, and PROBLEM SELECTION. Three methods are suggested for data input: mouse input, file input for dense-type data, and file input for sparse-type data. The model can be displayed graphically for the sense of reality. It can be stored in both dense-type and sparse-type data format, which enables the user to fit the model to his/her environment.

Shortest path tree problem, minimum spanning tree problem, and maximal flow problem are applied for illustrations. Since the network model is realized using the object-oriented concept, the user can apply any problem with its own subroutine.

The object-oriented programming gives the benefit of modifying or expanding the model easily and reusing the program. As a result, the user can dramatically save time and efforts for implementing the network model.

서 론

일반적으로 네트워크의 설계나 운영상의 문제와 관련된 해법을 평가할 때 평가의 신뢰도를 높이기 위해 많은 네트워크에 대하여 실험을 해야 한다. 그러나 현실적으로 그 대상이 많지 않아 모의실험 등을 통하여 가상적인 네트워크를 대상으로 평가하게 된다.

그런데 이러한 모의실험을 위한 네트워크 모형의 전산구현이 같은 종류의 문제라도 연구자마다 독자적으로 이루어져, 어떤 연구에 사용된 네트워크 전산모형을 다른 연구에서 사용하려면 모형을 변환하는데 상당한 노력이 든다. 그 결과 네트워크 전산모형의 호환성 유지가 어려우며 연구결과의 상호비교가 무의미해지는 경우가 종종 발생한다. 따라서 범용적으로 사용할 수 있는 네트워크 모형에 대한 연구가 필수적이다.

네트워크 모형에 대한 기존의 연구들을 보면 모듈을 이용한 구조적 프로그래밍 기법을 응용하고 있다. 그러나 사용자가 이러한 기법을 사용하려면 변수, 함수, 프로그램의 구조 등을 완벽히 이해해야 한다. 이 경우에 연구자가 네트워크 본연의 문제에 전념하기보다는 프로그래밍에 더 많은 시간을 소비하게 된다. 이러한 문제점들을 해소할 수 있는 방법이 개체지향적 프로그래밍(Object Oriented Programming)의 도입이다.

개체지향 프로그래밍이란 구조적 프로그래밍 개념과 추상화 개념을 바탕으로 나온 개념으로 개체를 중심으로 프로그램을 작성하는 것을 말한다.[11] 즉, 개체지향 언어에서는 개체 하나하나가 프로그램 모듈처럼 다루어져 그 안에 사용되는 데이터와 프로시져가 함께 정의된 상태에서 프로그램을 작성해 나간다. 따라서 프로그래머는 개체별로 오류발견이나 수정을 쉽게 할 수 있으며, 상황에 맞추어 프로그램을 용이하게 변환할 수 있어 좀더 탄력적인 소프트웨어를 생산할 수 있다.[2, 4]

본 연구에서는 네트워크 연구를 하는데 사용되는 네트워크 전산모형의 통합된 전산패키지를 개발하고자 한다. 즉 사용자가 기본적인 자료만을 입력하면 사용자의 의도에 맞는 전

산모형을 만들어 사용자에게 제공하는 프로그램을 개발하는 것이 본 연구의 목적이다. 네트워크 모형을 구현하기 위해 사용자가 내부구조를 모르더라도 쉽게 이용할 수 있고, 쉽게 수정 및 확장이 가능한 개체지향적 접근방법을 적용한다. 내부적인 자료구조로는 연결리스트로 다루어진 방식을 이용한다.

본 연구에서 개발하고자 하는 전산패키지는 자료 입력에서 몇 가지 공통된 형식을 제공하여, 그 가운데 사용자 자신의 형식과 쉽게 호환이 가능한 것을 선택하여 자료를 입력할 수 있도록 한다. 입력된 자료, 전산구현된 네트워크 및 적용된 해법의 결과를 윈도우를 통하여 시각적으로 보이고, 사용자가 자신이 이미 구현한 프로그램에 이용 가능한 데이터 형식을 가질 수 있도록 사용자의 요구에 맞는 데이터파일로 출력할 수 있도록 한다.

본 연구에 사용된 개체지향적 언어는 윈도우즈 환경을 구현하기 편리하고, 프로그램 개발에 지원되는 내부 Class(MFC, Microsoft Foundation Class)가 풍부하여 개발자로 하여금 실질적인 개발에만 전념할 수 있는 Microsoft 사의 Visual C++를 이용한다.[3,10,13]

본 연구에서 개발된 전산패키지 적용에 대한 예로서, 가장 기본적인 문제라 할 수 있는 최단경로나무문제, 최소걸침나무문제, 최대흐름문제에 대한 해법을 제시한다.[1]

이런 전산패키지를 이용하게 되면 사용자는 네트워크의 전산구현에 따르는 시간의 소모를 줄여 본연의 연구에만 전념할 수 있고, 다양한 네트워크를 평가할 수 있으며, 연구결과의 상호비교가 가능해진다.

1. 개체지향 개념을 이용한 네트워크 모형

본 연구에서는 Visual C++의 MFC에서 유도된 4가지 주요 클래스 및 대화상자를 위한 클래스와 3개의 사용자정의 클래스를 사용한다.

1.1 MFC 유도 클래스

MFC에서 유도된 주요 클래스는 document, view, main frame window, application 클래스이다.

본 연구에서 document 클래스는 CNetDoc라는 이름을 가지고 있으며 MFC의 CDocument로부터 유도된다. document 클래스는 프로그램 데이터를 읽고 저장하며, 이 데이터를 디스크 파일에 기록하는 일을 맡고 있다. 또한 본 연구에서 적용된 문제와 데이터의 저장에 관한 선언도 document 클래스에서 이루어진다.

View 클래스는 CNetView란 이름을 가지고 있으며 MFC의 CView로부터 유도된다. view 클래스는 프로그램 데이터를 보여주고 사용자로부터 입력을 처리하는 일을 한다. 본 연구에서는 view 클래스를 통해 네트워크 화면출력에 관한 일을 한다.

Main frame window 클래스는 CMainFrame 이란 이름을 가지고 있으며 CFrameWnd로부터 유도된 것이다. 이 클래스의 역할은 메뉴 바(menu bar)나 타이틀 바(title bar)의 전시와 같은 주요 프로그램의 윈도우를 관리한다.

마지막으로 application 클래스는 CNetApp란 이름을 가지고 있으며, MFC클래스의 CWinApp로부터 유도된 것이다. 이 클래스는 전체적인 프로그램을 관리한다. 즉, 프로그램의 초기화와 삭제와 같은 다른 클래스에서 행해지지 않는 일반적인 작업을 실행한다.

1.2 사용자 정의 클래스

사용자 정의에 의한 클래스는 마디와 가지 그리고 네트워크의 세 가지 개체들이다. 각각의 개체는 Visual C++의 MFC에서 제공되는 CObject 로부터 파생된다. 데이터의 추상화를 위해 데이터의 속성들을 private로 선언하여, 외부에서 데이터의 속성을 변화시킬 때 매개함수를 이용하는 간접적인 방법을 채택한다.[3, 10, 13]

1.2.1 마디 개체(node object)

마디 개체는 마디에 관련된 모든 정보를 다룬다. 즉 마디의 이름, 마디의 값, 그리고 화면상으로 표현하기 위한 화면상의 위치 등에 관한 정보를 가진다. 마디 고유의 속성은 모두 private로 선언되기 때문에 마디 클래스의 프로시져, 즉 멤버함수를 통해 각 속성에 간접적으로 접근할 수 있다. 마디 개체 하나가 생성되기 위해서는 생성되는 마디 개체의 정보를 가진 생성자를 통하여 마디의 이름과 위치가 입력된다. 본 연구에서는 연결리스트를 이용하기 위해 마디 개체 내에 가지 개체의 주소를 가지도록 한다. 마디 개체의 클래스 이름은 CNode로 클래스 정의는 그림 1과 같다.[5, 6]

```
class CNode: public CObject
{
private:
   friend class CEdge;     // 마디 개체의 속성으로 가지 개체를
                           // 가지기 위해 선언
   CString node_name;      // 마디의 이름
   float label;            // 마디의 값
   BOOL  node_status;      // 마디의 방문상태를 점검
   CRect node_point;       // 화면상의 마디 위치
   CNode* pred;            // 해법이 적용된 결과 각 마디 개체의
   ...                     // 선행 마디
public:
   CEdge* first_arc;       // 마디에 연결된 첫 번째 가지
   CNode(){};
   CNode(CString name);    // 생성자
   CNode(CString name, int x, int y);
   CRect GetnodePoint(int x, int y);   // 화면의 위치를 가져온다.
   CString Get_Node_Name();        // 마디 이름에 접근을 위한 함수
   ...
};
```

그림 1. 마디 클래스 정의

1.2.2 가지 개체(arc object)

가지 개체는 가지에 관련된 모든 정보를 다룬다. 가지의 시작점과 끝점을 위한 위치, 가지의 값, 그리고 문제적용에서 방문상태 등의 속성을 가진다. 가지의 생성은 시작마디와

끝마디의 주소와 가지의 값, 그리고 가지의 시작과 끝 위치를 가진 생성자를 통하여 입력된다. 그림 2는 가지 개체의 클래스이다.

```
class CEdge: public CObject
{
private:
    CRect arc_point;     // 가지의 위치
    float weight;        // 가지의 값
    BOOL arc_status;     // 가지의 상태
     ...
public:
    CEdge* next;         // 가지의 연결을 위한 가지 주소
    CNode* from_node;    // 가지의 시작 마디의 주소
    CNode* to_node;      // 가지의 끝 마디의 주소
    CEdge(){};
    CEdge(CNode* from, CNode* to, float new_data, CRect rect);
     ...                 // 가지 생성자
}
```

그림 2. 가지 클래스 정의

1.2.3 네트워크 표현

네트워크의 표현을 위한 전산구현 방법에는 크게 인접행렬에 의한 방법과 연결리스트에 의한 방법이 있다.

(1) 인접행렬

일반적으로 인접행렬을 이용한 네트워크 모형의 전산구현은 마디-마디 인접 행렬(node-node adjacency matrix)을 많이 이용한다. 인접행렬을 이용할 경우 다음과 같은 장점을 가진다. 첫째, 어떤 마디에서 나머지 마디에 접근하기 위해 임의의 첨자를 증가 또는 감소시킨다. 즉 첨자를 이용하여 주어진 배열의 한 원소에서 다른 원소에 쉽게 접근할 수 있다. 둘째, 배열에 저장된 데이터의 처리가 효율적이다.

그러나 초기에 각 배열의 크기를 정해야 한다. 만일 마디의 수가 배열의 크기보다 많을 경우 입력되는 마디를 모두 수용할 수 없다. 또한 마디의 수가 배열의 크기보다 작을 경우 불필요한 기억공간을 가지게 된다. 또한 새로운 자료의 삽입과 자료의 삭제가 요구될 때 배열을 재배치해야 하는 어려움이 따른다. [5, 12]

각 표현의 예를 들기 위해 그림 3과 같은 네트워크를 생각한다. 이것은 방향성을 가지는 네트워크이다.

그림 3을 인접행렬로 표현하면 그림 4와 같다. 인접행렬로 표현한 경우 각 배열에 들어가는 원소는 마디의 경우 각 마디 개체의 주소가 들어간다. 가지를 위한 배열에는 각 원소에 가지 개체의 주소가 들어간다.

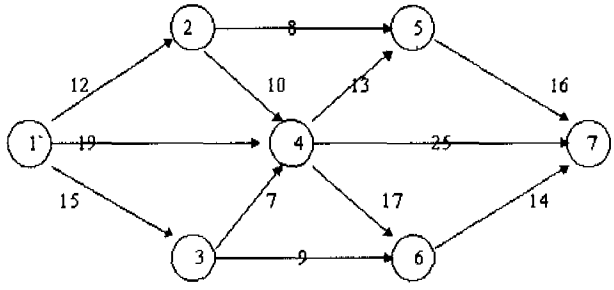


그림 3. 네트워크 모형의 예

마디 배열

0	&1
1	&2
2	&3
3	&4
4	&5
5	&6
6	&7

가지 배열

	0	1	2	3	4	5	6
0	/0	&(1, 2)	&(1,3)	&(1,4)	/0	/0	/0
1	/0	/0	/0	/0	&(2, 4)	&(2, 5)	/0
2	/0	/0	/0	&(3, 4)	/0	&(3, 6)	/0
3	/0	/0	/0	/0	&(4, 5)	&(4, 6)	&(4, 7)
4	/0	/0	/0	/0	/0	/0	&(5, 7)
5	/0	/0	/0	/0	/0	/0	&(6, 7)
6	/0	/0	/0	/0	/0	/0	/0

그림 4. 인접행렬로 표현한 네트워크 모형

(2) 연결리스트

연결리스트(linked list)는 각 노드에 다음 노드의 주소를 기억하는 공간을 주어 노드들을 연결하는 저장방법이다. 배열과는 달리 데이터가 기억장소내의 어느 곳에나 위치할 수 있다. 또한 노드의 삽입과 삭제가 요구될 때 리스트의 다른 데이터의 위치이동이 필요 없어 데이터 처리에서 유연성을 가진다. 따라서 프로그램의 실행시간, 저장장소의 사용량 및 프로그램의 복잡성이 감소된다. 그러나 배열의 크기와 동일한 자료의 크기에서 노드의 연결을 위한 주소가 추가적으로 저장되어야 하고, 원하는 데이터의 처리를 위해 다음 노드의 주소를 따라 경로를 추적하여 찾아야 하는 단점도 있다.

일반적으로 연결리스트는 단순 연결리스트와 이중 연결리스트를 많이 사용한다. 본 연구에서는 단순 연결리스트를 사용하여 전산구현하므로 마디의 수에 대한 제한을 두지 않는다.[5, 12] 그림 3을 연결리스트로 표현하면 그림 5와 같다.

본 연구에서는 연결리스트를 이용하여 전산구현한다. 네트워크를 연결리스트로 구현하기

위해 Graph 클래스의 개체에 마디 개체의 주소로 head와 tail을 멤버 데이터로 선언한다. 또한 마디 개체에는 각 마디에서 나가는 가지를 연결하기 위한 가지 개체의 주소를 first_arc로 선언한다. 네트워크에 대한 개체는 그림 6과 같다.

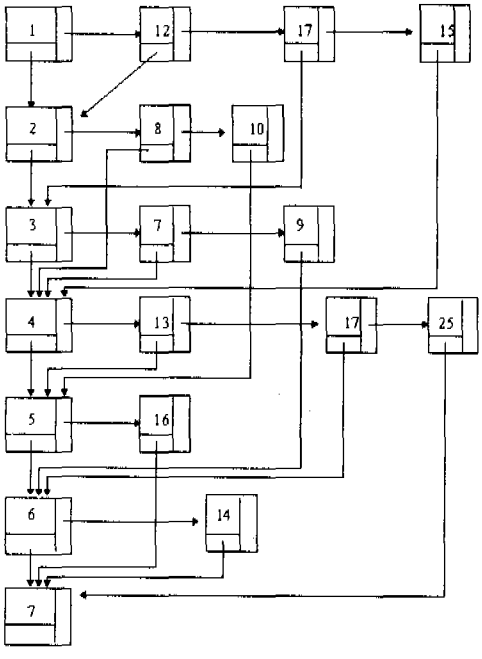


그림 5. 연결리스트로 표현한 네트워크 모형

```
class Graph: public CObject
{
private:
    ...
public:
    CNode* head;     // 연결리스트의 head 주소
    CNode* tail;     // 연결리스트의 tail 주소
public:
    Network(){};
    CEdge*  create_arc(void);  // 가지 생성 함수
      ...
};
```

그림 6. 네트워크 클래스

2. 네트워크 모형의 화면표현

본 연구에서 개발하고자 하는 전산패키지의 윈도우와 메뉴 그리고 대화상자 등은 Visual C++의 MFC에서 제공하는 클래스들을 이용한다. MFC를 이용하게 되면 개발자는 실질적인 네트워크 전산구현과 해법적용에 관한 작업에만 몰두할 수 있다. [9]

2.1 초기 메뉴

그림 7은 본 연구의 초기화면으로 주메뉴를 보여 준다. "파일"메뉴에서 새로운 윈도우를 만들거나 저장을 할 수 있다. 또한 저장형식을 결정할 수 있다. "입력양식" 메뉴는 정보의 입력형태를 나타낸다. 이 메뉴에 의해 화면으로 입력을 할 수도 있고 파일로도 입력할 수 있다. "그리기" 메뉴는 입력할 경우 마디의 생성, 삭제, 이동 그리고 가지의 생성과 수정 및 삭제를 할 때 사용되는 메뉴이다. "문제" 메뉴는 적용할 네트워크 문제의 종류를 결정한다.

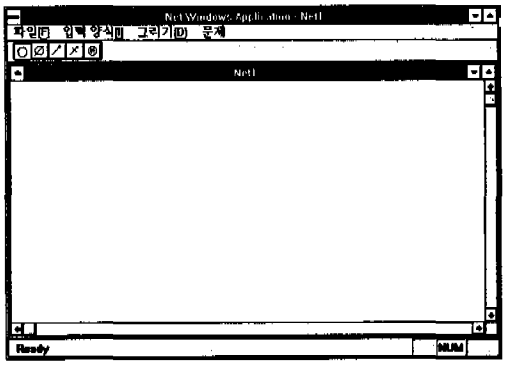


그림 7. 초기화면

2.2 마디생성 화면

마디를 화면에 보이기 위해 먼저 "그리기" 메뉴나 아이콘에서 마디 생성을 선택한다. 그리고 마디가 위치할 장소에서 왼쪽 마우스 버튼을 두 번 클릭한다. 마우스 클릭이 끝나면 그림 8에 보이는 것처럼 마디이름을 입력하기 위한 대화상자가 보인다. 대화상자에 마디이름을 입력한 후 확인 버튼을 누르면 하나의 마디가 생성된다.

생성된 마디를 삭제하고자 할 때, "그리기" 메뉴나 아이콘에서 마디 삭제를 선택하고 삭제하고자 하는 마디에서 왼쪽 마우스 버튼을 두 번 클릭한다. 또한 마디의 위치를 바꾸고자 할 때도 마디 이동을 선택하고 원하는 마디를 왼쪽마우스 버튼으로 눌러 원하는 지점에서 마우스 버튼을 놓는다.

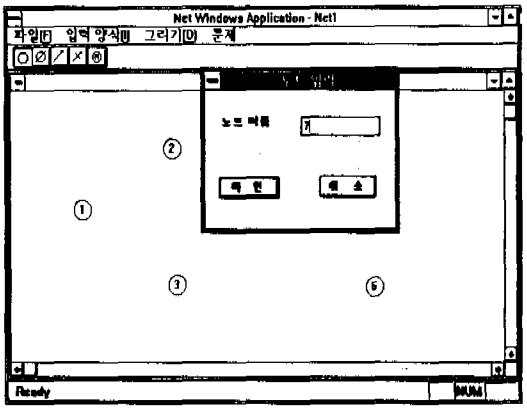


그림 8. 마디생성 화면

2.3 가지생성 화면

네트워크에 필요한 마디가 생성된 후 이제 각 마디를 연결하는 가지를 생성하여야 한다. "그리기" 메뉴나 아이콘에서 가지생성을 선택한 후, 가지의 시작마디에서 왼쪽마우스를 누른 채 마우스를 끌어, 가지가 끝나는 마디에서 마우스 버튼을 놓는다. 마우스를 놓는 순간 그림 9와 같은 가지의 값을 원하는 대화 상자가 보인다. 값을 입력하고 확인 버튼을 누르면 원하는 가지가 보인다.

생성된 가지의 삭제나 수정을 원할 때, "그리기" 메뉴나 아이콘에서 가지 변경을 선택하고 수정이나 삭제를 원하는 가지에서 왼쪽 마우스 버튼을 두 번 클릭하면 수정과 삭제를 원하는 대화상자가 나온다. 수정의 경우 그림 9와 같은 새로운 가지 값을 원하는 대화 상자를 보여 준다. 그림 10은 모든 마디와 가지가 생성된 하나의 네트워크를 보여 준다.

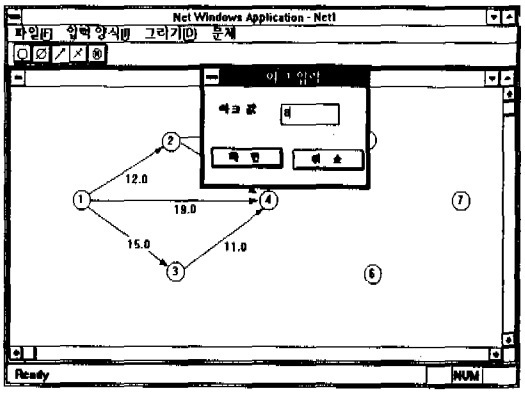


그림 9. 가지생성 화면

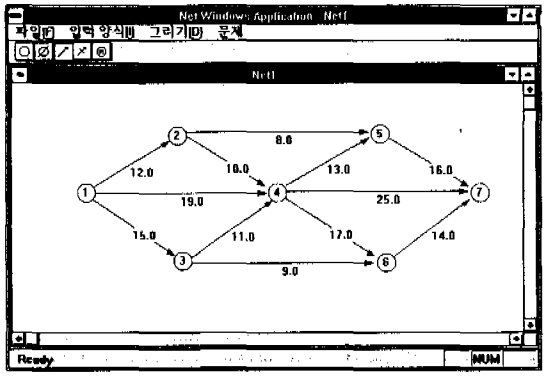


그림 10. 네트워크 모형

3. 네트워크 문제의 적용

사용자가 원하는 네트워크 모형이 완성되고 나면 사용자는 네트워크 문제의 해법을 적용시킬 수 있다. 본 연구에서는 네트워크 문제에서 기본적인 최단경로나무문제, 최소걸침나무문제, 최대흐름문제 세 가지 문제를 다룬다.

최단경로나무문제(Shortest Path Tree Problem)는 네트워크에서 시작마디로부터 다른 모든 마디까지 가장 짧은 경로를 찾는 것이다. 본 연구에서는 가지의 값이 비음인 경우만 다룬다. 또한 최단경로나무문제의 최적해를 구하는 많은 해법 중에서 Dijkstra 해법을 적용한다.

최소걸침나무문제(Minimum Spanning Tree Problem)는 네크워크 상에서 모든 마디를 연결하는 방법 중 전체의 비용 또는 길이가 최소가 되는 연결방식을 찾는 문제이다. 본 연구에서는 여러 해법 중 Prim 해법을 적용하며, 방향성이 없을 경우 마디에서 다른 한 마디로 나가고 들어오는 값은 동일한 경우로 간주한다.

최단경로나무문제와 최소걸침나무문제의 해법으로 실행시간의 단축을 위해 heap 개념을 사용한다. heap에 의한 방법이 항상 우수하다고 할 수는 없지만, 정렬을 할 때 계산시간 복잡도(time complexity)가 우수하다는 것이 기존의 연구결과 입증된 바 있다.

최대흐름문제(Maximal Flow Problem)는 네트워크에서 원점과 종점이 각각 한 개씩 존재하고, 이들 두 지점을 연결하는 여러 개의 경로에 유량의 제한이 있는 경우, 원점으로부터 종점까지 보내는 최대 유량을 구하는 문제이다. 원점에서 종점까지 수송하는 최대물량이나 정보통신 네트워크상의 두 지점 사이에서 교신할 수 있는 최대정보흐름 등의 문제들이 이 부류에 속하는 문제이다. 본 연구에서는 최대흐름문제의 해법으로 shortest augmenting path 해법을 적용한다. [7]

3.1 최단경로나무문제(Shortest Path Tree Problem)

"문제" 메뉴에서 최단경로나무문제를 선택하면, 그림 11과 같은 대화상자가 보인다. 그림 11에 시작마디의 이름을 입력하고 확인 버튼을 누르면 그림 12와 같은 결과가 화면에 나타난다.

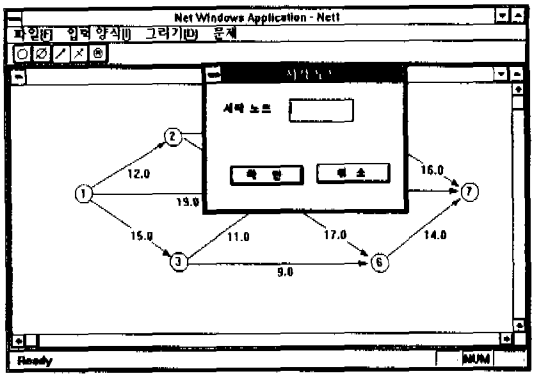


그림 11. 최단경로나무문제와 최소걸침
나무문제의 입력화면

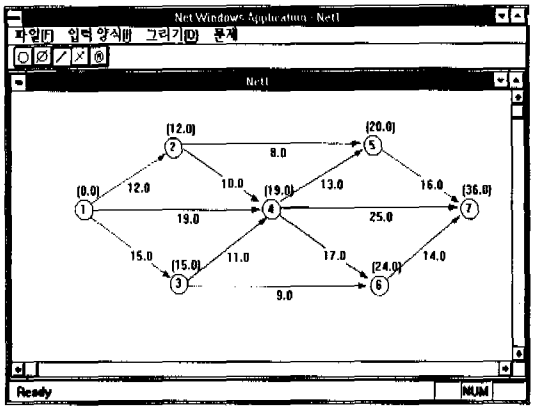


그림 12. 최단경로나무문제의 최적해

3.2 최소걸침나무문제(Minimum Spanning Tree Problem)

최소걸침나무문제의 적용과정은 최단경로나무문제와 동일하다. 그림 13은 최소걸침나무

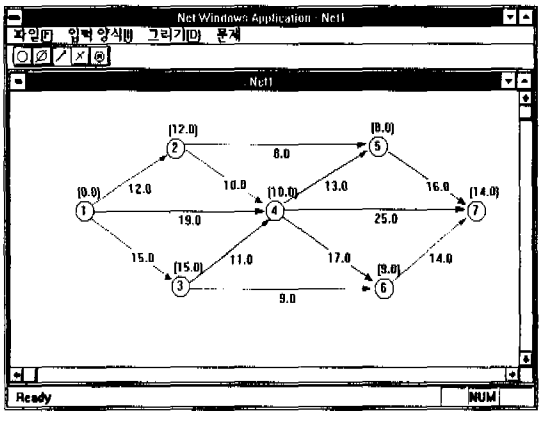


그림 13. 최소걸침나무문제의 최적해

문제의 최적해이다. 본 연구에서 사용된 자료구조가 단순 연결리스트이며 최소걸침나무 문제의 해법으로 Prim 해법을 이용하였기 때문에 문제적용에 제한을 둔다. 즉 두 마디사이의 가지가 양방향인 경우 두 가지의 값은 동일해야 한다.

3.3 최대흐름문제(Maximal Flow Problem)

"문제" 메뉴에서 최대흐름문제를 선택하면, 그림 14와 같은 대화상자가 보인다. 그림 14의 대화상자에 출발점과 도착점의 이름을 입력하고 확인 버튼을 누르면 그림 15와 같은 최대흐름문제의 최적해를 보여 준다.

그림 12, 13에서 두꺼운 선으로 보이는 가지는 최적해를 보이는 것으로 실제 화면상에서는 다른 색으로 표시된다. 최단경로문제의 최적해에서는 시작마디에서 각 마디까지의 최소거리를 보인다. 최소걸침나무문제에서는 각 마디로 들어오는 최소거리를 나타낸다. 또한 최대흐름문제의 경우, 그림 15에서 출발점과 도착점을 다른 색으로 표현하여 시각화하고 각 가지의 흐름량을 보여 준다.

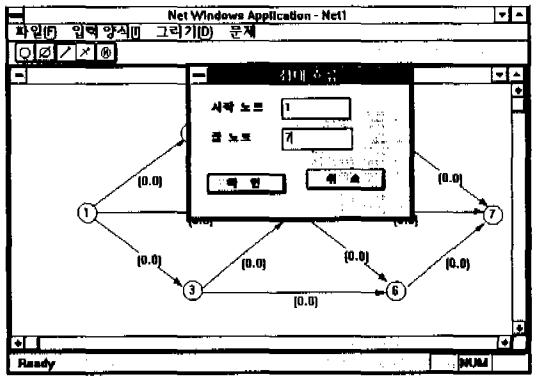


그림 14. 최대흐름문제의 입력화면

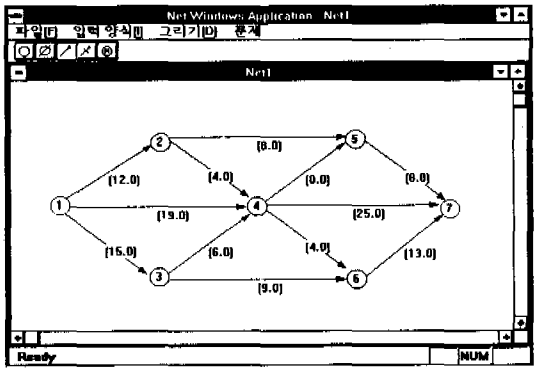


그림 15. 최대흐름문제의 최적해

4. 네트워크 모형의 파일 입출력

네트워크 모형의 입력과 출력 방법에는 세 가지 형태를 고려한다. 즉, 화면 입력과 dense형 파일 입력 그리고 sparse형 파일 입력의 세 가지 기본적인 형태를 생각한다. 출력은 그림, dense형 파일, sparse형 파일로 출력하는 세 가지 방법을 제시한다. 단, 그림 출력은 화면 입력의 경우에 한한다. 따라서 사용자가 기존의 dense나 sparse형의 데이터 파일을 사용할 수 있으며, 또한 문제적용의 결과를 사용자 자신의 dense형이나 sparse형의 데이터 파일로 저장할 수 있다.

화면에 의한 입력은 앞에서 설명한 방법에 의해 이루어진다. 여기에서는 dense형 파일 입출력과 sparse형 파일 입출력에 대해 설명한다.

4.1 파일 입력

dense형 데이터 파일은 가지의 시작마디와 끝마디를 행과 열로 정하여 가지의 값을 표현한 데이터형이다. 또한 sparse형 데이터 파일은 가지의 시작마디의 이름, 끝마디의 이름 그리고 가지의 값으로 표현된 데이터형이다.

dense형 또는 sparse형으로 저장된 데이터 파일을 입력하여 네트워크 모형을 전산구현하고자 할 때, 그림에서 원하는 양식을 선택한다. dense형이나 sparse형의 입력양식이 선택되고 나면, 그림 16과 같은 대화상자가 보인다. 그림 16에 입력하고자 하는 파일명과 경로를 주면 내부적으로 하나의 네트워크가 전산구현되고 그림 17, 18과 같은 화면이 보인다. 또한 문제의 결과를 파일로 저장하고자 할 경우 "파일" 메뉴에서 "문제결과저장"을 선택하고 그림 16의 대화상자를 통해 파일명과 경로를 입력하면 된다. 그림 21은 파일로 된 데이터를 dense형으로 화면에 출력한 결과이고, 22는 sparse형으로 출력한 결과이다.

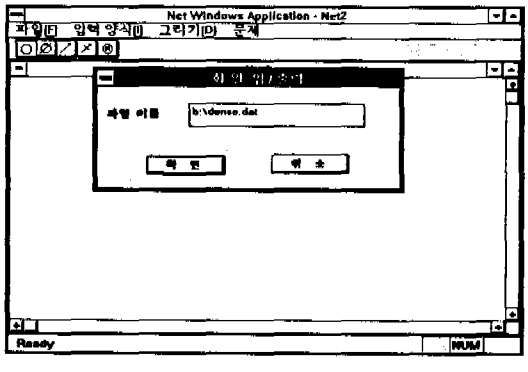
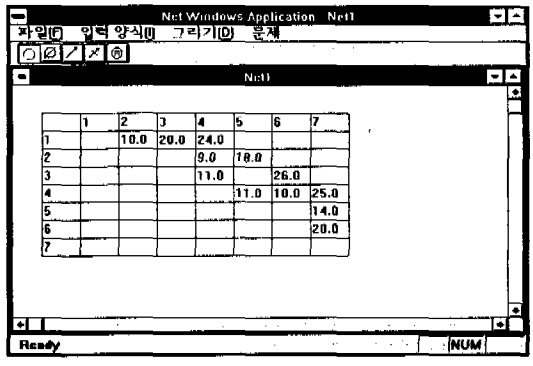


그림 16. 파일 입/출력의 파일명 삽입화면

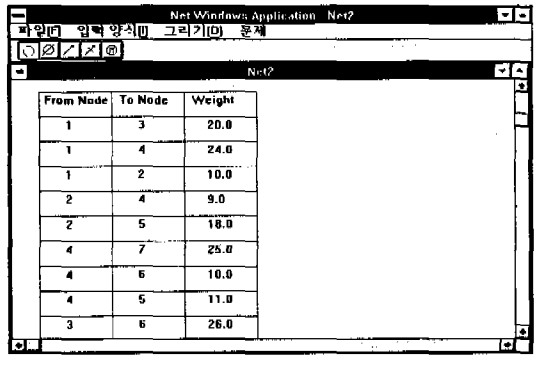
4.2 파일 출력

내부적으로 전산구현된 네트워크 모형을 dense형과 sparse형으로 저장하고자 할 때 "파일"메뉴에서 저장하고자 하는 형식을 선택한다. 저장형식이 선택이 되고 나면 그림 16의



	1	2	3	4	5	6	7
1		10.0	20.0	24.0			
2				9.0	18.0		
3				11.0		26.0	
4					11.0	10.0	25.0
5							14.0
6							20.0
7							

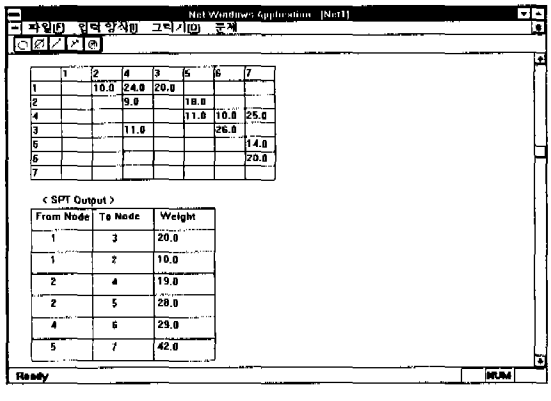
그림 17. Dense형 파일 입력자료 화면



From Node	To Node	Weight
1	3	20.0
1	4	24.0
1	2	10.0
2	4	9.0
2	5	18.0
4	7	25.0
4	6	10.0
4	5	11.0
3	6	26.0

그림 18. Sparse형 파일 입력자료 화면

대화상자가 보인다. 그림 16에 저장하고자 하는 파일명과 경로를 입력하고 확인 버튼을 누르면 원하는 형식으로 저장된다. 그림 19, 20은 입력이 dense형과 sparse형으로 된 파일 입력의 경우 문제가 적용된 결과를 화면으로 나타내 보인다.



	1	2	4	3	5	6	7
1		10.0	24.0	20.0			
2			9.0		18.0		
4					11.0	10.0	25.0
3			11.0			26.0	
5							14.0
6							20.0
7							

< SPT Output >

From Node	To Node	Weight
1	3	20.0
1	2	10.0
2	4	19.0
2	5	28.0
4	6	29.0
5	7	42.0

그림 19. Dense형 화면출력과 문제적용 결과

5. 결 론

본 연구에서는 윈도우즈 환경에서 네트워크 모형을 개체지향적으로 전산구현하였다. 사용자 정의 개체로 마디개체, 가지개체, 그리고 네트워크를 위한 개체를 정의하였다. 또한 네트워크를 위한 내부적 자료구조는 단순 연결리스트를 사용하였다.

모든 작업은 메뉴를 통하여 할 수 있도록 설계하였으며, 작업 메뉴는 파일, 입력방식, 그리기, 문제선택 및 적용으로 구성하였다. 자료의 입력 방식으로 화면을 통한 입력과 dense형 파일 및 sparse형 파일에 의한 입력의 세 가지 방식을 제공하였다. 출력 방식으로는 화면을 통한 시각적 출력으로 현실감을 주었으며, dense형 파일 및 sparse형 파일로 저장할

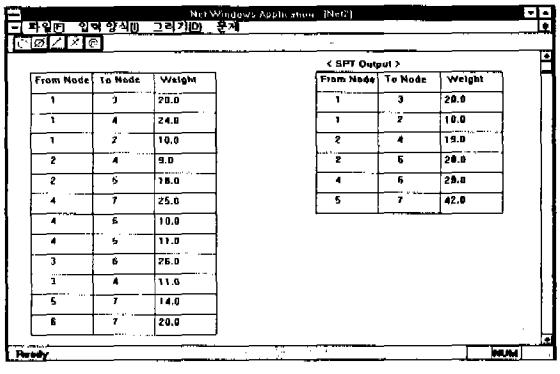


그림 20. Sparse형 화면출력과 문제적용 결과

수 있게 하여 사용자의 환경에 맞도록 하였다.

전산구현에 대한 응용문제로서 최단경로나무문제, 최소걸침나무문제, 최대흐름문제를 적용하였다. 또한 네트워크 모형이 개체지향 개념으로 구현되었으므로 사용자가 원하는 어떤 문제라도 서브루틴으로 적용할 수 있다.

기존의 연구에서는 네트워크 모형이 수정 또는 변경되었을 때 전산모형을 바꾸는데 있어 상당한 어려움을 겪었으며, 다른 사람이 그러한 전산모형을 사용할 때 어떻게 작동하는지 이해하기가 어려웠다. 그러나 개체지향 프로그래밍에서는 개체가 하나의 클래스로 이루어져 개체의 데이터와 프로시져가 함께 정의되어 있으므로 프로그램의 이해, 변경, 확장이 용이하다.

또한 네트워크 모형과 관련된 상용 S/W는 제한된 몇 가지만의 네트워크 문제가 적용되었으나 본 연구에서는 개체지향적으로 프로그래밍을 하였기 때문에 네트워크 문제와 관련된 대부분의 문제를 적용할 수 있다.

본 연구의 결과로서 얻을 수 있는 효과는 다음과 같다.

1) 네트워크 연구자는 프로그래밍에 소비하는 시간을 획기적으로 단축함으로써 본연의 연구에 전념할 수 있게 된다.
2) 네트워크 자료 입력양식을 몇 가지 제공함으로써 사용자는 자기에 맞는 양식을 고를 수 있게 된다. 이렇게 되면 자료의 입력에 따르는 번거로움과 노력을 줄일 수 있다.

3) 프로그램의 내부구조를 모르더라도 사용자는 쉽게 입력 및 작업지시를 할 수 있다.
4) 윈도우 화면에서 메뉴 형태로 제공하기 때문에 사용자는 쉽게 입력 및 작업지시를 할 수 있다.
5) 비슷한 주제를 연구하는 경우 같은 구현방식을 택함으로써 연구를 상호 비교할 수 있다
6) 본 연구의 결과로서 제공되는 몇 가지의 네트워크 기본해법을 사용자는 적은 노력으로 이용할 수 있다.
7) 본 연구의 한 결과로 네트워크 모형을 윈도우에서 시각적으로 보여줌으로써 사용자는 현실감을 느낄 수 있다.

본 연구에 대한 확장으로 다음과 같은 것들을 들 수 있다.
1) 네트워크문제 적용의 다양화 : 본 연구에서는 최단경로나무문제, 최소걸침나무문제, 최대흐름문제만을 적용했으나, 다른 네트워크문제도 적용한다.
2) 가지의 변화 : 본 연구에서는 가지가 직선으로만 표현되어 있어 가지 사이에 마디가 있을 경우 그 마디를 지나게 되어 있다. 이런 경우 중간 마디를 피할 수 있도록 해야 한다.
3) dense형과 sparse형 자료의 시각적 표현 : dense형과 sparse형으로 입력된 데이터를 화면에서 입력하는 것과 같이 그래픽으로 표현하도록 한다.
4) 이중 연결리스트의 사용 : 본 연구에 사용된 해법은 단순 연결리스트로 적용이 가능하지만 새로운 문제나 해법을 적용하기 위해서는 이중 연결리스트를 사용하여 자료구조의 적용을 보편화한다.

참 고 문 헌

[1] 김세헌, 현대경영과학, 무역경영사, 1993.
[2] 김형주, 알기 쉬운 객체지향시스템, 동아출판사, 1993.
[3] 방재희 역, Inside Visual C++, 영진출판사, 1994.
[4] 최진성 역, C++ 프로그래밍 테크닉과 응용, 동일출판사, 1992.
[5] 한상영, 자료구조론, 영지문화사, 1990.
[6] A.V.Aho, J.E.Hopcroft, and J.D.Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.
[7] R.K.Ahuja, T.L.Magnanti, and J.B.Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall Editions, 1992.
[8] M.S.Bazaraa, J.J.Jarvis, and H.D.Sherall, Linear Programming and Network Flows, John Wiley & Sons, 1990.
[9] A.I Holub, C+ C++ Programming with Objects in C and C++, McGraw-Hill, 1992.
[10] S. Holzner, VISUAL C++ Programming, IDG BOOKS, 1995.
[11] S. Shlaer and S. J. Mellor, Object Lifecycle, Yourdon, 1992.
[12] A.M.Tenenbaum, Y.Langasm, M.J.Augenstein, Data Structures Using C,

Prentice-Hall, 1990.
[13] M. J. Young, Microsoft Visual C++ Programming, Sybex Inc. , 1993.