МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій
Кафедра систем штучного інтелекту

Лабораторна робота №1

з курсу “Дискретна математика ”

Виконав:
ст. гр. КН-110
Чорній Юрій

Викладач:
Мельникова Н.І.

Львів – 2018

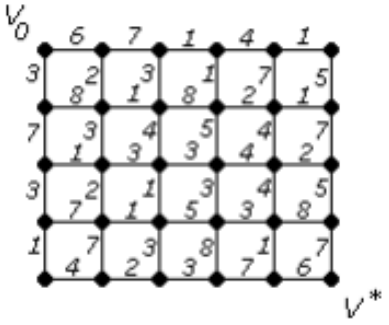
Варіант 15

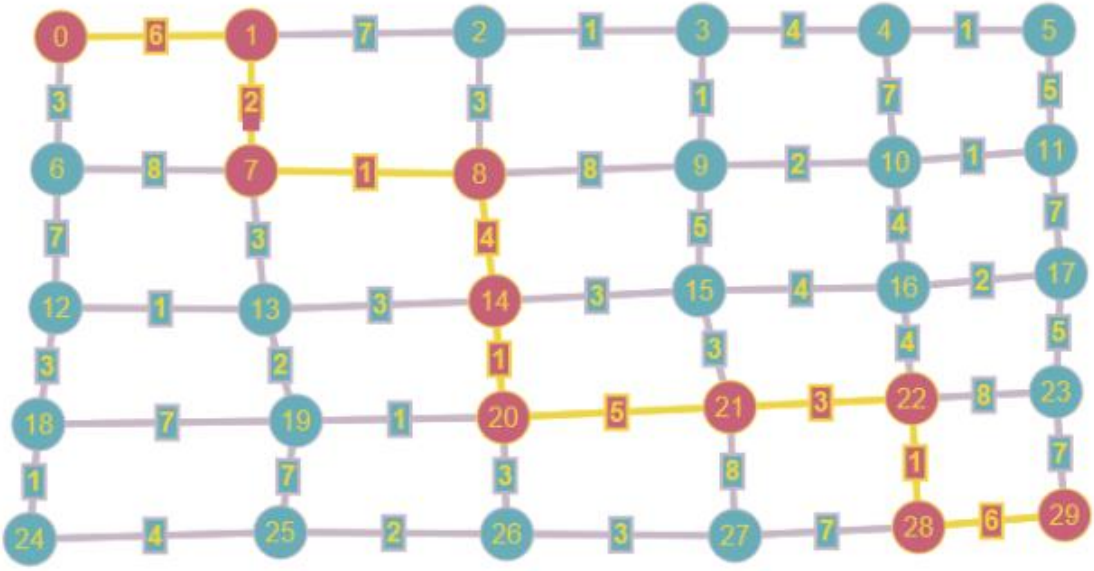
Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи

Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

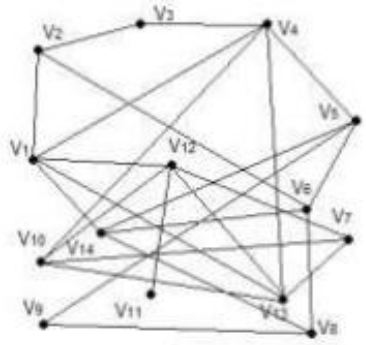
Завдання № 1. Розв'язати на графах наступні 2 задачі:

1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V_1 і V^*.

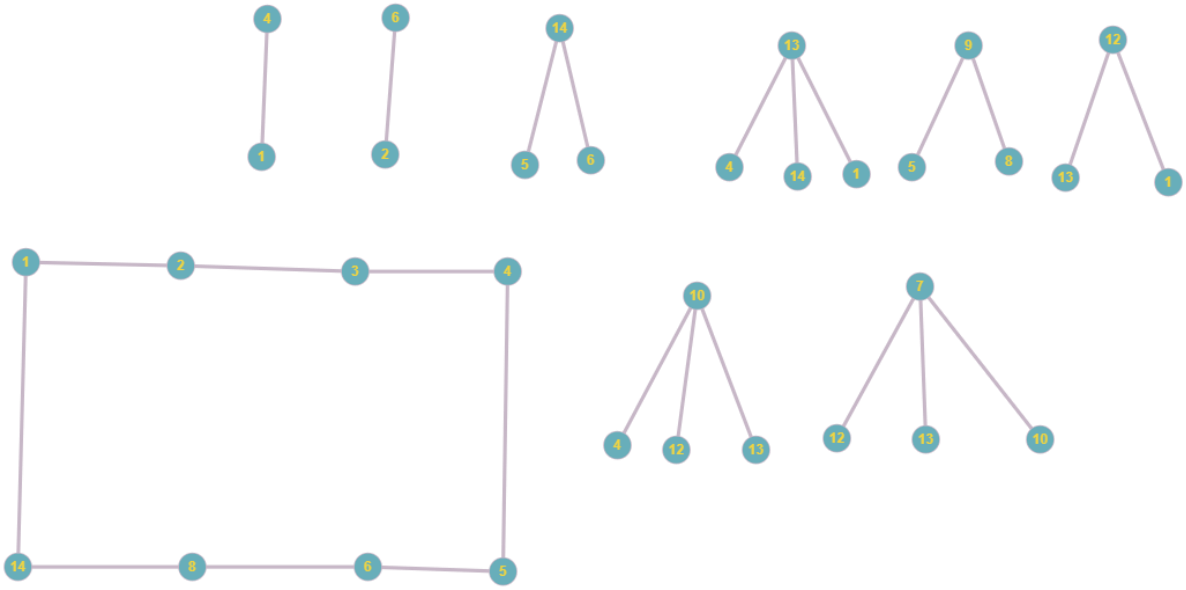


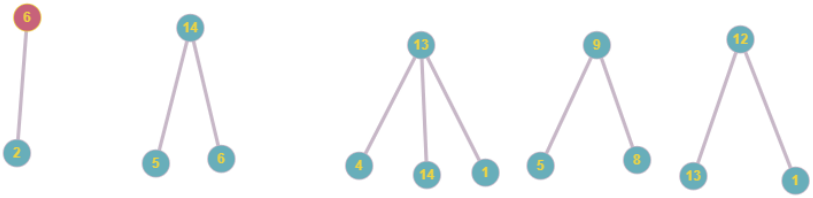


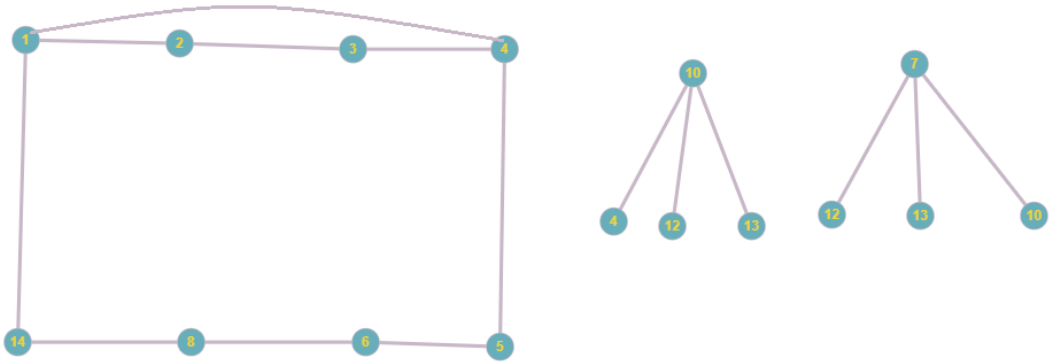
2. За допомогою γ -алгоритма зробити укладку графа у площині, або довести що вона неможлива.

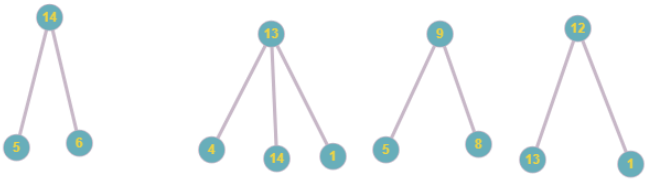


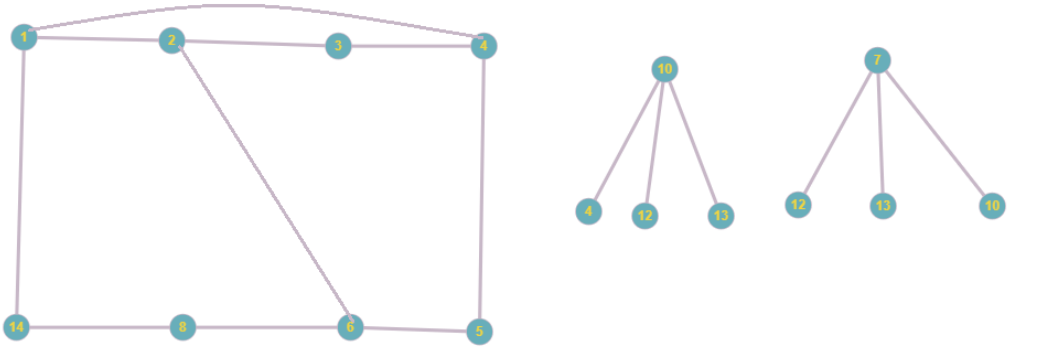
.

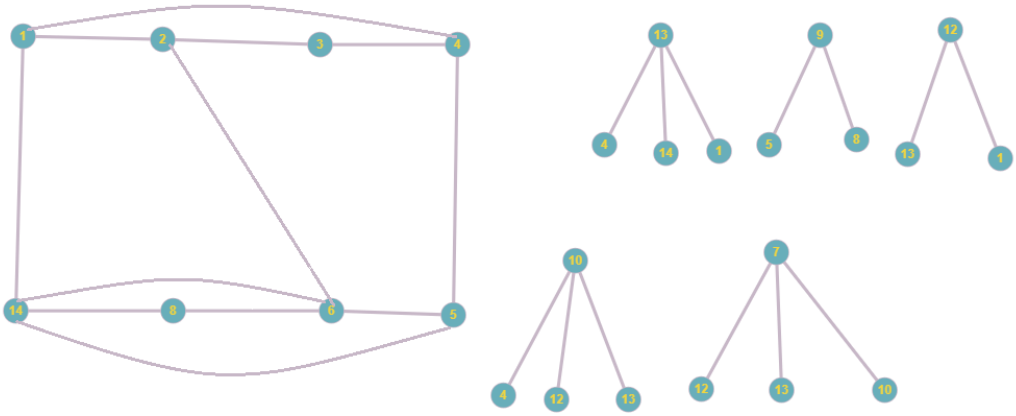


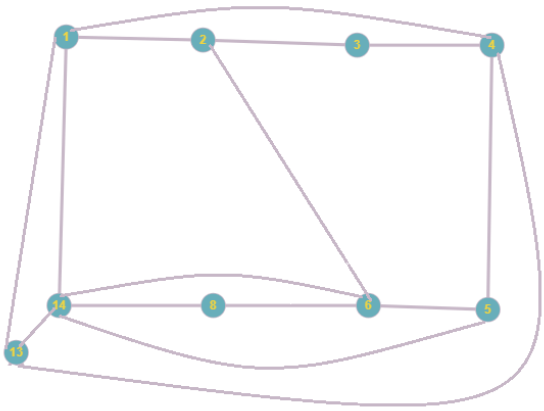

6
2
14
5
6
13
4
14
1
9
5
8
12
13
1

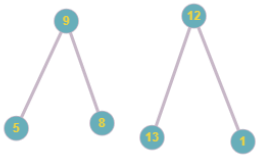

1
2
3
4
14
8
6
5
10
4
12
13
7
12
13
10


14
5
6
13
4
14
1
9
5
8
12
13
1

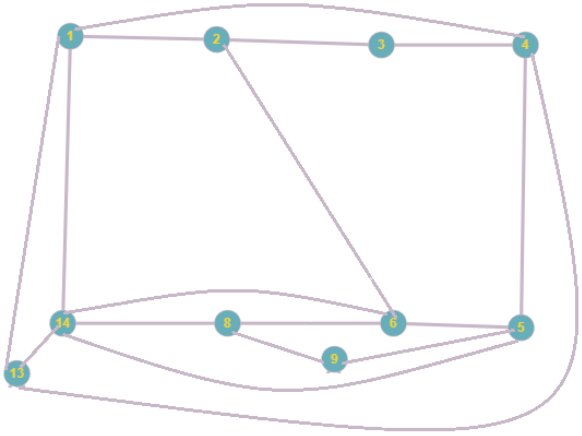

1
2
3
4
14
8
6
5
10
4
12
13
7
12
13
10

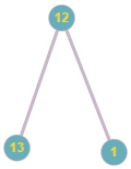

1
2
3
4
14
8
6
5
13
4
14
1
9
5
8
12
13
1
10
4
12
13
7
12
13
10


1
2
3
4
14
8
6
5
13

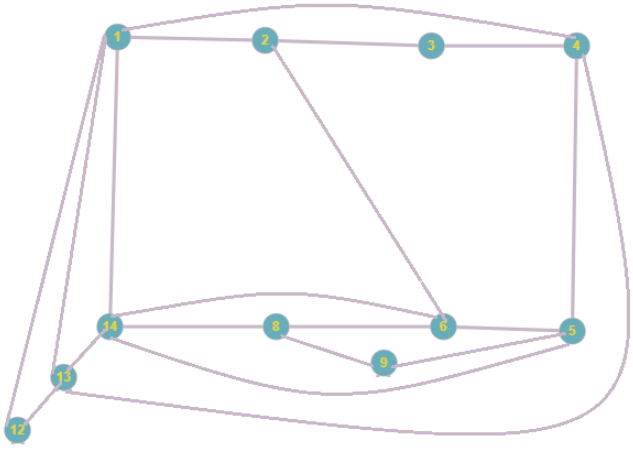

9
5
8
12
13
1


10
4
12
13
7
12
13
10

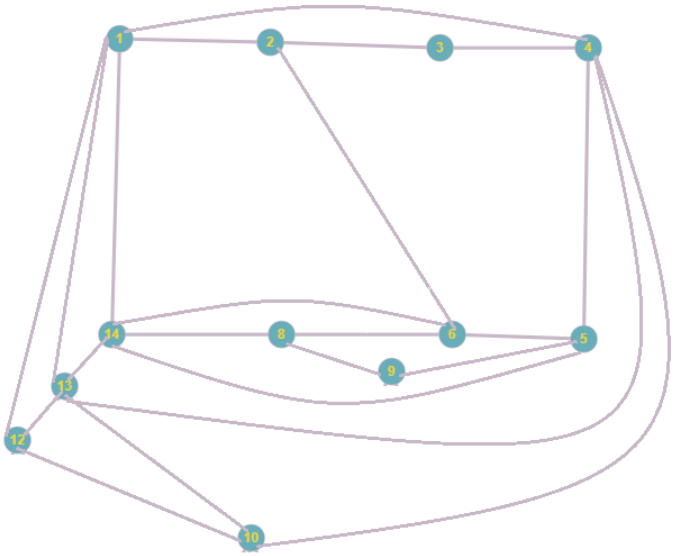

1
2
3
4
14
8
6
5
9
13

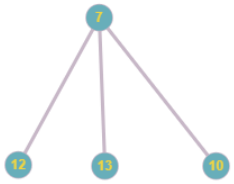

12
13
1

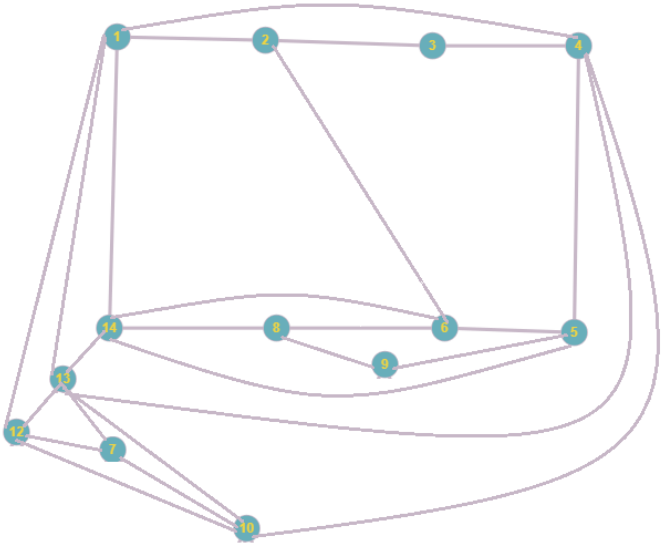

10
4
12
13
7
12
13
10


1
2
3
4
14
8
6
5
9
13
12


10
4
12
13
7
12
13
10


1
2
3
4
14
8
6
5
9
13
12
10


7
12
13
10


1
2
3
4
14
8
6
5
9
13
12
7
10

Додаток

Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Ver{
   int vertic;
   int number;
   struct Ver *next;
   struct Edg *edg[100];
   int Edgnum;
}Ver;

typedef struct Edg{
   int vert1;
   int vert2;
   int weight;
   struct Edg *next;
   struct Ver *ver1;
   struct Ver *ver2;
}Edg;

Ver * head1=NULL;
Edg * head2=NULL;

void Deikstre_part1()
```

```
{
   for (int j=0;j<20; j++)
   {
    Ver *ptr = head1;
    Edg *ptr1 = head2;
    Ver *ptr2;
    ptr->number=0;
    ptr= ptr->next;
    while (ptr!=NULL)
     {

         for (int k=0;k<(ptr->Edgnum);k++)
        {
           int s1=0;
           ptr1=ptr->edg[k];
           s1+=ptr1->weight;
           if (ptr->vertic==ptr1->vert2)
              ptr2=ptr1->ver1;
           if (ptr->vertic==ptr1->vert1)
              ptr2=ptr1->ver2;

           s1+=ptr2->number;

           if(ptr->number>s1)
           ptr->number=s1
}
     ptr= ptr->next;
      }
```

```
   }
}

void Deikstre_part2(int ver)
{
    Ver *ptr = head1;
    Edg *ptr2 = head2;
    Ver *ptr1;

      while (ptr->vertic!=ver)
     {
      ptr= ptr->next;
      }
      ptr1=ptr;
      printf("\n[%d ",ptr1->vertic);
      while(ptr->vertic!=1)
      {
         ptr1=ptr;

          for (int k=0;k<(ptr1->Edgnum);k++)
           {
              ptr2=ptr1->edg[k];

              if (ptr1!=ptr2->ver1) ptr=ptr2->ver1;
              if (ptr1!=ptr2->ver2) ptr=ptr2->ver2;

             if (ptr2->weight+ptr->number==ptr1->number)
              {
```

```
                printf(" ,%d",ptr->vertic);
                break;
              }
            }

      }
printf("]\n ");
}
void printList()
{
   Edg *ptr = head2;
   printf("\n[ ");

   while(ptr != NULL)
   {
      printf("( 1-st Vertice: %d,2-st Vertice:%d, Weight: %d ) ",ptr->vert1,ptr->vert2,ptr->weight);
      ptr = ptr->next;

   }
  printf(" ]\n");

}

void printList1()
{
   Ver *ptr = head1;

   printf("\n[ ");
```

```
   while(ptr != NULL)
   {
      printf("( # of Vertice: %d number %d, number of ints. edg.: %d ) ",ptr->vertic,ptr->number,ptr->Edgnum);
      ptr = ptr->next;

   }
  printf(" ]\n");
}
int main(void)
{   int numver;
   printf("Input number of vertices:");
   scanf("%d",&numver);
   for (int i=0; i<numver;i++)
   {
      Ver *link = (Ver*)malloc(sizeof(Ver));
      link->next = head1;
      head1 = link;
      printf("Input # of vertices:");
      scanf("%d",&link->vertic);
      link->number=1000;
   }

   while (1)
   {
      int k;
      printf("If you want to add edge write 1, else write 0: ");
```

```
    scanf("%d",&k);

     if (k==1)
    {
        Edg *link = (Edg*)malloc(sizeof(Edg));
        link->next = head2;
        head2 = link;
        printf("Input # of 2  adjacent to edge and weight of edge:");
        scanf("%d %d %d",&link->vert1,&link->vert2,&link->weight);
    }
    else break;
}

Ver *ptr1 = head1;

while(ptr1 != NULL)
{
    Edg *ptr2 = head2;

    ptr1->Edgnum=0;
    while(ptr2 != NULL)
    {
    if (ptr1->vertic==ptr2->vert1)
    {

        ptr2->ver1=ptr1;
        ptr1->edg[ptr1->Edgnum]=ptr2;
        ptr1->Edgnum++;
```

```
        }
        if (ptr1->vertic==ptr2->vert2)
        {
           ptr2->ver2=ptr1;

           ptr1->edg[ptr1->Edgnum]=ptr2; ptr1->Edgnum++;
        }
        ptr2 = ptr2->next;
        }

        ptr1->edg[ptr1->Edgnum]=0;
        ptr1 = ptr1->next;
     }

     int de_num;
     printf("Input last verice for algo Deikstre: ");
     scanf("%d",&de_num);
printList();
Deikstre_part1();
printList1();
Deikstre_part2(de_num);
}
```