Теорія графів

Турнір «Графи»

1

Пошук в глибину

```
void p(int k, int v)

{c[k]=v;

if (v==v2 || k>=n) {

if (v==v2)

{

//for (int i=1;i<=k;i++)cout<<c[i]<<" ";cout<<endl;

int s=0;
```

```
for (int i=1;i<k;i++)s=s+a[c[i]][c[i+1]];
```

```
//cout<<"s="<<s<<endl;
```

```
if (s<m) m=s;
```

```
}
```

```
}
```

```
else
```

```
for (int i=1;i<=n;i++)
```

```
if (a[v][i]>0) p(k+1,i);
```

```
}
```

```
2
```

```
Флойда
```

```
k= 1;
```

```
    for(i= 1;i<=n;i++)x[i]=a[v1][i];

//    {инвариант: x[i] = МинСт(1,i,k)}

                                    while (k!=n)

                                    {

                                    for(s=1;s<=n;s++){

                                    y[s]=x[s];

                                    for(i=1;i<=n;i++) if (y[s]>x[i]+a[i][s]) y[s]= x[i]+a[i][s];

// {y[s] = МинСт(1,s,k+1)}

    for (i=1;i<=n;i++) x[s]=y[s];

                                    }

    k++;

                                    }
```

3

Форда

```
for (int k=1; k<=n; k++)
```

```
for (int i=1; i<=n; i++)
```

```
for (int j=1; j<=n; j++)
```

```
a[i][j] = min (a[i][j], a[i][k] + a[k][j]);
```

4

Прима

5

Дейкстри

Алгоритм дуже простий, його реалізація представлена нижче. Алгоритм працює з вершинами як

C - матриця довжин дуг. Її розмір C[p, p]. C[v1, v2] - містить довжину дуги від точки v1 до точки v2

s - початкова точка шляху в графі. Та, звідки треба знайти шлях в точку t.

t - кінцева точка шляху в графі. Це мета пошуку.

X - цей одновимірний масив служить сховищем міток, для вершин, в яких би ми вже проходили в

v - поточна вершина обходу (украй динамічна змінна)

////ініціалізація і опис початкового стану змінних

Перебір значень u від 1 до p:

{

	T[u] = ∞;
	X[u] = 0;

}

	//

H[s] = 0;	
T[s] = 0;	
X[s] = 1;	
v = s;	
Мітка M :	
// // далі перебираємо усі прилеглі до поточної вершини точки, і визначаємо для них	
// // мінімальна відстань, порівнюючи записане в масив T і шлях по	
// // поточній вершині.	
Перебір значень u від 1 до p:	
{	
	якщо X[u]=0 і T[u] > T[v]+C[v, u] те:
{	

```
                                                    T[u] =

                                                    H[u] = v;

}

}

// // далі шукаємо наступну точку для обходу, грунтуючись на її відстані від

// // почала. Для наступного циклу виберемо точки, що мають найменшою шлях,

// // і цей шлях вибирається, грунтуючись на даних масиву T, де довжина

// // шляхи враховує тільки довжину зв'язків, по яких вони йдуть. Т. о.

// // перший знайдений шлях від s до t і буде найкоротшим.

l = ∞;

v = 0;

Для змінної u значення від 1 до p:
```

```
{
    якщо X[u] = 0 і T[u] < l те:
    {
        v = u;
        l = T[u];
    }
}
Якщо v = 0 те:
{
    exit(); //немає шляху
} інакше
Якщо v = t те:
```

{	
	exit(); //шлях знайдений, наступна вершина і кінц
} інакше	
{	
	X[v] = 1;
	Перейти на мітку M;
}	

Якщо залишилися якісь порожні місця, рекомендую пройтися по алгоритму наново, і усвідомити й

Як бачите, алгоритм не зайде в пошуку далі, ніж треба, і за умови, що вершини лежать близько о

За допомогою алгоритму Дейкстры добре шукати конкретні одноразові шляхи. Але найчастіше на