Seminar 13

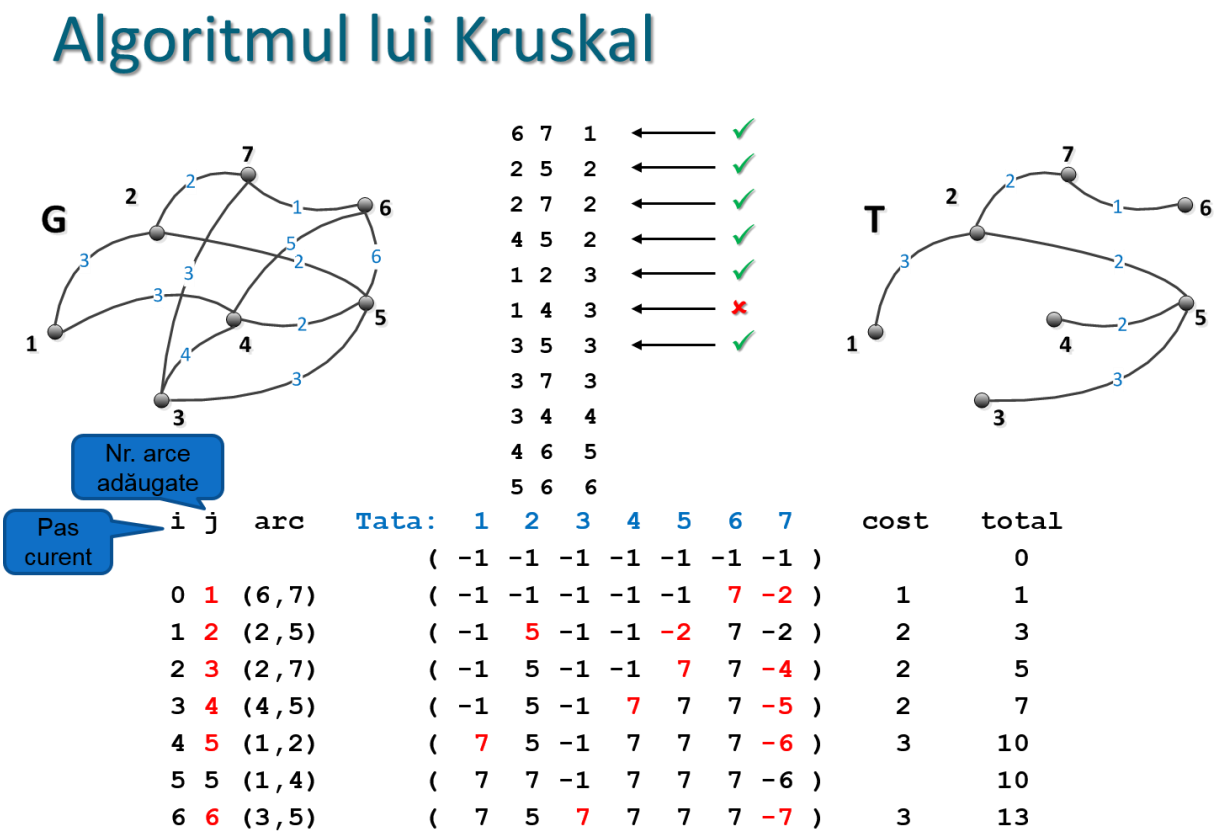
Arbori parțiali de cost minim: Kruskal, Prim

Arborele parțial de cost minim

Atât în algoritmul lui Prim cât și în algoritmul lui Kruskal putem identifica elementele generale specifice metodei Greedy astfel:

- există o mulțime inițială A din care se aleg elementele care vor compune soluția (A este mulțimea muchiilor grafului);
- fiecărui element al mulțimii A îi este asociată o valoare numerică (ponderea muchiei);
- mulțimea inițială se sortează crescător, în ordinea ponderilor asociate muchiilor;
- din mulțimea A se aleg, în ordine, primele $m-1$ elemente care nu formează cicluri (criteriul de alegere).

1. Algoritmului Kruskal: exemplu, explicații, implementare, exemplu de execuție



Pentru implementarea algoritmului Kruskal, graful conex ponderat este reprezentat sub formă tabelară, muchiile fiind ordonate crescător după ponderi. Muchiile selectate de algoritm pot fi menţinute de asemenea într-o structură tabelară, sau doar marcate ca fiind incluse în mulţimea de muchii din arborele parţial minim a cărui construcţie este dorită. În varianta prezentată în continuare muchiile selectate sunt afişate.

Verificarea condiţiei ca muchia selectată să nu formeze nici un ciclu cu muchiile selectate la etapele precedente este realizată prin utilizarea un vector TATA. Pentru fiecare vârf *i* (vârfurile grafului fiind numerotate de la 1 la *n*, unde *n* este numărul de noduri ale grafului), componenta TATA [i] este predecesorul său în arborele care conţine vârful *i* construit până la momentul curent dacă *i* nu este rădăcina acelui arbore, respectiv TATA[i] este egal cu *–numărul de vârfuri ale arborelui de rădăcină i*, în caz contrar componentele vectorului TATA sunt iniţializate cu valoarea -1.

Calculul care realizează adăugarea unei noi muchii poate fi descris astfel:
- Este determinată o muchie de cost minim e=v1v2 care nu a fost selectată anterior;
- Dacă vârfurile v1 şi v2 nu aparţin aceluiaşi arbore, atunci proprietatea de aciclicitate este îndeplinită şi muchia *e* este adăugată la structura curentă.
- Adăugarea muchiei *e* selectate este realizată prin reunirea arborilor din care fac parte *v1* şi *v2* de rădăcini *r1*, respectiv *r2*, astfel: dacă TATA[r1]<TATA[r2], atunci arborele rezultat prin reunirea celor doi arbori are ca rădăcină vârful *r1*, iar vârful *r2* devine fiu al lui *r1*. Altfel, rădăcina arborelui rezultat prin reunire fiind *r2*, iar r1 devenind fiu al rădăcinii.
- Calculul se încheie după ce a fost adăugată şi cea de-a (*n*-1)-a muchie.

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

int radacina(int v, int *tata)
{
	int u = v;
	while (tata[u] >= 0)
		u = tata[u];
	return u;
}

//a - tabela muchiilor in ordinea crescatoare a costurilor
//nm - numarul de muchii
//nv - numarul de noduri
int kruskall(int a[][3], int nm, int nv)
{
	int tata[50], i, j;
	int v1, v2, k, p, c = 0;

	//se pastreaza in tata[i] este parintele nodului i
	for (i = 0; i < nv; i++)
		tata[i] = -1;

	//Din teorie stim ca un arbore este un graf conex si aciclic. Un arbore cu N
noduri are N-1 muchii.
	//Vrem sa gasim in graful dat un arbore partial de cost minim
	//=> conditia de iesire este cand se gaseste a (n-1) muchie.

	for (j = i = 0; i < nv - 1; j++)
	{
```

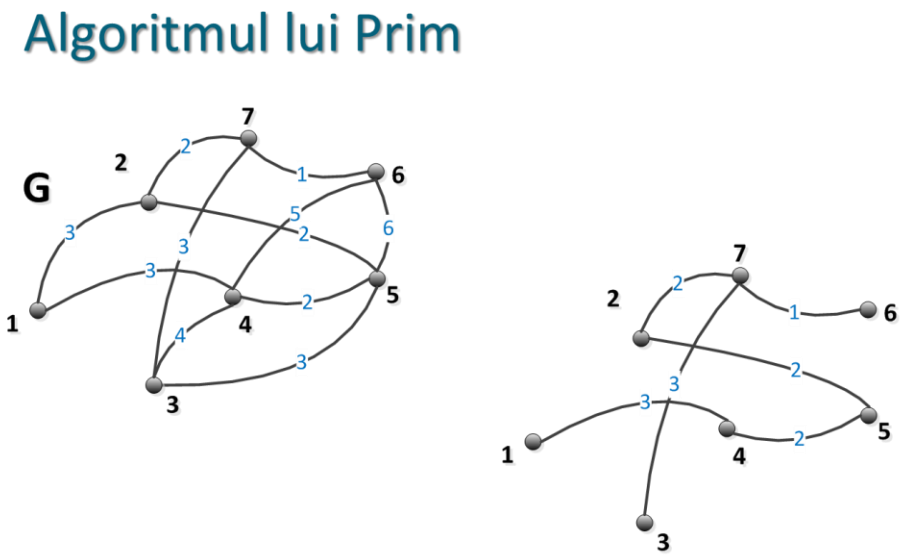
```
			//preluam varfurile din muchia "j"
			v1 = a[j][0];
			v2 = a[j][1];

			//determinam radacinile arborilor din vare fac parte nodurile v1 si v2
			k = radacina(v2, tata);
			p = radacina(v1, tata);

			if (k - p)//daca diferenta dintre radacini (k-p) != 0
			{
				if (tata[k] < tata[p])
				{
					tata[k] += tata[p];
					tata[p] = k;
				}
				else
				{
					tata[p] += tata[k];
					tata[k] = p;
				}
				c += a[j][2];
				printf("%i -> %i cost %i\n", v1 + 1, v2 + 1, a[j][2]);
				i++;
			}
		}
		return c;
}

void main()
{
		int cost, i, j, nv, nm, a[100][3];
		//graful este preluat de la tastatura
		//datele citite:numarul de virfuri, numarul de muchii si tabela muchiilor in
ordinea crescatoare a costurilor

		printf("Numarul de virfuri:"); scanf("%i", &nv);
		printf("Numarul de muchii:"); scanf("%i", &nm);

		//muchiile trebuie citite neaparat in ordine crescatoare in fct. de costuri!!!
		printf("Matricea de reprezentare\n");
		for (i = 0; i < nm; i++)
		{
			printf("Muchia %i si ponderea:", i + 1);
			for (j = 0; j < 3; j++)
				scanf("%i", &a[i][j]);
		}

		//se adapteaza fiecare muchie pentru a putea folosi si indexul 0. Ex: muchia
citita de la tastatura (2,3) devine (1,2)
		//<= dar ea va reprezenta in continuare muchia dintre nodul 2 si nodul 3

		for (i = 0; i < nm; i++)
			for (j = 0; j < 2; j++)a[i][j]--;

		printf("Arborele de cost minim: \n");
		cost = kruskall(a, nm, nv);

		printf("\ncu costul %i", cost);
		_getch();
}
```

Exemplu de executie:

```
Numarul de virfuri:7
Numarul de muchii:11
Matricea de reprezentare
Muchia 1 si ponderea:6 7 1
Muchia 2 si ponderea:2 5 2
Muchia 3 si ponderea:2 7 2
Muchia 4 si ponderea:4 5 2
Muchia 5 si ponderea:1 2 3
Muchia 6 si ponderea:1 4 3
Muchia 7 si ponderea:3 5 3
Muchia 8 si ponderea:3 7 3
Muchia 9 si ponderea:3 4 4
Muchia 10 si ponderea:4 6 5
Muchia 11 si ponderea:5 6 6
Arborele de cost minim:
6 -> 7 cost 1
2 -> 5 cost 2
2 -> 7 cost 2
4 -> 5 cost 2
1 -> 2 cost 3
3 -> 5 cost 3

cu costul 13_
```

2. Algoritmul lui Prim: exemplu, explicații, implementare și exemplu de execuție

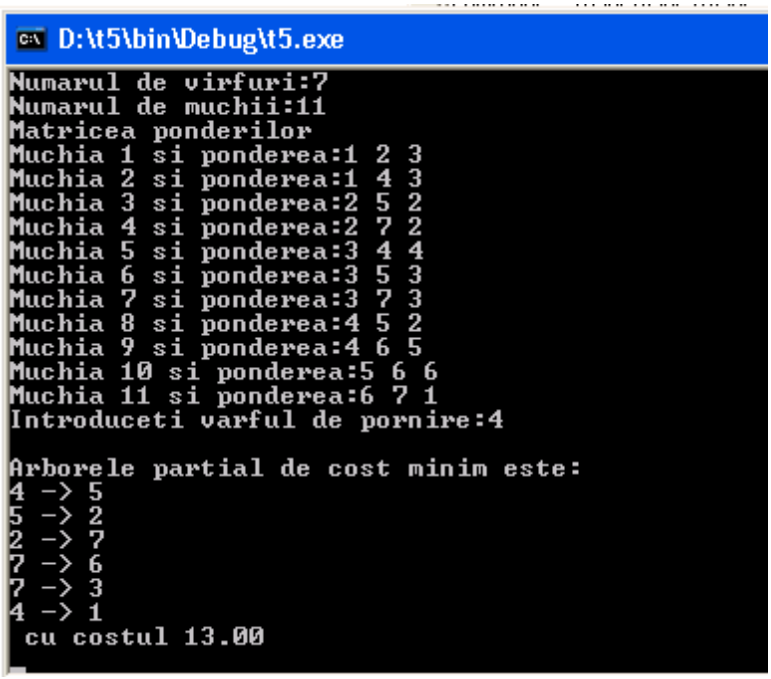
Exemplu:v0=4



Funcția care implementează algoritmul lui Prim primește matricea ponderilor (*w*), numărul de vârfuri ale grafului (*n*), un vârf inițial (*v0*), adresa unde depune tabela muchiilor selectate (*arb*), o valoare folosită pentru inițializare la căutarea muchiei cu cost minim (*MAX*). Prin numele funcției se întoarce costul arborelui construit.

```c
#include<stdio.h>
#include<conio.h>
#include<malloc.h>

//calculeaza costul minim
//w - matricea ponderilor
//n - nr de noduri
//v0 - vf. de start
//arb - se vor salva muchiile ce construiesc arborele de cost minim
//MAX - valoarea infinit
float Prim(float **w, int n, int v0, int ***arb, float MAX)
{
	float cost, cmin;
	int i, u, v, vf1, vf2;
	cost = 0;

	//ind este vectorul indicator:
	//ind[i]=1 daca varful i+1 a fost trecut in arbore, altfel este 0
	int *ind = (int*)malloc(n * sizeof(int));
	for (i = 0; i < n; i++)
		ind[i] = 0;

	//vf. de start este primul varf adaugat in arborele de cost minim
	ind[v0 - 1] = 1;

	//se construieste o matrice temporara de N linii si 2 coloane ce va stoca muchiile
care se vor adauga in arborele de cost minim
	int **muchii = (int**)malloc((n - 1) * sizeof(int*));
	for (i = 0; i < n - 1; i++)
		muchii[i] = (int*)malloc(2 * sizeof(int));


	for (i = 0; i < n - 1; i++)
	{
		cmin = MAX;
		for (u = 1; u <= n; u++)
			if (ind[u - 1])
			{
				for (v = 1; v <= n; v++)
					if ((!ind[v - 1]) && (w[u - 1][v - 1] <= cmin))
					{
						vf1 = u;
						vf2 = v;
						cmin = w[u - 1][v - 1];
					}
			}
		cost += cmin;
		muchii[i][0] = vf1;
		muchii[i][1] = vf2;
		ind[vf2 - 1] = 1;
	}

	//atribuire matrice temporara
	*arb = muchii;

	//eliberare memorie
	free(ind);

	return cost;
}
```

```
void main()
{
	int i, j, nv, nm, v, u, v0, **arb; float **w, MAX = 1000000, cost, p;
	//graful este preluat de la tastatura
	//datele citite:numarul de virfuri, numarul de muchii si matricea ponderilor
	printf("Numarul de virfuri:"); scanf("%i", &nv);
	printf("Numarul de muchii:"); scanf("%i", &nm);

	//alocare memorie pentru matricea ponderilor
	w = (float**)malloc((nv) * sizeof(float*));
	for (i = 0; i < nv; i++)
		w[i] = (float*)malloc(nv * sizeof(float));

	//initializarea matricii ponderilor
	printf("Matricea ponderilor\n");
	for (i = 0; i < nv; i++)
		for (j = 0; j < nv; j++)
			w[i][j] = MAX;

	//citire de la tastatura a matricii ponderilor
	for (i = 0; i < nm; i++)
	{
		printf("Muchia %i si ponderea:", i + 1);
		scanf("%i %i %f", &v, &u, &p);
		w[u - 1][v - 1] = w[v - 1][u - 1] = p;
	}

	printf("Introduceti varful de pornire:");
	scanf("%i", &v0);

	cost = Prim(w, nv, v0, &arb, MAX);

	printf("\nArborele partial de cost minim este:\n");
	for (i = 0; i < nv - 1; i++)
		printf("%i -> %i\n", arb[i][0], arb[i][1]);

	printf(" cu costul %4.2f\n", cost);


	getch();
}
```


D:\t5\bin\Debug\t5.exe
Numarul de virfuri:7
Numarul de muchii:11
Matricea ponderilor
Muchia 1 si ponderea:1 2 3
Muchia 2 si ponderea:1 4 3
Muchia 3 si ponderea:2 5 2
Muchia 4 si ponderea:2 7 2
Muchia 5 si ponderea:3 4 4
Muchia 6 si ponderea:3 5 3
Muchia 7 si ponderea:3 7 3
Muchia 8 si ponderea:4 5 2
Muchia 9 si ponderea:4 6 5
Muchia 10 si ponderea:5 6 6
Muchia 11 si ponderea:6 7 1
Introduceti varful de pornire:4
Arborele partial de cost minim este:
4 -> 5
5 -> 2
2 -> 7
7 -> 6
7 -> 3
4 -> 1
cu costul 13.00