Arbori parțiali de cost minim

Fișierul `grafpond.in` **are următoarea structură: numărul de vârfuri n, numărul de muchii m şi lista muchiilor cu costul lor (o muchie fiind dată prin extremităţile sale şi cost). Costul unei muchii este număr întreg.**

grafpond.in
5 7 1 4 1 1 3 5 1 2 10 2 3 2 4 2 6 4 5 12 5 2 11

1. Implementați algoritmul lui Kruskal pentru determinarea unui arbore parţial de cost minim al unui graf conex ponderat cu n vârfuri și m muchii. Graful se va citi din fişierul **`grafpond.in`**. **O(m log n)** (+ și versiunea **O(n^2 + m log n))**

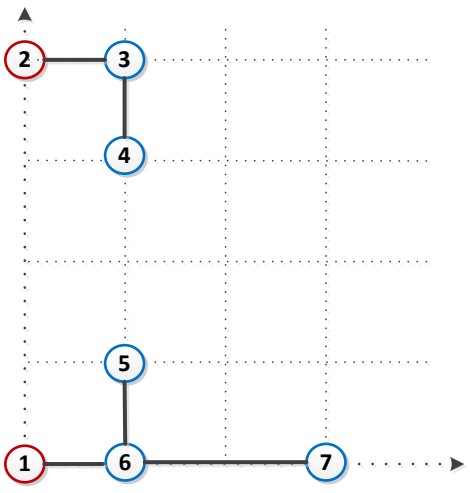
2. Implementați algoritmul lui Prim pentru determinarea unui arbore parţial de cost minim al unui graf conex ponderat cu n vârfuri și m muchii. Graful se va citi din fişierul **`grafpond.in`**. **O(m log n)** (+ și versiunea **O(n^2))**

3. **Clustering.** Fişierul `cuvinte.in` conţine cuvinte separate prin spaţiu. Se citeşte de la tastatură un număr natural k. Se consideră distanţa Levenshtein între două cuvinte https://en.wikipedia.org/wiki/Levenshtein_distance, calculată prin subprogramul `distanta` definit mai jos (detalii la cursul de Tehnici de programare).
   Să se împartă cuvintele din fişier în k clase (categorii) nevide astfel încât gradul de separare al claselor să fie maxim (= distanţa minimă între două cuvinte din clase diferite) - v. curs; se vor afişa pe câte o linie cuvintele din fiecare clasă şi pe o altă linie gradul de separare al claselor. **O(n^2 log n) / O(n^2)**

cuvinte.in	**Ieșire pentru k=3 (clasele nu sunt unice, dar gradul de separare da)**
martian care este sinonim ana case apa arbore partial minim	care este ana case apa arbore martian partial sinonim minim 4

```
int distanta(char *s1, char *s2){
	int n1=strlen(s1);  int n2=strlen(s2);
	int *ci1=new int[n2+1]; int *ci=new int[n2+1];
	for(int j=0;j<=n2;j++)     ci1[j]=j;
	for(int i=1;i<=n1;i++){
		ci[0]=i;
		for(int j=1;j<=n2;j++)
			if(s1[i-1]==s2[j-1]) ci[j]=ci1[j-1];
			else   ci[j]=1+min(min(ci1[j],ci[j-1]),ci1[j-1])      ;
		for(int j=0;j<=n2;j++)     ci1[j]=ci[j];
	}
	return ci[n2];
}
```

4. **Conectarea cu cost minim a nodurilor la mai multe surse** – varianta a problemei
   http://www.infoarena.ro/problema/retea2 (doar anumite clădiri pot fi conectate)
   Damia s-a hotărât să își deschidă N centrale electrice, numerotate 1,2,…, N. În orașul ei sunt M blocuri care trebuie să primească curent, numerotate N+1, N+2,…, N+M. Un bloc primește curent electric dacă este conectat la alt bloc care primește curent electric sau dacă este conectat la o centrală electrică. Se cunosc coordonatele (euclidiene) celor N centrale și **ale** celor M blocuri. Pentru a conecta două dintre aceste puncte trebuie plătit un cost egal cu distanța euclidiană dintre ele. Dintre toate aceste puncte însă doar unele pot fi conectate în mod direct. Ajutați-o pe Damia să determine costul minim pentru a transmite curent electric către toate blocurile.
   Date de intrare: În fișierul de intrare `retea2.in` se găsesc trei numere naturale N, M si E. Pe fiecare dintre următoarele M+N linii este câte o pereche de numere întregi reprezentând coordonatele în plan ale unei clădiri (centrală sau bloc). Primele N coordonate vor fi coordonatele asociate centralelor, următoarele M fiind coordonatele asociate blocurilor de locuințe. Pe următoarele E linii se găsesc perechi de indici i,j $\in$ {1,…,N+M} reprezentând clădiri care pot fi conectate in mod direct.
   Date de ieşire: În fișierul `retea2.out` pe prima line va fi un număr real, reprezentând costul minim pentru a transmite curent electric blocurilor în modul descris în enunț, iar pe următoarele linii perechile de clădiri care trebuie conectate **O(E log (M+N))**

retea2.in	retea2.out	
2 5 9 0 0 0 4 1 4 1 3 1 1 1 0 3 0 1 2 2 3 3 4 4 5 5 6 1 6 3 5 5 7 6 7	6 1 6 5 6 6 7 2 3 3 4	

5. **Graf dinamic** - Se citesc din fișierul **`grafpond.in`** informații despre un graf ponderat. Folosind unul dintre algoritmii Kruskal sau Prim se determină muchiile unui arbore parțial de cost minim T al grafului **O(mlogn)**.
   Se citește de la tastatură o nouă muchie *e* (dată prin extremitățile sale și costul ei) care trebuie adăugată la G.
   a) Să se afișeze costul maxim al unei muchii din ciclul elementar pe care îl închide *e* în *T* (nu în G) și extremitățile acesteia (**O(n)**)
   b) Folosind eventual a), determinați dacă prin adăugarea muchiei *e* noul graf obținut are un arbore parțial de cost minim cu costul mai mic decât G (**O(1)/O(n)**).
   (O problemă similară, dar mai dificilă: https://www.infoarena.ro/problema/online)

grafpond.in	**iesire**
5 7 1 4 1 1 3 5 1 2 10 2 3 2 4 2 6 4 5 12 5 2 11	Muchiile apcm in G: 1 4 2 3 1 3 2 5 Cost 19 Muchia de cost maxim din ciclul inchis de 3 5 in apcm este 2 5 de cost 11 Dupa adaugarea muchiei apcm are costul 12
De la tastatura 3 5 4	

6. **(SUPLIMENTAR) Second best minimum spanning tree** – Implementați un algoritm eficient pentru determinarea **primilor doi** arbori parțiali cu cele mai mici costuri, pentru un graf conex ponderat, în ipoteza că **muchiile au costuri distincte** (**$O(n^2)$**). Graful se va citi din fișierul `grafpond.in`.

Determinarea următorului arbore cu costul cel mai mic după arborele parțial de cost minim poate fi utilă spre exemplu ca soluție secundară în probleme care se reduc la determinarea unui apcm (conectare, revizie rețea etc)

grafpond.in	**iesire**
6 7 1 2 13 1 3 14 2 3 16 1 4 11 1 5 12 4 5 15 5 6 10	Primul Cost 60 Muchii 1 2 1 3 1 4 1 5 5 6 Al doilea Cost 62 Muchii 1 2 2 3 1 4 1 5 5 6