1. Formularea enuntului problemei

Se da un graf cu n varfuri. Sa se conceapa un algoritm care sa determine un arbore de valoare minima , folosind algoritmul lui Kruskal (varianta 4)

Datele de intrare se dau de la tastatura .

Datele de iesire care prezinta un arbore de valoare minima

2. Dezvoltarea algoritmului

* Subalgoritmul Kruskal4(X,U,V) este:

Preconditie: G =(X,U) este un graf oarecare conex si cu

|U|>=n-1

(am presupus ca G este conex si |U|>=n-1 pentru

ca, daca nu ar fi îndeplinite aceste conditii

atunci nu am avea solutie)

Postconditie: G’=(X,V) este un graf partial al grafului

G=(X,U) si este un arbore de pondere minima

V:=U

Pentru (∀(i,j)∈X) executa

V:=V-{(i,j)}

A:=∅

Marcaj(i,A)

(1) Daca (|A|<>n) atunci

\* Fie v muchia de valoare minima din ω(A),

unde prin ω(A) am notat multimea

{(i,j) | (i,j)∈U si i∈A si j∈X-A)

(2) V:=V∪{v}

SfDaca

SfPentru

SfKruskal4

In punctul (1) conditia |A|<>n este echivalenta cu propozitia “G’=(X,V) nu este conex”. Conexitatea grafului G’ s-ar realiza doar în cazul în care |A|=n (adica toate varfurile grafului s-ar afla in aceeasi componenta conexa).

In puncul (2) se reface conexitatea lui G’ (conexitate care s-a stricat prin eliminarea muchiei (i,j)) deoarece muchia v, care se adauga, este din ω(A). Acest lucru asigura conexitatea grafului G’.

Subalgoritmul Marcaj construieste in multimea A componenta conexa a varfului i. Acest algoritm este bazat pe mecanismul recursivitatii. Prin Γ(i) am notat multimea {j | j∈X si (i,j)∈U}.

* Subalgoritmul Marcaj(i,A) este:

Preconditie: i∈X

Postconditie: A este componenta conexa a lui i

Pentru (∀j∈(Γ(i)-A)) executa (aleg j, un varf neparcurs)

A:=A∪{j}

Marcaj(j,A) (apel recursiv)

SfPentru

SfMarcaj

3. Descrierea algoritmului

Dintre toate muchiile grafului se alege muchia de valoare minimă (maximă). Dacă minimul este multiplu se alege la întâmplare una din muchiile respective. Deoarece acest "la întâmplare" trebuie cumva tradus în limbajul calculatorului, în cazul implementării unui program bazat pe acest algoritm, vom perturba din start valorile muchiilor, la k muchii cu aceiaşi valoare V adunând respectiv valorile ε, 2ε, ... , kε, unde ε este foarte mic (în orice caz, kε mai mic decât diferenţa dintre valoarea acestor arce si valoarea imediat superioară a unui arc), pozitiv. Dintre toate muchiile rămase, se alege cea de valoare minimă (maximă); Dintre toate muchiile rămase, se alege cea de valoare minimă (maximă), astfel încât să nu se formeze cicluri cu cele deja alese;Se reia algoritmul de la pasul 3 până se colectează n-1 muchii.

Deşi s-a demonstrat că algoritmul găseşte întotdeauna arborele optim, el are dezavantajul că este foarte laborios (de fiecare dată trebuie calculat minimul unei mulţimi mari sau foarte mari – există situaţii în practică în care graful are sute de mii de arce) şi, în plus, trebuie aplicat un algoritm special ca să respectăm condiţia de a nu se forma cicluri, la alegerea unui nou arc.

O metodă posibilă este ca, după adăugarea fiecărui arc, să se împartă graful în componente conexe şi să alegem apoi un arc care nu are ambele extremităţile în aceeaşi componentă conexă.

4. Demonstrarea corectitudinii algoritmului

Tot ceea ce am folosit se gaseste in manual si ni s-a predat la curs

5. Codul sursa : in format electronic

6. Date de test

|  |  |
| --- | --- |
| Date de intrare | Date de iesire |
| 4  6  1 2 6  1 3 2  3 4 5  2 4 9  1 4 1  3 2 1 | 1 3  1 4  3 2 |
| 5  8  1 2 16  1 3 11  1 4 12  1 5 28  2 4 9  2 3 36  4 3 10  5 2 17 | 1 3  2 4  4 3 |