BOG‘LANGAN GRAFLARDA MARSHRUTLAR. ENG QISQA MARSHRUTNI ANIQLASH ALGORITMI. UNI VARIANTLAR SONI BO‘YICHA XAJMINI BAXOLASH

Yusupova Janar Kamolovna
Muhammad al-Xorazmiy nomidagi TATU Urganch filiali assistenti

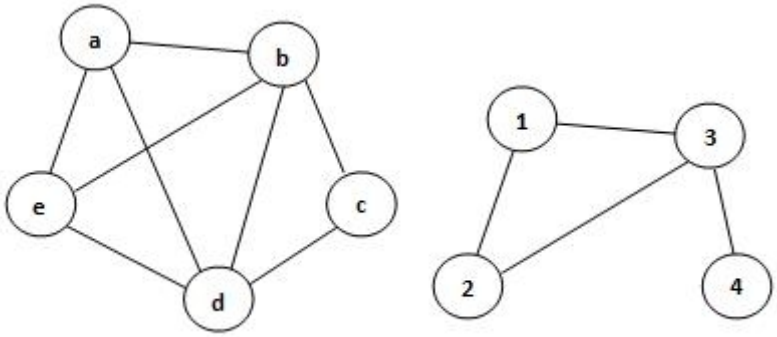
Cho‘ponov Otajon Shuxrat o‘g‘li
Muhammad al-Xorazmiy nomidagi TATU Urganch filiali talabasi

Annotatsiya. Ushbu maqolada bog‘langan graflarda bir uchdan ikkinchi uchga boradigan eng qisqa yo‘lni topish algoritmi berilgan. Ushbu algoritm1959-yilda Niderlandiyalik olim Edsger Deykstra tomonidan ixtiro qilingan, algoritm Grafning bir uchidan qolgan barcha uchlariga boradigan eng qisqa yo‘llarni topish uchun qo‘llaniladi. Algoritm faqat manfiy og‘irlikdagi qirralari bo‘lmagan graflar uchun ishlaydi. Algoritm dasturlashda keng qo‘llaniladi.

Kalit so‘zlar: Graf, algoritm, minimal marshrut, Deykstra, graf uchlari, graf qirralari.

Kirish. Graf - bu abstrakt tushuncha bo‘lib, obyektlar va ular orasidagi bog‘liqliklarni tasvirlashda yoki ifodalashda ishlatiladi. Obyektlarni ko‘p hollarda nuqtalar bilan belgilab olinadi va ularga nomer beriladi. Bu grafning uchlari deb ham ataladi. Grafning uchlarini sonlar to‘plami sifatida qaraymiz va uni V harfi bilan belgilaymiz. Grafning uchlarini 1 dan N gacha nomerlash mumkin (yoki 0 dan n-1 gacha). Graf uchlari orasidagi bog‘liqliklarni sonlar jufti bilan belgilaymiz (u_i , v_i) va bu grafning u_i hamda v_i nomerli uchlari o‘zaro bog‘liqligini bildiradi. Bunday juftliklarni grafning qirralari deyiladi va ular E harfi bilan belgilanadi. E to‘plam elementlari juftlik sonlardan iborat.

Ixtiyoriy grafni uning uchlarini bildiruvchi to‘plam V va qirralarini bildiruvchi to‘plam E bilan berish mumkin. Grafni G harfi belgilasak, uni quyidagicha ifodalash mumkin:G(V, E). Bundan tashqari graflarni oddiygina qilib rasmli ko‘rinishda tasvirlash mumkin. Bunda uchlari uchun nuqtalar qo‘yib, keraklilarini chiziqlar bilan tutashtiramiz. Qizig‘i shundaki, bu yoqda nuqtalarning o‘rni ahamiyatga ega emas, faqat bog‘liqliklar ko‘rinsa bo‘ldi. Graflarni bu usulda tasvirlash ularga oid misollarni qo‘lda yechganda, yoki tahlil qilganda juda qo‘l keladi.



1-rasm. Graflarni tasvirlanishi

Graflarga juda ko‘plab misollar keltirish mumkin:

1. Ixtiyoriy tarmoq - graf. Bunda tarmoq elementlari va ular orasidagi bog‘lanishlar bor.
2. Shaharlar va ularni tutashtiruvchi yo‘llar
3. Kishilar va ular orasidagi bog‘liqliklar. Ota-bola-nabira va h.k

Maslaning qo‘yilishi. n ta uchli va m ta qirraga ega bo‘lgan orietrlangan yoki orientrlanmagan graf berilgan. Qandaydir boshlang‘ich uch s ko‘rsatilgan. s uchdan boshqalarigacha bo‘lgan eng qisqa masofalarni topish kerak va yo‘lning o‘zinin ham chiqarish talab etiladi. Bu masala “yagona manbaali eng qisqa yo‘llarni izlash masalasi” (single-source shortest paths problem) deyiladi[1].

Algoritm. Bu yerda Deykstra (Dijkstra) tamonidan 1959 yilda ishlab chiqilgan algoritm tavsiflangan.

Harbir *v* uch uchun ungacha bo‘lgan eng qisqa masofani belgilovchi *d[ ]* massiv kiritamiz. Dastlab d[s] = 0 boshqalari uchun esa uni qandaydir bir katta son bilan to‘ldirib qo‘yamiz. (Masalada bu uzunlik mavjud eng katta yo‘l uzunligidan ham katta bo‘ladigan shartni qanoatlantirishi kerak):

$$d[v] = \infty, v \neq s$$

Undan tashqari harbir v uch uchun uning ko‘rilgani yoki ko‘rilmaganini bildiruvchi matqiqiy(boolean) u[] massiv kiritamiz.Dastlab barcha uchlar hali ko‘rilmagan ya’ni:

$$u[v] = false$$

Deyksta algoritmi *n* ta iteratsiyadan iborat. Navbatdagi iteratsiyada *d[v]* kattalik eng kichik bo‘lgan hali ko‘rilmagan v uch tanlanadi ya’ni:

$$d[v] = \min_{p:u[p]=false} d[p]$$

(Birinchi iteratsiyada boshlang‘ich *s* uch ko‘riladi).

Shunday usul bilan tanlangan *v* uch ko‘rilgan uchlar qaotiga kiritiladi. Undan so‘n esa joriy iteratsiyada *v* uchdan relaksatsiya tekshiriladi. Ya’ni barcha (v, to) qirralar qarab chiqiladi va to uch uchun $d[to]$ qiymat yaxshilashtirishga uriniladi.

Ko‘rilayotgan qirraning uzunligi *len* ga teng bo‘lsin, u holda relaksatsiyani tekshirish quyidagicha bo‘ladi:

$$d[to] = \min(d[to], d[v] + len)$$

Joriy iteratsiya yakunlangacha algoritm keying iteratsiyaga o‘tadi(yana *d* masofa eng kichik hali ko‘rilmagan uch tanlanadi va undan relaksatsiya tekshiriladi va h.). Bunda *n* iteratsiyadan keyin ohir-oqibatda grafning barcha uchi ko‘rilgan bo‘lib qoladi va algoritm o‘z ishini yakunlaydi. *d[v]* qiymat *s* uchdan *v* uchgacha bo‘lgan eng qisqa masofa bo‘ladi.

Yana shuni eslatib o‘tish kerakki *s* uchdan mavjud yo‘llar orqali borib bo‘lmaydigan uchlar uchun *d[v]* qiymat cheksiz kattaligicha qoladi. Ma’lumki algoritmning ohirgi birnechta iteratsiyasida bu uchlar tanlanadi lekin hech qanday yaxshilanish sodir bo‘lmaydi. Chinki cheksiz kata masofaga qirra uzunligini qo‘shganda cheksiz kata sondan kichik bo‘lmaydi. Shuning uchun bu holda algoritmni darhol to‘xtatish mumkin.

Yo‘lni tiklash. Bizdan faqat eng qisqa yo‘lning uzunligi so‘ralmasdan, balki bu yo‘lnig o‘zi ham ya’ni unga borishdagi qaysi uchlardan o‘tish ketma-ketligi so‘ralishi mumkin. *s* uchdan boshqa uchlargacha yo‘lni ketma-ket qanday tiklash mumkinligini ko‘rsatamiz. Buning uchun **predka massiv** deb nomlanadigan *p[]* massiv kiritamiz. Bu massiv harbir $v \neq s$ uch uchun $p[v]$ kattalik v uchga undan oldin qaysi uch orqali kelganligini bildiruvchi ma’lumot saqlashimiz. Bunda biz quyidagicha faktdan foydalanamiz. Agar biz qandaydir v uchgacha bo‘lgan eng qisqa yo‘lni olib undan ohirgi uchni o‘chirsak u holda bu uchdan oldingi $p[v]$ uchda yakunlanadigan qandaydir yo‘l hosil bo‘ladi va bu yo‘l $p[v]$ uchun eng qisqa bo‘ladi. Shunday qilib, agar biz predka massiv bo‘yicha yurib borsak ohir oqibatda boshlang‘ich uchga yetib boramiz. Bu esa butun bosib o‘tilgan yo‘lni tiklab olish imkoniyatini beradi. Lekin bu ketma-ketlikni teskari tartibda olishimiz kerak[2].

Shunday qilib v uchgacha bo‘lga eng qisqa masofa P quyidagiga teng:

$$P = (s, \dots, p\left[p[p[v]]\right], p[p[v]], p[v], v)$$

Bu predka massivni qanday qilib hosil qilishni tushinish qoldi. Lekin bu juda oddiy amalga oshiriladi: har bir relaksatsiyada v uchdan to uchgacha bo‘lgan masofa yaxshilansa u holda to uchga v uch orqali kelgan bo‘ladi. Demak to uchning predkasini v uch qilib belgilab qo‘yish mumkin.:

$$p[to] = v$$

Isbotlash. Deykstra algoritmining to‘g‘riligi quyidagi tasdiq orqali tushintiriladi. Agar qandaydir v uch ko‘rilgan uchlar qatoriga kiritilsa u holda ungacha bo‘lgan eng qisqa masofa $d[v]$ allaqachon hisoblangan bo‘ladi va u boshqa o‘zgarmaydi.

Isbotlash induksiya bo‘yicha amalga oshiriladi. Birinchi iteratsiya uchun uning to‘g‘riligi aniq— s uch uchun $d[s] = 0$, uning uchun eng qisqa masofa hisoblanadi. Endi bu tasdiq oldin ko‘rilgan barcha iteratsiyar uchun to‘g‘ri deb tasavvur qilaylik, ya’ni ko‘rilgan uchlar uchun; joriy iteratsiyadan so‘ng ham u buzilmasligini isbotlaymiz. Joriy iteratsiyada tanlab olingan ya’ni ko‘rilgan uchlar qatoriga qo‘shilishga tayyorlanayotgan uch v bo‘lsin. $d[v]$ masofa uning uchun haqiqatdan ham eng qisqa masofa bo‘lishini isbot qilamiz. (bu uzunlikni $l[v]$ orqali belgilaymiz). v uchgacha bo‘lgan eng qisqa masofa P ni qarab chiqamiz. Bu yo‘lni ikki qismga ajratish mumkin: P_1, faqat ko‘rilgan uchlardan tashkil topgan uchlardan tashkil topgan (minimum boshlang‘ich s uch bu yo‘lda bo‘ladi), va qolgan qismi P_2 yo‘l(u ko‘rilgan uchlardan iborat bo‘lishi mumkin lekin albatta ko‘rilmagan uchdan boshlanishi kerak). P_2 yo‘lning birinchi uchini P bilan belgilaymiz, P_1 ning ohirgi uchini esa q orqali belgilaymiz.

Dastlab bizning tasdiqni P uchun ko‘rsatamiz,ya’ni:

$$d[p] = l[p]$$

tenglikni tasdiqlaymiz. Lekin bu amaliy ma’lum: undan oldingi iteratsiyalardan birida biz q uchni tanladik va undan relaksatsiyani bajardik. p uchgacha bo‘lgan eng qisqa yo‘l q uchgachabo‘lgan eng qisqa yo‘l + (p, q) qirra uzunligiga teng, shuning uchun q dan relaksatsiya bajarilganda $d[p]$ kattalik haqiqatdan kerakli qiymat o‘rnatiladi.

Qirralarning nomanfiyligidan kelib chiqadigan bo‘lsak, $l[p]$ eng qisqa yo‘l (u faqat isbodlangandan so‘ng $d[p]$ ga teng bo‘ladi) v gacha bo‘lgan eng qisqa yo‘l $l[v]$ dan oshmaydi. $l[v] \leq d[v]$ ekanligini hisobga olib(Deykstra algoritmi eng qisqa yo‘ldan ham qisqasini topa olmaydi, bu umuman mumkin emas), natijada quyidagicha munosabadni olamiz.

$$d[p] = l[p] \leq l[v] \leq d[v]$$

Boshqa tamondan ham p, ham v — ko‘rilmagan uchlar, shuning uchun joriy iteratsiyada aynan v uch tanlangan, p uch emas, shunday qilib quyidagicha tengsizlikni olamiz:

$$d[p] \geq d[v]$$

Bu ikkita tengsizlikdan natijaviy $d[p] = d[v]$ tenglikni hosil qilamiz, u holda bungacha topilgan munosabatlardan shuni hosil qilamiz

$$d[v] = l[v]$$

isbotnashi talab qilingan.

Amalga oshirish.

Shunday qilib Deykstra algoritmi n iteratsiyadan iborat. Harbir iteratsiyada $d[v]$ qiymat eng kichik bo‘lgan hali ko‘rilmagan uch tanlanadi. Bu uch ko‘rilgan uchlar

ro‘yxatiga kiritladi va bu uchdan chiquvchi harbir uch bo‘yicha relaksatsiya tekshiriladi va har bir qirra bo‘yicha $d[]$ massiv qiymatini yaxshilashga harakat qilinadi.

Algoritmning ishlash vaqti quyidagicha hisoblaymiz:

• n marta hali ko‘rilmagan uchlar orasida eng kichik $d[v]$ li uchni topamiz, ya’ni n uch ichidan.

• m marta relaksatsiyaga urinish amalga oshiriladi

Bu amallarni oddiy amalga oshirishda, uchni izlash uchun $O(n)$ operatsiya, bitta relaksatsiya uchun esa— $O(1)$ operatsiya ketadi. Algoritmning natijaviy asimptotikasi:

$$O(n^2 + m)$$

Xulosa

Ushbu Dekstra algoritmini 1959-yil olim Deykstra tomonidan taklif qilingan. Algoritm ishlash prinsipi juda oddiy bitta d[] massiv olamiz va unda grafning har bir V uchidan bizga berilgan S uchgacha bo‘lgan eng qisqa qiymatni saqlaymiz. Buning uchun oldin d[s]=0 tenglaymiz va d[] ning barcha qiymatlarini musbat cheksizlar bilan to‘ldiramiz. So‘ngra har bir uchni tekshirilganini bilish uchun bitta boolean tipidagi used[] massiv ham ochamiz. Keyin n itaratsiya orqali d[i] ning hali tekshirilmagan va eng kichik d[v] uchini topamiz. Tanlangan d[v] uch uchun u bilan bog‘langan har bir uch uchun relaksatsiyasini ko‘rib chiqamiz. Relaksatsiya quyidagi ko‘rinishda bo‘ladi:

$$\mathrm{d[i]} = \min(\mathrm{d[i]}, \mathrm{d[v]} + \mathrm{len})$$

Shu tariqa bizda d[] massivda S uchdan har bir V uchgacha bo‘lgan eng qisqa masofa joylanadi.

FOYDALANILGAN ADABIYOTLAR

1. H. Bast, S. Funke, P. Sanders. and D. Schultes. Fast routing in road networks with transit nodes. Science 316 (2007), no. 5824, 566.

2. K. Madduri, D.A. Bader, J. W. Berry, and J. R. Crobak, An experimental study of a parallel shortest path algorithm for solving large-scale graph instances, 2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX), New Orleans, Louisiana, January 2007.