Станка Матковић

АЛГОРИТМИ НАЛАЖЕЊА НАЈКРАЋИХ ПУТЕВА У ГРАФОВИМА И ЊИХОВА ПРИМЕНА

У овом чланку су описани најпознатији алгоритми за налажење најкраћих путева у графу: Дајкстрин и Флојд-Воршелов алгоритам. На почетку су дефинисани основни појмови који ће се користити у даљем тексту. Затим следи опис алгоритама и њихове примене.

1. Дефиниције основних појмова

Граф у ознаци G је уређен пар (V, E), где је V скуп чворова, а $E \subset V \times V$ скуп грана (ивица) графа. Ако су x, y чворови графа и $(x, y) \in E$, онда кажемо да су x и y *суседни* чворови у графу G. Граф је *неоријентисан* ако $(\forall x, y \in V)\,(x, y) \in E \implies (y, x) \in E$, у супротном је граф *оријентисан*. У оријентисаном графу ивице можемо посматрати као уређене парове (x, y), $x, y \in V$, а у неоријентисаном ивице су двочлани скупови $\{x, y\}$, $x, y \in V$.

Пут у графу $G = (V, E)$ је низ $v_1, v_2, \ldots, v_k$ ($v_i \in V$, $1 \leqslant i \leqslant k$) при чему $(v_i, v_{i+1}) \in E$ за $1 \leqslant i < k$, тј. свака два узастопна чвора у низу $v_1, v_2, \ldots, v_k$ су суседна. За пут $v_1, v_2, \ldots, v_k$ кажемо да повезује чворове v_1 и v_k. Граф је *повезан* ако за свака два чвора тог графа постоји пут који их повезује.

Граф $G = (V, E)$ најчешће представљамо помоћу матрице $(A_{|V|\times|V|})$ повезаности (суседства), при чему су чланови матрице овако дефинисани: $A[u][v] = 1$ ако је $(u, v) \in E$, $A[u][v] = 0$ ако $(u, v) \notin E$ (слика 1). Како матрица $A_{|V|\times|V|}$ има $|V|^2$ чланова, за представљање графа помоћу матрице потребан је меморијски простор $O(|V|^2)$.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Сл. 1. Пример оријентисаног графа представљеног помоћу матрице повезаности

У пракси често није довољно посматрати само повезаност чворова у графу, већ су од интереса неке карактеристике грана (нпр. дужина гране, капацитет гране). Зато се графу $G = (V, E)$ придружује функција $f\colon E \to \mathbf{R}$ којом се свакој грани графа придружује неки реалан број.

У даљем тексту посматрамо граф $G = (V, E)$ и функцију $f\colon E \to \mathbf{R}^+$ која свакој грани графа G придружује дужину гране. Претпостављамо да су чворови графа нумерисани од 1 до n, тј. $V = \{1, 2, \ldots, n\}$. Граф G представљамо помоћу матрице повезаности $(A_{|V|\times|V|})$, $A[u][v] = f((u, v))$ ако је $(u, v) \in E$, $A[u][v] =$ NE_POSTOJI ако је $(u, v) \notin E$, где је NE_POSTOJI симболичка константа, нпр. неки број који не може бити вредност функције f. Треба истаћи да овде посматрамо само позитивне дужине грана, јер ако неке гране имају „негативне дужине“ проблеми које ћемо разматрати не решавају се на исти начин (а понекад и немају решење).

2. Одређивање дужина најкраћих путева од једног чвора до свих осталих (Дајкстра)

Проблем. *Ако је дат оријентисан граф $G = (V, E)$, функција $f\colon E \to \mathbf{R}^+$ која свакој грани придружује дужину гране, и чвор $s \in V$, наћи дужине најкраћих путева од чвора s до свих осталих чворова из G, и одредити те најкраће путеве.*

Најпознатији алгоритам за решавање овог проблема развио је Дајкстра, и у њему се користи итеративни поступак.

При решавању овог проблема скуп чворова V делимо на два дисјунктна подскупа. „Маркирани“ чворови, у ознаци M, су чворови за које смо израчунали дужину најкраћег пута од стартног чвора s и „немаркирани“ су чворови за које дужина најкраћег пута није израчуната. На почетку једини „маркиран“ чвор је s, значи $M = \{s\}$.

Поред тога за сваки „немаркиран“ чвор v пратимо дужину најкраћег пута од s до v, при чему су сви чворови тог пута осим v „маркирани“, ако такав пут постоји. Дужине тих путева памтимо низом d реалних бројева. На почетку $d[v] = A[s][v]$ за све $v \in V$.

Да бисмо могли одредити најкраћи пут од чвора s до свих осталих чворова, упоредо са низом d, користимо низ p, где је $p[v]$, за све $v \in V$, редни број суседног чвора за чвор v на путу дужине $d[v]$, који повезује чворове s и v. На почетку $p[v] = s$ за све чворове v који су повезани са s.

У свакој итерацији алгоритма скупу „маркираних“ чворова додајемо „немаркиран“ чвор ω за који је $d[\omega]$ најмање, тј. „немаркирани“ чвор који је „најближи“ стартном чвору s. После додавања новог чвора скупу M низ (d) дужина најкраћих путева „немаркираних“ чворова се мора кориговати. За сваки чвор $v \notin M$, $d[v]$ је дужина најкраћег пута од s до v преко чворова из скупа M; како је скуп M сада проширен са ω треба проверити да ли је пут од s до v преко ω краћи. Зато упоређујемо величине $d[v]$, $d[\omega] + A[\omega][v]$ за чворове који су суседни чвору ω и мању од тих величина придружујемо $d[v]$. При томе ако мењамо $d[v]$, морамо променити и $p[v]$, тј. $p[v]$ добија вредност ω.

Ако је граф повезан, алгоритам се завршава када све чворове „маркирамо“ и тада је за сваки чвор $v \in V$, $d[v]$ дужина најкраћег пута од s до v. Ако је граф неповезан, алгоритам се завршава кад је $d[v]$ једнако NE_POSTOJI за све „немаркиране“ чворове.

Опис алгоритма.

Улаз: Матрица растојања A и стартни чвор s

Излаз: Низ d ($d[v]$ дужина најкраћег пута од s до v), низ p помоћу којег се реконструише пут од s до свих осталих и логичка променљива ind (`true` ако је граф повезан, `false` у супротном)

```
begin
for sve cvorove ω do
  ω_mark=false;
  d[ω]=A[s][ω];
  if d[ω] <> NE_POSTOJI
    then p[ω]=s;
s_mark=true;
while postoji "nemarkiran" cvor do
  Naci "nemarkiran" cvor ω kome je d[ω] najmanje.
  if d[ω]=NE_POSTOJI then
    ind = false;
    exit;
  Markiraj cvor ω (ω_mark=true).
  for sve grane (ω,v) gde je v "nemarkiran" cvor do
    if d[v]<d[ω] + A[ω][v] or d[v]=NE_POSTOJI
      then d[v]=d[ω] + A[ω][v];
        p[v]=ω;
ind=true;
end.
```

Ево како изгледа један начин имплементације описаног алгоритма у Pascal-у.

```
type niz=array[1..100]of real;
  mat=array[1..100] of niz;
  put=array[1..100]of integer;
procedure dajkstra(n,s:integer; var A:mat; var d:niz; var p:put; var ind:boolean);
{n-broj cvorova, s-startni cvor, A-graf, d-niz duzina najkracih puteva od s,
p-niz pomocu kojeg rekonstruisemo put od cvora s do svih ostalih
ind-logicka promenljiva koja ukazuje da li je graf povezan}
var i,j,w:integer;
  mark:array[1..100]of boolean;
begin
  for i:=1 to n do
    begin
      d[i]:=A[s][i];
      mark[i]:=false;
      if d[i]<>NE_POSTOJI
        then p[i]:=s;
  end;
mark[s]:=true;
d[s]:=0;
for i:=1 to n-1 do
    begin
      {trazimo nemarkiran cvor za koji postoji put od s do njega}
      j:=1;
      while (j<=n) and (mark[j] or (d[j]=NE_POSTOJI)) do
        j:=j+1;
      {ako takav cvor ne postoji onda graf nije povezan}
      if j>n then
        begin ind:=false;exit;end;
      {trazimo nemarkiran cvor w za koji je d[w] najmanje}
      w:=j; j:=j+1;
```

```
      while j<=n do
        begin
          if not mark[j] and (d[j]<>NE_POSTOJI) and (d[j]<d[w])
            then w:=j;
          j:=j+1;
        end;
      {markiramo nadjeni cvor}
      mark[w]:=true;
      {korigujemo ako je potrebno, pomocu w nizove d i p za cvorove
      koji nisu markirani,a povezani su sa w}
      for j:=1 to n do
        if not mark[j] and (A[w][j]<>NE_POSTOJI)
          then if (d[j]=NE_POSTOJI) or (d[w]+A[w][j]<d[j])
            then begin
              d[j]:=d[w]+A[w][j];
              p[j]:=w;
              end;
    end;
  ind:=true;
end;
```

Временска и просторна сложеност овог алгоритма је $O(n^2)$.

Ако је дат граф G приказан на слици 2. Налажење дужина најкраћих путева од чвора s, нпр. чвора (1), до преосталих чворова графа G коришћењем Дајкстриног алгоритма приказано је на слици 3. Чворови су представљени кружићима, број поред кружића означава редни број чвора, а број у кружићу дужину најкраћег пута од чвора s до чвора представљеног тим кружићем, слово N значи да не постоји пут од s до тог чвора преко маркираних чворова. Маркирани чворови су затамњени. Бројеви на гранама су дужине грана. Подебљане су оне гране које су саставни делови најкраћих путева.

Сл. 2

За исписивање најкраћег пута од чвора s до чвора k користимо процедуру *pisi*:

```
procedure pisi(k,s:integer; var p:put);
  if k<>s
    then pisi(p[k],s,p);
  write(k:3);
end;
```

Сл. 3

Пример 1. На забави се налази n људи нумерисаних бројевима од 1 до n, матрицом $A_{n\times n}$ дато је ко се с ким познаје. Особа s зна вест коју треба да саопшти осталима. Особа i може препричати вест само особама које познаје. Свако препричавање траје 1 минут. Препричавања се могу одвијати паралелно и свака особа може окупити своје познанике којима ће саопштити вест. Свака особа чим чује вест, одмах је препричава (не губи се време) и није могуће да једна особа прима вест истовремено од два познаника. Написати процедуру којом се одређује минимум времена у минутима да би сви чули вест, и ако је то могуће за сваку особу исписује редне бројеве особа преко којих је до ње стигла вест.

Претпоставимо да је $A[i][j] = 1$ ако особа i познаје особу j, иначе је $A[i][j] = 0$. Матрица A представља матрицу повезаности у графу $G = (V, E)$ где је $V = \{1, 2, \ldots, n\}$ а дужине грана су 1. Ако коришћењем процедуре *dajkstra* одредимо низ d дужина најкраћих путева од особе s која зна вест до свих осталих особа у

графу G, минимум времена у минутима који је потребан да би сви чули вест је максимални члан низа d.

```
const NE_POSTOJI=0;
procedure prepricavanje(n,s:integer;A:mat;var ind:boolean;var x:integer);
{n-broj ljudi, s-osoba koja zna vest, A-martica poznanstva,
ind-logicka promenljiva koja ukazuje da li svi mogu cuti vest,
x-minimum vremena u minutima koji je potreban da bi svi culi vest}
var i:integer;
  p:put;
  d:niz;
begin
  dajkstra(n,s,A,d,p,ind);
  if ind then
    begin
      x:=d[1];
      for i:=2 to n do
        if (d[i]>x)
          then x:=d[i];
      writeln(' Minimum vremena u minutima da bi svi culi vest je ',x);
      for i:=1 to n do
        if i<>s
          then begin
            write('Osoba ', i, ' :  ');
            pisi(i,s,p);
            writeln;
          end
    end
end;
```

Задатак 1. Дате су информације о директним путевима између n градова помоћу матрице веза $A_{n\times n}$ ($A[i][j]$ је дужина пута од града i до града j ако постоји пут, у супротном је $A[i][j]$ је 0). Написати програм који ће за пар целих бројева (k, m) $(1 \leqslant k, m \leqslant n)$ штампати редне бројеве градова кроз које треба проћи, приликом путовања од града k до града m, при чему је укупна дужина пута најмања могућа.

3. Одређивање дужина најкраћих путева између свих парова чворова

Проблем. За дати оријентисан граф $G = (V, E)$ са ненегативним гранама одредити дужине најкраћих путева између свих парова чворова и одредити те путеве.

Овај проблем решава Флојд-Воршелов (Floyd-Warshell) алгоритам, чији опис следи. Основни принцип који се користи у решењу је: ако је k чвор најкраћег пута од чвора i до чвора j, онда је део пута од i до k , и од k до j такође најкраћи. Кроз n итерација формирамо матрицу D која садржи дужине најкраћих путева између свих парова чворова. На почетку D иницијализујемо на мартрицу A. Означимо матрицу D после k-те итерације са $D^{(k)}$. После k-те итерације D садржи дужине најкраћих путева у којима су међучворови из скупа $\{1, 2, \ldots, k\}$ (тј. $D[i][j]^{(k)}$ представља дужину најкраћег пута од чвора i до чвора j при чему су редни бројеви свих чворова тог пута осим крајњих мањи

или једнаки k). Према томе после n итерација матрица D садржи резултат. У k-тој итерацији алгоритма проверавамо за све парове чворова (i, j) да ли постоји пут преко чвора k који је краћи од пута који пролази кроз чворове из скупа $\{1, 2, \ldots, k-1\}$. Потребно проверавање можемо записати на следећи начин:

$$D[i][j]^{(k)} = \min(D[i][j]^{(k-1)}, \quad D[i][k]^{(k-1)} + D[k][j]^{(k-1)})$$

за $1 \leqslant i, j, k \leqslant n$.

У пракси често није довољно знати само дужине најкраћих путева, већ је потребно знати и најкраћи пут. У том случају користимо матрицу P коју иницијализујемо на 0. У претходно описаном алгоритму у свакој итерацији кад мењамо $D[i][j]$ мењамо и матрицу P, тако што $P[i][j]$ добија вредност k. Тако у $P[i][j]$ памтимо број последње итерције у којој је $D[i][j]$ мењано, тј. чвор са највећим редним бројем на путу од i до j. Кад се алгоритам заврши, да бисмо реконструисали пут од i до j посматрамо $P[i][j]$. Ако је $P[i][j] = 0$, најкраћи пут је грана (i, j), ако она постоји; иначе ако је $P[i][j] = k$, најкраћи пут од i до j пролази кроз чвор k, зато рекурзивно посматрамо $P[i][k]$ и $P[k][j]$ да бисмо нашли остале међучворове најкраћег пута.

Имплементација описаног алгоритма на Pascal-у би изгледала овако:

```
type niz=array[1..100]of real;
  mat=array[1..100] of niz;
  matput=array[1..100] of array [1..100] of integer;
procedure Flojd(n:integer;var A,D:mat; var P:matput);
{n-broj cvorova; A-graf; D-matrica najkracih rastojanja, P-matrica puteva}
var i,j,k:integer;
begin
  {inicijalizacija matrica D i P}
  for i:=1 to n do
    for j:=1 to n do
      begin
        D[i][j]:=A[i][j];
        P[i][j]:=0;
      end;
for k:=1 to n do
  for i:=1 to n do
    for j:=1 to n do
      if i<>j then
        if(D[i][k]<>NE_POSTOJI) and (D[k][j]<>NE_POSTOJI)
          then if (D[i][j]=NE_POSTOJI) or (D[i][k]+D[k][j]<D[i][j])
            then
              begin
                D[i][j]:=D[i][k]+D[k][j];
                P[i][j]:=k;
              end;
end;
```

Временска сложеност датог алгоритма је $O(n^3)$ уз просторну сложеност $O(n^2)$. Ако желимо да испишемо међучворове најкраћег пута између чворова i и j користимо процедуру *pisi*:

```
procedure pisi(i,j:integer; var P:matput);
var k:integer;
begin
```

```
  if P[i][j]<>0
  then begin
    k:=P[i][j];
    pisi(i,k,P);
    write(k:3);
    pisi(k,j,P);
  end
end;
```

Пример 2. Ако је над скупом $\{1, 2, \ldots, n\}$ задата бинарна релација α матрицом $A_{n\times n}$, $A[i][j] =$ `true` ако су i и j у релацији, иначе $A[i][j] =$ `false`. Одредити релацију α_1 која представља минимално транзитивно затворење релације α.

Релација α_1 је минимално транзитивно затворење релације α ако је релација α_1 транзитивна и $\alpha \subset \alpha_1$ и ако за сваку транзитивну релацију β где је $\alpha \subset \beta$ важи $\alpha_1 \subset \beta$.

Бинарна релација α индукује граф $G = (V, E)$, где је $V = \{1, 2, \ldots, n\}$, а између чворова u и v постоји ивица само ако елементи u и v јесу у релацији α, тј. $(u, v) \in E$ само ако је $A[u][v] =$ `true`. Елементи u и v биће у релацији α_1 акко постоји пут од чвора u до чвора v у графу G. Према томе за минимално транзитивно затворење релације можемо искористити алгоритам за налажење најкраћих путева између свих парова чворова.

```
type niz=array [1..100] of boolean;
  mat=array [1..100] of niz;
procedure tranz_zatv( n:integer; var A,A1:mat)
var i,j,k:integer;
begin
  for i:=1 to n do
    for j:=1 to n do
      A1[i][j]:=A[i][j];
  for k:=1 to n do
    for i:=1 to n do
      for j:=1 to n do
        A1[i][j]:=A1[i][j] or (A1[i][k] and A1[k][j]);
end;
```

Задатак 2 Дата је матрица $A_{n\times n}$ при чему је $A[i][j] = 1$ ако постоји директан пут од града i до града j, у супротном је $A[i][j] = 0$. Написати програм који ће за сваки пар целих бројева (i, j) $(1 \leqslant i, j \leqslant n)$ одредити најмањи број градова кроз које треба проћи, приликом путовања од града i до града j, ако постоји пут између i и j.

ЛИТЕРАТУРА

[1.] Brassard G., Brately B., *Algorithmics, Theory and Practice*, Prentice-Hall International 1988.

[2.] Cormen T.H., Leiserson C.E., Rivest R.L., *Introduction to Algorithms*, MIT Press-McGraw Hill, 1991.

[3.] Manber U., *Introduction to Algorithms, A Creative Approach*, Addison-Wesley Publishing Company, 1989.

[4.] Urošević D., *Algoritmi u programskom jeziku C*, Mikro knjiga 1996.