Олександр Рудик

Пошук максимального потоку

(Інформатика та інформаційні технології в навчальних закладах,
2008, № 6, с. 94–98)

Означення 1. *Запровадимо такі поняття й позначення.*

1. *Для довільної вершини v:*
   - in(v) — *множина дуг з кінцевою вершиною v;*
   - out(v) — *множина дуг з початковою вершиною v.*
2. *Мережею називають орієнтований граф* (G, V, E) *разом з ваговою функцією c: E → N (з множини ребер E у множину натуральних чисел N) та вершинами a, z, що мають відповідно нульові степені входу й виходу:*
   - *жодна дуга не закінчується у вершині a;*
   - *жодна дуга не починається у вершині z.*

   $|\,\mathrm{in}\,(a)\,| = 0 = |\,\mathrm{out}\,(z)\,|$ — стислий запис останніх двох висловлювань.
3. *Потоком у мережі* (G, V, E) *називають функцію f: E →* N_0 *(з множини ребер E у множину невід'ємних цілих чисел* N_0*), при якій:*
   - *для довільної дуги e справджуються нерівності:* $0 \le f\,(e) \le c(e)$;
   - *для довільної вершини, відмінної від a та z, маємо:*

$$\sum_{e \in \mathrm{out}\,(v)} f\,(e) = \sum_{e \in \mathrm{in}\,(v)} f\,(e).$$

Мережа є моделлю водогону, у якій:
$c(e)$ — максимальна швидкість транспортування у напрямку дуги e;
$f\,(e)$ — (реальна) швидкість транспортування у напрямку дуги e.

Рівність сум в означенні потоку описує відсутність накопичення рідини на проміжних станціях, поданих вершинами графа.

Теорема 1. *Для довільного потоку f маємо:*

$$\sum_{e \in \mathrm{out}\,(a)} f\,(e) = \sum_{e \in \mathrm{in}} f\,(e),$$

(z)

тобто сумарні потоки через джерело a і стік z збігаються.

Доведення. Нехай S — довільна підмножина V, що містить a, але не містить z, $T = V \setminus S$. Додавши почастинно рівності з означення потоку за всіма вершинами S, відмінними від a, отримаємо еквівалентні рівності:

$$\sum_{v \in S \setminus \{a\}} \sum_{e \in \mathrm{out}(v)} f(e) = \sum_{v \in S \setminus \{a\}} \sum_{e \in \mathrm{in}(v)} f(e);$$

$$\sum_{v \in S \setminus \{a\}} \sum_{e \in \mathrm{out}(v)} f(e) - \sum_{v \in S \setminus \{a\}} \sum_{e \in \mathrm{in}(v)} f(e) = 0;$$

$$\sum_{v \in S} \sum_{e \in \mathrm{out}(v)} f(e) - \sum_{v \in S} \sum_{e \in \mathrm{in}(v)} f(e) = \sum_{e \in \mathrm{out}(a)} f(e).$$

Позначимо через $(S; T)$ множину дуг, спрямованих з S у T, через $(T; S)$ — множину дуг, спрямованих з T в S. У лівій частині останньої рівності, якщо обидві вершини дуги e належать до S, то потік по e буде враховано в обох сумах, а відповідні доданки взаємно знищаться:

- від зменшуваного залишаться потоки по дугам з $(S; T)$;
- від від'ємника залишаться потоки по дугам з $(T; S)$.

Маємо:

$$\sum_{e \in (S; T)} f(e) - \sum_{e \in (T; S)} f(e) = \sum_{e \in \mathrm{out}(a)} f(e).$$

Отже, для довільної множини вершин S, що містить a і не містить z, різниця потоків, які виходять з S і входять в S дорівнює потоку, що витікає з a.

Виберемо: $S = V \setminus \{z\}$, $T = \{z\}$, при яких:

- $(T; S) = \emptyset$ — порожня множина, а відповідна сума у лівій частині останньої рівності дорівнює 0;
- $(S; T) = \mathrm{in}\,(z)$ — множина дуг, спрямованих у z.

Врахувавши все це, маємо:

$$\sum_{e \in \mathrm{in}\,(z)} f(e) = \sum_{e \in \mathrm{out}\,(a)} f(e),$$

що й потрібно було довести.

Означення 2. *Запровадимо такі поняття й позначення:*

1. *Величиною потоку f називають величину:*

$$\sum_{e \in \mathrm{out}\,(a)} f(e) = \sum_{e \in \mathrm{in}\,(z)} f(e),$$

яку позначають через $\mathrm{val}\,(f)$.

2. *Нехай S — довільна підмножина V, $T = V \setminus S$. Перерізом $(S; T)$ називають множину дуг, спрямованих з S у T. Якщо S містить джерело a і T містить стік z, то такий переріз називають $a - z$ перерізом.*

3. *Величину*

$$C(S, T) = \sum c(e)$$

$$e \in (S; T)$$

називають пропускною спроможністю перерізу.

4. *Потік* $f_{\max}$ *називають максимальним, якщо його величина не менша від величини будь-якого можливого потоку* f *у мережі:* $\mathrm{val}\,(f) \le \mathrm{val}\,(f_{\max})$.

5. $a - z$ *переріз* $(S; T)$ *називають мінімальним перерізом, якщо* $C(S, T)$ *не перевищує пропускну спроможність довільного* $a - z$ *перерізу.*

Теорема 2. *Нехай* f *— довільний потік у мережі* S *— довільна підмножина* V*, що містить джерело* a *і не містить стік* z*,* $T = V \setminus S$*. Тоді* $\mathrm{val}\,(f) \le C(S, T)$.

Доведення. Як встановлено вище, величина потоку $\mathrm{val}\,(f)$ дорівнює такій різниці:

$$\sum_{e \in (S; T)} f(e) - \sum_{e \in (T; S)} f(e) \le \sum_{e \in (S; T)} f(e) \le \sum_{e \in (S; T)} c(e) = C(S, T).$$

Наслідок 1. *Якщо* $\mathrm{val}\,(f) = C(S, T)$ *при деяких потоці* f *та* $a - z$ *перерізі* $(S; T)$*, то* f *— максимальний потік,* (S, T) *— мінімальний переріз.*

Наслідок 2. *Рівність* $\mathrm{val}\,(f) = C(S, T)$ *справджується тоді й лише тоді, коли:*

$$\forall\, e \in (S; T) \quad f(e) = c(e);$$

$$\forall\, e \in (T; S) \quad f(e) = 0.$$

Пошук максимального потоку здійснимо шляхом збільшення потоку, починаючи з нульового, таким чином. Формуємо ланцюг — послідовність дуг, що сполучають вершини a – z без урахування напряму дуг. Для кожного такого ланцюга збільшуємо загальний потік, якщо це можливо, змінивши потік лише вздовж ланок ланцюга:

- збільшуємо потік вздовж дуг, спрямованих вздовж напрямку руху від a до z без виходу за межі пропускної спроможності;
- зменшуємо потік вздовж дуг, спрямованих проти напрямку руху від a до z без виходу за область невід'ємних чисел.

Кожній вершині поставимо у відповідність впорядковану пару, в якій:

- перший елемент — попередня вершина у побудованому ланцюгу;
- другий елемент — резерв, тобто величина, на яку можна:
    - збільшити потік, якщо напрям дуги збігається з напрямом руху від a до z;
    - зменшити потік, якщо напрям дуги протилежний до напряму руху від a до z.

Алгоритм Форда-Фалкерсона (Ford-Fulkerson algorithm)

1. Встановлюємо, що кожної вершини її попередник невизначений і резерв від'ємний (невизначений). Означимо резерв для вершини a як ∞, щоб не обмежувати резерв решти вершин. Покладемо $A = \{a\}$.
2. Якщо A — порожня множина, то припиняємо виконання алгоритму, бо потік максимізовано. Інакше вибираємо довільну вершину v з множини A і вилучаємо її з A.
3. Розглянемо всі вершини w, що задовольняють такі умови:
    - $(v; w)$ є дугою;
    - $f((v; w)) < c((v; w))$.

    Для кожної такої вершини w обчислимо мінімум з різниці $c((v; w)) - f((v; w))$ та резерву вершини v. Якщо обчислена на попередньому кроці величина більша за резерв вершини w, то:
    - замінюємо величину резерву w на цю величину;
    - (пере)означимо попередника w — вершину v;
    - якщо w відмінна від z, то долучаємо w до множини A.
4. Розглянемо всі вершини w, що задовольняють такі умови:
    - $(w; v)$ є дугою;
    - $0 < f((w; v))$.

    Для кожної такої вершини w обчислимо мінімум з $f((v;w))$ та резерву вершини v. Якщо обчислена на попередньому кроці величина більша за резерв вершини w, то:
    - замінюємо величину резерву w на цю величину;
    - (пере)означимо попередника w — вершину v;
    - якщо w відмінна від z, то долучаємо w до множини A.
5. Якщо резерв і попередник вершини z не визначено, то переходимо до виконання пункту 2.
6. Якщо резерв і попередник вершини z визначено, то:

- використовуючи попередників вершин, відновлюємо ланцюг $a - z$;
- для кожної дуги ланцюга, орієнтація якої збігається з напрямком руху від a до z вздовж ланцюга, збільшуємо потік на визначений резерв вершини z;
- для кожної дуги ланцюга, орієнтація якої протилежна до напрямку руху від a до z вздовж ланцюга, зменшуємо потік на визначений резерв вершини z;
- переходимо до виконання пункту 1.

Теорема 3. *Алгоритм Форда-Фалкерсона будує максимальний потік у мережі.*

Доведення. Позначимо через S множину всіх тих вершин, для яких визначено резерв під час останнього проходження алгоритму, $T = V \setminus S$. Множина S непорожня, бо містить вершину a. При цьому:

- якщо e — довільна дуга з $(S; T)$, то $f(e) = c(e)$, бо інакше для кінця дуги e визначено попередника;
- якщо e — довільна дуга з $(T; S)$, то $f(e) = 0$, бо інакше для кінця початку e визначено попередника.

Згідно з наслідками 1–2 f є максимальним потоком, а $(S; T)$ — мінімальним перерізом.

Наслідок 3. *Потік f у мережі є максимальним тоді й лише тоді, коли існує переріз $(S; T)$, при якому*

$$\mathrm{val}(f) = C(S, T).$$

Подамо приклад програми мовою Turbo Pascal 7.0 з коментарями щодо стуктури вхідних і вихідних даних.

```
{$I+}{$N+}                    {Верхні межі:}
const nv_max=2000;      {кількості вершин}
      l_max=2147483647;            {потоку}
type  pointe=^edge;
      edge=record                   {Дуга:}
      v,                          {початок}
      w: word;                      {кінець}
      c,         {пропускна спроможність}
      f: longint;                    {потік}
             {вказівник на наступну дугу}
      b,              {з тим самим початком}
      e: pointe end;           {... кінцем}

      pointa=^forsetA;
      forsetA=record   {Елемент списку A}
      v: word;                    {вершина}
      n: pointa end;  {наступний елемент}
   pointer=array[1..nv_max] of pointe;
```

```
   longers=array[1..nv_max] of longint;
var p:^pointer;    {Дуги з попередниками}
    r:^longers;          {Резерви вершин}
   n0,           {Вказівники на першу й}
   n,     {останню дуги з даним початком}
   m0,           {Вказівники на першу й}
m:^pointer; {останню дуги з даним кінцем}
  A1,           {Перший елемент списку A}
  A ,          {Поточний елемент списку A}
  Aw: pointa;        {Елемент w списку A}
   e: pointe;                    {Ребро}
  nv,                 {Кількість вершин}
   aa,                         {Джерело}
    z,                            {Стік}
  v,w: word;      {Дуга: початок, кінець}
    c,          {і пропускна спроможність}
   rw: longint; {Можливий резерв вершини}
    o: text;                {Файл даних}
 stop: boolean; {Потрібно припинити цикл}

       {Долучення вершини w до множини A}
procedure includew;                BEGIN
if A1=nil then begin
new(A1);  {A порожня}
A1^.v:=w;
A1^.n:=nil       end           else begin
A:=A1;    {A не порожня}
stop:=false;
while (A^.v<>w) and (A^.n<>nil)do A:=A^.n;
if     A^.v<>w then                begin
new(Aw);    {Якщо w не належить до A}
 A^.n:=Aw;
Aw^.v:=w;
Aw^.n:=nil                    end end END;

 {Запис у вихідний файл даних про потік:
 у кожному рядку - початок і кінець дуги
 й потік вздовж неї. Дуги перераховано у
 порядку зростання початку, а для сталого
 початку - у тому самому порядку, як вони
          зустрічалися у вхідному файлі}
procedure output;                  BEGIN
assign (o,'FLOW.RES');   rewrite(o);
for v:=1 to nv do                  begin
if n0^[v]<>nil then                begin
e:=n0^[v];
stop:=false;
repeat writeln(o,v,' ',e^.w,' ',e^.f);
       if e^.b=nil then stop:=true
                   else e:=e^.b
until stop                       end end;
close(o);  halt                      END;
                                   BEGIN
```

```
nv:=0;  new(n0); new(n); new(m0); new(m);
for v:=1 to nv_max do              begin
n0^[v]:=nil;
m0^[v]:=nil                          end;
assign(o,'FLOW.DAT');   reset(o);
{Зчитування рядків вхідного файлу, кожний
з яких містить: початок дуги, кінець дуги
                і пропускну спроможність}
REPEAT readln(o,v,w,c);
       if v>nv then nv:=v;
       if w>nv then nv:=w;
       new(e);
       e^.v:=v;
       e^.w:=w;
       e^.c:=c;
       e^.f:=0;
       e^.b:=nil;
       e^.e:=nil;
                 {Облік дуг з початком v}
       if n0^[v]=nil then n0^[v]   :=e
                     else  n^[v]^.b:=e;
       n^[v]:=e;
                   {Облік дуг з кінцем w}
       if m0^[w]=nil then m0^[w]   :=e
                     else  m^[w]^.e:=e;
       m^[w]:=e;
UNTIL  seekeof(o);        close(o);
new(p);   new(r);   new(A1);
aa:=1;                          {Джерело}
z:=nv;                             {Стік}
REPEAT                           {Крок 1}
for v:=1 to nv do begin
p^[v]:=nil;
r^[v]:=-1          end;
r^[aa]:=l_max;
A1^.v:=aa;
A1^.n:=nil;
REPEAT   if A1=nil then output else begin
                             {Кроки 2-5}
v:=A1^.v;    {Вилучення першого елемента
                            зі списку A}
if A1^.n=nil then A1:=nil
             else          begin
                   A:=A1;
                  A1:=A1^.n;
                  dispose(A) end;
n^[v]:=n0^[v];
stop:=false;
repeat               {Розгляд дуг (v;w)}
  w:=n^[v]^.w;
  if n^[v]^.f < n^[v]^.c       then begin
     rw:=n^[v]^.c-n^[v]^.f     ;
     if r^[v] < rw then rw:=r^[v];
```

```
     if r^[w] < rw then             begin
        r^[w]:=rw;
        p^[w]:=n^[v];
        if w<>z then includew     end end;
     if n^[v]^.b<>nil then n^[v]:=n^[v]^.b
                      else stop:=true;
until stop;
if m0^[v]<>nil then                 begin
   m^[v]:=m0^[v];
   stop:=false;
   repeat             {Розгляд дуг (w;v)}
     w:=m^[v]^.v;
     if  0 < m^[v]^.f          then begin
       rw:=m^[v]^.f;
        if r^[v] < rw then rw:=r^[v];
        if r^[w] < rw then          begin
           r^[w]:=rw;
           p^[w]:=m^[v];
           if w<>z then includew  end end;
     if m^[v]^.e<>nil then m^[v]:=m^[v]^.e
                      else stop:=true;
   until stop                     end end
UNTIL r^[z]>0;
w:=z;                              {Крок 6}
repeat e:=p^[w];
       if w =e^.w then   begin
          w:=e^.v;
          e^.f:=e^.f+r^[z] end else begin
          e^.f:=e^.f-r^[z];
          w:=e^.w                     end
until  w=aa;
UNTIL A1=nil;  output                 END.
```