Олександр Рудик

Розв'язання задачі про призначення

Задача 1. *Задано дві скінчені множини J і K з однаковою кількістю елементів і матрицю A, елементи якої мають індекси з J і K. Потрібно знайти взаємно однозначну відповідність f: J → K, при якій сума ементів $a_{j\,f(j)}$ за всіма j з J найменша.*

Зауваження 1. *Зауважимо таке.*

1. *При відображенні $g : K \to J$, оберненому до f, вказана в умові задачі сума дорівнює сумі за всіма k з K елементів $a_{g(k)\,k}$.*
2. *Розв'язок, тобто відображення f, зберігається при додаванні довільної сталої до всіх елементів будь-якого рядка чи стовпчика матриці A.*
3. *Розглянемо випадок, коли всі елементи матриці A невід'ємні. Побудуємо дводольний граф (J, K, E_0), у якому вершини j та k сполучено ребром тоді й лише тоді, коли $a_{jk} = 0$. Якщо у побудованому графі є повна сполука пар, то вона задає розв'язок задачі про призначення.*

Автор і походження назви. Американський математик Х.Кун (Harold D. Kuhn) вперше запропонував так званий угорський метод розв'язання цієї задачі. Він використовував теорію сполук пар, відому йому з робіт угорських учених Д.Кеніґа та Е.Еґеварі, тому й назвав метод угорським.

Угорський алгоритм Куна (Kuhn)

1. **Зведення рядків матриці.** Для кожного рядка матриці *A*:
   - визначаємо найменший елемент у вибраному рядку;
   - кожний елемент вибраного рядка зменшуємо на знайдений найменший елемент рядка.

   Перетворена таким чином матриця міститиме нуль (можливо, не один) у кожному рядку.
2. **Зведення стовпчиків матриці.** Для кожного стовпчика матриці *A*:
   - визначаємо найменший елемент у вибраному стовпчику;
   - кожний елемент вибраного стовпчика зменшуємо на знайдений найменший елемент стовпчика.

   Перетворена таким чином матриця міститиме нуль (можливо, не один) у кожному стовпчику.
3. **Побудова максимальної сполуки пар** дводольного графа (*J, K, E*), у якому вершини *j* та *k* сполучено ребром тоді й лише тоді, коли $a_{jk} = 0$.
4. Знаходимо J_1 та K_1 — відповідно множини усіх вершин з *J* та *K*, що ненасичені сполукою пар, побудованою на попередньому кроці. Якщо ці множини непорожні, тобто побудована сполука пар неповна, то робимо таке *збільшення кількості нулів матриці:*

- дводольний граф (J, K, E) перетворимо на орієнтований, спрямувавши дугу з K до J, якщо її кінці є елементами однієї пари, і спрямувавши дугу з J до K в іншому випадку;
- знайдемо вершини, досяжні у побудованому орієнтованому графі з множини J_1. Ці вершини належать до однієї з двох множин J_2 та K_2, що є відповідно підмножинами J та K. Маємо:
  - J_1 — підмножина J_2, бо її досягають за нуль кроків, рухаючись дугами орієнтованого графа;
  - K_2 — підмножина $K \setminus K_1$, інакше в орієнтованому графі існує маршрут непарної довжини, у якому:
    - на непарних місцях розташовано дуги, побудовані за ребрами, що не належать до сполуки пар;
    - на парних місцях розташовані дуги, побудовані за ребрами, що належать до сполуки пар;
    - початкова й кінцева вершини не насичені сполукою пар.

    У цьому випадку кількість ребер сполуки пар можна збільшити щонайменше на 1, що суперечить твердженню про її максимальність;
  - K_1 — підмножина $K \setminus K_2$ (наслідок попереднього зауваження);
  - якщо $a_{jk} = 0$, то j належить до $J \setminus J_2$ або k належить до K_2, причому вказані дві умови несумісні;
  - якщо j належить до J_2 і k належить до $K \setminus K_2$, то $a_{jk} > 0$;
- знаходимо додатне число $a_{\min}$ — найменший елемент a_{jk} при:
  - j з J_2 (остання множина непорожня, бо містить непорожню J_1 як підмножину);
  - k з $K \setminus K_2$ (остання множина непорожня, бо містить непорожню K_1 як підмножину);
- до всіх елементів стовпчиків k з K_2 додаємо $a_{\min}$;
- від усіх елементів рядків j з J_2 віднімаємо $a_{\min}$;
- після збільшення кількості нульових елементів матриці внаслідок виконання двох останніх кроків (нулі матриці, що існували до виконання двох останніх кроків, залишаються нулями, але з'явиться щонайменше один новий нуль) переходимо до виконання пункту 3 даного алгоритму.

Необмежуючи загальності міркувань (інакше потрібно перенумерувати елементи J і K відповідним чином), можна вважати, що перші $| J_2 |$ елементів множини J належать до J_2, а останні $| K_2 |$ елементів множини K належать до K_2.

Унаочнимо крок збільшення кількості нульових елементів матриці у цьому випадку такою таблицею.

Таблиця 1

Збільшення кількості нульових елементів матриці

	$K \setminus K_2$	K_2
J_2	$- a_{\min}$	
$J \setminus J_2$		$+ a_{\min}$

Коментар до Таблиці 1

1. Ліворуч вказано область значень індексу j з множини J;
2. Вгорі вказано область значень індексу k з множини K;
3. Світлосірим тлом позначено додатні елементи матриці, серед яких вибирають найменший елемент $a_{\min}$ ($j \in J_2$, $k \in K \setminus K_2$). Внаслідок зменшення на $a_{\min}$ хоча б один з них перетворюють на 0.
4. Білим тлом позначено елементи матриці, які можуть дорівнювати нулю і які залишають тими самими при перетворенні:
   - або з ними нічого не роблять взагалі (лівий нижній кут, $j \in J \setminus J_2$, $k \in K \setminus K_2$);
   - або їх збільшують, а потім зменшують на $a_{\min}$ (правий верхній кут, $j \in J_2$, $k \in K_2$).
5. Темносірим тлом позначено додатні елементи матриці, які збільшують на $a_{\min}$ ($j \in J \setminus J_2$, $k \in K_2$).

Подамо приклад програми мовою Turbo Pascal 7.0, що використовує описаний алгоритм для невід'ємних цілих чисел у межах базового типу змінних word.

```
{$N+}
```

```
const
  filein ='HUNGAR.DAT';  {Вхідний файл}
  fileout='HUNGAR.RES'; {Вихідний файл}
    n_max=205; {Верхня межа розміру матриці}
type  num=word;      {Тип елементів матриці}
    linen=array[1..n_max] of num;
    linew=array[0..n_max] of word;
   plinen=^linen;
   plinew=^linew;
  matrixn=array[1..n_max] of plinen;
  matrixw=array[1..n_max] of plinew;
   labels=array[0..n_max] of boolean;
    chain=array[0..n_max*2] of word;
var a:^matrixn;                  {Матриця A}
   k0,            {Дані про нулі матриці A:
    k0[j.0] - кількість елементів a[j,k]=0;
    k0[j.k] при k=1, 2, ... , k0[j.0] -
          номери відповідних стовпчиків k}
   j0:^matrixw;   {Дані про нулі матриці A:
    j0[k.0] - кількість елементів a[j,k]=0;
    j0[k.j] при j=1, 2, ... , j0[k.0] -
                номери відповідних рядків}
    c,             {Граф M-змінних ланцюгів}
   nc,     {Вказівник на масив даних с про
      граф найкоротших M-чергових ланцюгів}
 newc:^chain;             {M-змінний ланцюг}
   jp,      {Номер відповідного елемента J}
   kp:^linew;                          { K
              для побудованої сполуки пар.
       0, якщо елемент відповідно з K чи J
                не входить до сполуки пар}
j2,k2,          {Належність до множин J2,K2}
        {При побудові M-чергового ланцюга:}
usedj,              {використано вершину J}
usedk,              {використано вершину K}
stop:^labels;                            {
stop[k] -існування M-чергового ланцюга:
      при k = 0 - взагалі;
      при k > 0 - з кінцем у вершині k з K}

 done: boolean;     {Побудовано M-черговий
                    з вибраним закінченням}
  min: num;             {Найменший елемент}
    n,     {Кількість: рядків (стовпчиків)}
    np,                     {пар у сполуці}
    j,                       {Номер: рядка}
    k,                          {стовпчика}
    nm,  {Максимальна кількість M-чергових
               ланцюгів без спільних точок}
    i,    {Збільшений на 1 рівень побудови
     графа найкоротший M-чергових ланцюгів}
    im,     {Кількіcnm вершин найкоротшого
                       M-чергового ланцюга}
```

```
    ip,{На скільки збільшено кількість пар}
    l,m: word;                  {Лічильники}
    o: text;                    {Файл даних}
  sum: extended;     {Шукана найменша сума}

 {Побудова найкоротшого M-чергового ланцюга}
procedure getchain (i,l : word);
             var m,ii,ll: word;       BEGIN
case i mod 2 of
     0: if usedk^[c^[l]] then exit;
     1: if usedj^[c^[l]] then exit end;
newc^[i]:=c^[l];
if i=1 then                              begin
        {Збільшення кількості пар на 1}
  for m:=1 to im do case m mod 2 of 0:begin
       usedk^[newc^[m]]:=true;
          jp^[newc^[m]]:=newc^[m-1]     end;
                                     1:begin
       usedj^[newc^[m]]:=true;
          kp^[newc^[m]]:=newc^[m+1] end end;
  done:=true;  inc(np);  inc(ip)        end
  else{Пошук невідомого кінця ланцюга}begin
  ii:=i-1;
  ll:=nc^[ii-1];
  repeat inc(ll); case i mod 2 of
   0: if a^[c^[ll]]^[c^[l]]=0
                  then getchain(ii,ll);
   1: if a^[c^[l]]^[c^[ll]]=0
                  then getchain(ii,ll) end;
   until done or (ll=nc^[ii])       end END;
                                      BEGIN
new(c);   new(nc);  new(newc);   new(stop);
new(j0);  new(jp);  new(usedj);  new(j2);
new(k0);  new(kp);  new(usedk);  new(k2);
new(a);           {Зчитування вхідних даних}
assign(o,filein);   reset(o);    read(o,n);
                     for j:=1 to n do begin
     new(a^[j]);  new(j0^[j]);  new(k0^[j]);
     for k:=1 to n do read(o,a^[j]^[k]) end;
close(o);
nc^[0]:=0;
               {1. Зведення рядків матриці}
for j:=1 to n do                         begin
  min:=a^[j]^[1];
  for k:=2 to n do if   min >a^[j]^[k]
                  then min:=a^[j]^[k];
  if min>0 then for k:=1 to n do
              a^[j]^[k]:=a^[j]^[k]-min end;

            {2. Зведення стовпчиків матриці}
for k:=1 to n do                         begin
  min:=a^[1]^[k];
  for j:=2 to n do if   min > a^[j]^[k]
```

```
                  then min :=a^[j]^[k];
  if min>0 then for j:=1 to n do
              a^[j]^[k]:=a^[j]^[k]-min end;

      {3. Побудова максимальної сполуки пар}

REPEAT      {Визначення нульових елеммментів}
                     for j:=1 to n do
                     for k:=0 to n do begin
                          j0^[j]^[k]:=0;
                          k0^[j]^[k]:=0 end;
for j:=1 to n do
for k:=1 to n do if a^[j]^[k]=0 then  begin
inc(j0^[k]^[0]); j0^[k]^[j0^[k]^[0]]:=j;
inc(k0^[j]^[0]); k0^[j]^[k0^[j]^[0]]:=k end;

                      {Побудова сполуки пар}
np:=0;                for j:=1 to n do begin
                                jp^[j]:=0;
                                kp^[j]:=0 end;
for k:=1 to n do
if (j0^[k]^[0]>0) and (jp^[k]=0) then begin
 l:=0;
 repeat inc(l);
  until (kp^[j0^[k]^[l]]=0)or(l=j0^[k]^[0]);
     if (kp^[j0^[k]^[l]]=0) then      begin
                        jp^[k] :=j0^[k]^[l];
                   kp^[jp^[k]]:=k;
                        inc(np)     end end;

  {Власне побудова максимальної сполуки пар
           за допомогою М-чергових ланцюгів}
repeat                for l:=0 to n do begin
                         stop^[l]:=false;
                       usedj^[l]:=false;
                       usedk^[l]:=false end;
                {Побудова графа найкоротших}
  i :=1;                {М-чергових ланцюгів}
  ip:=0;
  nc^[1]:=0;                {Нульовий рівень}
  for j:=1 to n do
  if (kp^[j]=0)and(k0^[j]^[0]>0) then begin
                          inc(nc^[1]);
                           c^[nc^[1]]:=j end;
  repeat
    inc(i);                  {Непарний рівень}
    nc^[i]:=nc^[i-1];
    for  m:=nc^[i-2]+1 to nc^[i-1] do begin
    j:=c^[m];
    for l:=1 to k0^[j]^[0] do
     if kp^[j]<>k0^[j]^[l] then       begin
     inc(nc^[i]);
      c^[nc^[i]]:=k0^[j]^[l];
```

```
     stop^[k0^[j]^[l]]:=
     stop^[k0^[j]^[l]]or(jp^[k0^[j]^[l]]=0);
     stop^[0]:=stop^[0] or stop^[k0^[j]^[l]]
                                    end end;
    if not stop^[0] and (nc^[i-1] < nc^[i])
                                  then begin
      inc(i);                {Парний рівень}
      nc^[i]:=nc^[i-1];
      for m:=nc^[i-2]+1 to nc^[i-1] do
      if jp^[c^[m]]>0 then            begin
            inc(nc^[i]);
             c^[nc^[i]]:=jp^[c^[m]] end end
  until stop^[0] or (nc^[i]=nc^[i-1]);

  if stop^[0] and (i>=4) then         begin
  {Збільшення кількості пар з використанням
                        М-чергових ланцюгів}
  im:=i;
  for m:=nc^[im-1]+1 to nc^[im] do
                if stop^[c^[m]] then begin
                     done:=false;
                     getchain(i,m) end end
until (ip=0) or (np=n);

     {4. Збільшення кількості нулів матриці}
if np < n                           then BEGIN
                    for j:=1 to n do begin
                         j2^[j]:=false;
                         k2^[j]:=false end;
nc^[1]:=0;  i:=1;
for j:=1 to n do    {Знаходження множини J1}
if (k0^[j]^[0]=0) or
   (k0^[j]^[0]>0) and (kp^[j]=0) then begin
                         inc(nc^[i]);
                          c^[nc^[i]]:=j;
                          j2^[j]:=true end;
repeat
   inc(i);        {Знаходження підмножини K2}
   nc^[i]:=nc^[i-1];
   for  l:=nc^[i-2]+1 to nc^[i-1] do
         if 0 < k0^[c^[l]]^[0] then
   for m:=1 to  k0^[c^[l]]^[0] do
   if  not  k2^[k0^[c^[l]]^[m]] and
   (kp^[c^[l]]<glt;k0^[c^[l]]^[m])  then begin
               inc(nc^[i]);
                c^[nc^[i]]:=k0^[c^[l]]^[m];
            k2^[c^[nc^[i]]]:=true      end;
   if nc^[i]>nc^[i-1]            then begin
       inc(i);   {Знаходження підмножини J2}
       nc^[i]:=nc^[i-1];
       for l :=nc^[i-2]+1 to nc^[i-1] do
       if     (jp^[c^[l]]>0) and
       not j2^[jp^[c^[l]]]       then begin
```

```
                inc(nc^[i]);
                 c^[nc^[i]]:=jp^[c^[l]];
             j2^[c^[nc^[i]]]:=true end end
until nc^[i]=nc^[i-1];
min:=0;
for j:=1 to n do if     j2^[j] then
for k:=1 to n do if not k2^[k] then
if  (min=0) or (min>a^[j]^[k])
then min:=a^[j]^[k];
for k:=1 to n do if k2^[k] then
for j:=1 to n do a^[j]^[k]:=a^[j]^[k]+min;
for j:=1 to n do if j2^[j] then
for k:=1 to n do a^[j]^[k]:=a^[j]^[k]-min
                                        END;
UNTIL np=n;
assign(o,filein);  reset(o);  read(o,n);
for j:=1 to n do
for k:=1 to n do read(o,a^[j]^[k]);
close(o);
sum:=0;
for j:=1 to n do sum:=sum+a^[j]^[kp^[j]];
                          {Запис відповіді}
assign(o,fileout);  rewrite(o);
writeln(o,sum:0:0);
for j:=1 to n-1 do write  (o,kp^[j],' ');
                  writeln(o,kp^[n]);
close(o);              {Вивільнення пам'яті}
for j:=1 to n do begin
dispose( a^[j]);
dispose(j0^[j]);
dispose(k0^[j])    end;
dispose(a);  dispose(j0);    dispose(k0);
dispose(c);  dispose(nc);    dispose(newc);
dispose(jp); dispose(usedj); dispose(j2);
dispose(kp); dispose(usedk); dispose(k2);
dispose(stop)                           END.
```