INFORMATIKA

Optimalizace tras přívozů
(Úlohy z MO – kategorie P, 33. část)

PAVEL TÖPFER

Matematicko-fyzikální fakulta UK, Praha

Pro dnešní pokračování naší dlouhodobé série článků o úlohách z Matematické olympiády kategorie P jsme si vybrali jednu úplně čerstvou úlohu. Byla zadána v březnu 2015 jako praktická úloha ústředního kola 64. ročníku MO (školní rok 2014/15). Ačkoliv za ni téměř každý z řešitelů získal nějaké body, zcela správně na plný počet bodů ji nevyřešil nikdo. Podle původního zadání bylo totiž zapotřebí nejen navrhnout optimální vedení tras budovaných přívozů, ale navíc také vypořádat se s rozsáhlými vstupními daty a přitom omezeným množstvím operační paměti. Ukládaná data bylo proto nutné složitě komprimovat, což působilo řešitelům značné problémy. My si původní zadání trochu zjednodušíme a budeme se v článku věnovat pouze té zajímavější části řešeného problému – tedy návrhu vlastního algoritmu. Vhodným způsobem proto snížíme velikost zpracovávaných vstupních údajů, abychom neměli žádné technické problémy s uložením dat ve vnitřní paměti počítače. Formulaci zadání zároveň mírně upravíme, aniž bychom tím ovšem změnili podstatu řešené úlohy.

* * * * * * * * * * * *

Kocourkovští radní přemýšleli, jak by do svého města mohli nalákat více turistů. Při zkoumání turistických prospektů zjistili, že jejich město nemá na rozdíl od mnoha dalších svůj aquapark, řeku s nábřežími, lyžařské středisko a mnoho dalších lákadel. Protože však byli limitováni rozpočtem města, rozhodli se vybudovat jen jednu z těchto atrakcí a volba padla na řeku. Naplánovali tedy výstavbu dlouhého vodního kanálu. Na každé jeho straně vybrali místa, kde by mohly být mosty. Bohužel koordinace

plánování selhala. Vybraná místa nebyla na protilehlých pozicích podél kanálu a co hůř, vícepráce při stavbě kanálu způsobily, že na výstavbu mostů již nezbyly žádné peníze. Radní se proto rozhodli mezi vybranými místy místo mostů zřídit přívozy.

Soutěžní úloha

Vaším úkolem je určit dvojice míst na levé a pravé straně kanálu, která mají být spojena přívozy tak, aby každé vybrané místo bylo spojeno přívozem alespoň s jedním vybraným místem na druhé straně kanálu a zároveň aby součet délek všech zřízených přívozů byl nejmenší možný. Pokud existuje více takových řešení, vypište jedno libovolné z nich.

Formát vstupu

První řádek vstupu obsahuje čtyři celá čísla L, S, A, B oddělená mezerami. Tato čísla udávají postupně délku kanálu, jeho šířku a počet vybraných míst na levé a na pravé straně. Můžete předpokládat, že platí $1 \leq L \leq 1\,000\,000$, $1 \leq S \leq 1\,000$ a $1 \leq A, B \leq 1\,000$. Následuje A řádků, z nichž i-tý obsahuje vzdálenost a_i i-tého vybraného místa na levé straně kanálu od počátku kanálu. Poté následuje B řádků, z nichž j-tý obsahuje vzdálenost b_j j-tého vybraného místa na pravé straně kanálu od jeho počátku. Můžete předpokládat, že všechny vzdálenosti a_i (pro $1 \leq i \leq A$), b_j (pro $1 \leq j \leq B$) jsou celá čísla, žádná dvě vybraná místa nejsou stejná a na vstupu jsou polohy vybraných míst zadány podle rostoucí vzdálenosti od počátku kanálu, tedy že platí $0 \leq a_1 < \ldots < a_A \leq L$ a $0 \leq b_1 < \ldots < b_B \leq L$.

Formát výstupu

Každý řádek výstupu obsahuje dvě celá čísla i a j, která udávají, že mezi i-tým místem na levé straně kanálu a j-tým místem na pravé straně kanálu má být zřízen přívoz. Na pořadí řádků ve výstupu nezáleží.

Příklad 1.

Vstup:	Výstup:
10 5 3 2	1 1
0	2 2
5	3 2
10	
2	
8	

Příklad 2.

Vstup:	Výstup:
1000 1 3 3	1 1
0	2 1
1	3 2
1000	3 3
0	
999	
1000	

Při rozboru úlohy si nejprve všimněte několika důležitých věcí. Především si uvědomte, že v optimálním řešení se trasy žádných dvou přívozů nekříží. Toto pozorování snadno dokážeme pomocí trojúhelníkové nerovnosti. Kdyby se trasa přívozu spojujícího místa a, b křížila s trasou přívozu spojujícího místa a', b' (tzn. $a < a'$, $b' < b$), potom by se vyplatilo vyměnit tyto dva přívozy za přívoz spojující dvojici míst a, b' a přívoz spojující dvojici míst a', b. Vzniklo by tím jiné přípustné řešení, ve kterém je součet délek tras všech přívozů menší. Označme si průsečík tras původních přívozů symbolem s. Součet jejich délek můžeme vyjádřit jako

$$|ab| + |a'b'| = (|as| + |sb|) + (|a's| + |sb'|) = (|as| + |sb'|) + (|a's| + |sb|).$$

Obsah první závorky je podle trojúhelníkové nerovnosti větší než $|ab'|$, zatímco obsah druhé nerovnosti je opět podle trojúhelníkové nerovnosti větší než $|a'b|$ (nakreslete si obrázek!). Celkově proto skutečně platí

$$|ab| + |a'b'| > |ab'| + |a'b|.$$

Dále si povšimněte, že se nemusí vyplatit usilovat o řešení s minimálním počtem přívozů. Tuto skutečnost dobře demonstruje příklad 2 ze zadání úlohy. V něm máme tři vybraná místa na levé straně kanálu a tři vybraná místa na pravé straně kanálu, takže jistě existuje řešení se třemi přívozy, ve kterém z každého vybraného místa vyjíždí právě jeden přívoz. Jelikož už víme, že trasy přívozů se nesmí křížit, takové řešení existuje jediné a je tvořeno přívozy 1-1, 2-2, 3-3. Vzhledem k poloze vybraných míst toto řešení ale není optimální z hlediska celkové délky tras přívozů, výhodnější je zvolit čtyři přívozy 1-1, 2-1, 3-2, 3-3. K řešení úlohy tedy nelze použít nějaký jednoduchý hladový algoritmus, který by prostě spojoval dvojice míst na obou stranách kanálu v tom pořadí, jak po sobě následují od začátku kanálu.

Správné řešení úlohy je založeno na myšlence dynamického programování. Abychom si postup řešení dobře vysvětlili, budeme v první etapě hledat pouze nejmenší možnou celkovou délku tras všech vybudovaných přívozů. Teprve až vyřešíme tuto část úlohy, program doplníme ještě o určení a vypsání dvojic míst, která jsou jednotlivými přívozy spojena.

Vytvoříme si dvojrozměrnou tabulku T velikosti $A \times B$ prvků (hodnoty A, B podle zadání udávají, kolik bylo vybráno míst na levém a na pravém břehu kanálu). Číslo $T[i, j]$ bude udávat nejmenší možný součet délek tras přívozů, které spojují prvních i míst na levé straně kanálu s prvními j místy na jeho pravé straně. Jak tuto hodnotu určíme? V každém optimálním řešení musí být spolu spojeny přívozem poslední místo na levé straně kanálu (tzn. i-té místo) a poslední místo na pravé straně kanálu (tzn. j-té místo), jinak by se trasy některých přívozů křížily. Navíc pro $i \geq 2$ a $j \geq 2$ bude v libovolném optimálním řešení přívozem spojena právě jedna z následujících tří dvojic míst: $(i-1)$-té místo a j-té místo, i-té místo a $(j-1)$-té místo, nebo $(i-1)$-té místo a $(j-1)$-té místo. Jinak by se opět buď některé trasy přívozů křížily, nebo by do některého vybraného místa nevedl žádný přívoz. Aby bylo $T[i, j]$ minimální možné, musí tedy pro $i \geq 2$ a $j \geq 2$ platit

$$T[i, j] = d(i, j) + \min(T[i-1, j], T[i, j-1], T[i-1, j-1]).$$

Výraz $d(i, j)$ zde udává vzdálenost i-tého místa na levé straně a j-tého místa na pravé straně kanálu. Hodnoty $d(i, j)$ nám bude snadno počítat pomocná funkce pomocí Pythagorovy věty – je to vzdálenost dvou bodů v rovině, přičemž známe kartézské souřadnice obou těchto bodů.

Uvedený rekurzivní vztah je základem řešení úlohy pomocí metody dynamického programování. Hodnoty $T[i, j]$ nebudeme počítat rekurzivní funkcí, i když i takové řešení by bylo možné. Raději však budeme postupně vyplňovat tabulku T v pořadí podle rostoucích hodnot indexů i, j – tedy v takovém pořadí, abychom při výpočtu prvku $T[i, j]$ již znali všechny tři hodnoty obsažené v uvedeném rekurzivním vzorci. Začneme od prvků s indexem $i = 1$ nebo $j = 1$, další prvky tabulky T potom můžeme počítat třeba po řádcích zleva doprava (existují ale i jiné možnosti vhodného pořadí výpočtu). Hledaným konečným výsledkem pak bude hodnota prvku $T[A, B]$. Tu vypočítáme jako úplně poslední po vyplnění celé tabulky T.

Zbývá ukázat správné nastavení počátečních hodnot tabulky T, tedy hodnot pro $i = 1$ nebo $j = 1$. Máme-li na jedné straně kanálu jediné

vybrané místo, musíme toto jediné místo nutně spojit přívozy se všemi vybranými místy na druhé straně kanálu. Je tedy zřejmé, že hodnota $T[1, j]$ je rovna součtu všech vzdáleností $d(1, j)$ pro $1 \leq j \leq B$ a podobně hodnota $T[i, 1]$ je rovna součtu všech vzdáleností $d(i, 1)$ pro $1 \leq i \leq A$. Tyto hodnoty spočítáme nejlépe postupným přičítáním délek přívozů. Začneme od prvku $T[1, 1] = d(1, 1)$. Pro rostoucí index $j > 1$ pak počítáme

$$T[1, j] = T[1, j - 1] + d(1, j).$$

Obdobně postupujeme při výpočtu hodnot $T[i, 1]$.

Asymptotická časová složitost popsaného řešení je $O(AB)$. Počítáme $A \cdot B$ prvků tabulky T, výpočet každého z nich zvládneme podle výše uvedeného vzorce v konstantním čase. Při omezení hodnot $A, B \leq 1\,000$ to představuje reálnou dobu výpočtu ve zlomcích sekundy. Paměťová složitost popsaného postupu vychází rovněž $O(AB)$, což je velikost jediné datové struktury T použité v řešení.

Právě popsané řešení si nyní ukážeme zapsané v programovacím jazyce Pascal:

```
program Privozy1;
{základní řešení s dvojrozměrnou tabulkou T}

const Max = 1000; {počet vybraných míst na každé straně kanálu}
var L: longint;  {délka kanálu}
    S: integer;  {šířka kanálu}
    A, B: integer;  {počet vybraných míst vlevo a vpravo}
    aa, bb: array[1..Max] of longint;  {poloha vybraných míst}
    T: array[1..Max, 1..Max] of real;  {součty délek přívozů}
    i, j: integer;

function d(i, j: integer): real;
{počítá vzdálenost i-tého místa vlevo od j-tého místa vpravo}
begin
  d := sqrt(sqr(s) + sqr(aa[i]-bb[j]))
end; {function d}

function min(K, L, M: real): real;
{minimum ze tří reálných čísel}
var Q: real;
begin
  if K < L then Q := K else Q := L;
  if M < Q then Q := M;
  min := Q
end; {function min}
```

```
begin
readln(L, S, A, B);
for i:=1 to A do readln(aa[i]);
for j:=1 to B do readln(bb[j]);

T[1,1]:=d(1,1);
{první řádek tabulky T}
for j:=2 to B do T[1,j]:=T[1,j-1] + d(1,j);
for i:=2 to A do  {i-tý řádek tabulky T pro i>=2}
  begin
  T[i,1]:=T[i-1,1] + d(i,1);
  for j:=2 to B do  {počítáme prvek T[i,j]}
    T[i,j]:=d(i,j) + min(T[i-1,j], T[i,j-1], T[i-1,j-1]);
  end;

writeln(T[A,B]:8:2)  {výsledkem je hodnota T[A,B]}
end.
```

Všimněte si, že v uvedeném řešení úlohy počítáme hodnoty prvků v tabulce T postupně po řádcích a že hodnoty prvků v i-tém řádku závisejí pouze na hodnotách z předchozího $(i-1)$-tého řádku, nikoliv na hodnotách z řádků vzdálenějších. Místo celého dvojrozměrného pole T, které jsme uvažovali v odvození algoritmu, můžeme proto při implementaci použít dvě jednorozměrná pole, která budou vždy představovat aktuální a předcházející řádek tabulky. Paměťová složitost našeho řešení se tím výrazně sníží na $O(B)$. Toto je běžný trik, který se při implementaci dynamického programování často využívá. Nepotřebujeme dokonce ani dvě jednorozměrná pole, postačí nám pouze jedno jednorozměrné pole U (a k němu ještě pomocná proměnná X), ve kterém budeme vždy hodnoty nově počítaného i-tého řádku tabulky T přepisovat přes hodnoty předchozího $(i-1)$-tého řádku. V pomocné proměnné X si přitom budeme vždy udržovat hodnotu $T[i-1, j-1]$, která už sice byla v poli U na indexu $j-1$ přepsána hodnotou $T[i, j-1]$, ale my ji ještě budeme potřebovat při výpočtu následujícího prvku $T[i, j]$.

Uvedenou paměťovou optimalizaci si ukážeme ve formě upraveného programu:

```
program Privozy2;
{optimalizace paměti -
 dvojrozměrná tabulka T je nahrazena jednorozměrným polem U}

const Max = 1000; {počet vybraných míst na každé straně kanálu}
var L: longint;  {délka kanálu}
    S: integer;  {šířka kanálu}
    A, B: integer;  {počet vybraných míst vlevo a vpravo}
```

```
    aa, bb: array[1..Max] of longint;  {polohy vybraných míst}
    {součty délek přívozů = řádky tabulky T}
    U: array[1..Max] of real;
    X, Y: real;  {pracovní pro zaplňování pole U}
    i, j: integer;

function d(i, j: integer): real;
{počítá vzdálenost i-tého místa vlevo od j-tého místa vpravo}
begin
  d := sqrt(sqr(s) + sqr(aa[i]-bb[j]))
end; {function d}

function min(K, L, M: real): real;
{minimum ze tří reálných čísel}
var Q: real;
begin
  if K < L then Q := K else Q := L;
  if M < Q then Q := M;
  min := Q
end; {function min}

begin
readln(L, S, A, B);
for i:=1 to A do readln(aa[i]);
for j:=1 to B do readln(bb[j]);
U[1]:=d(1,1);
for j:=2 to B do U[j]:=U[j-1] + d(1,j); {první řádek tabulky T}
for i:=2 to A do  {i-tý řádek tabulky T pro i>=2}
  begin
  X := U[1];  {zaznamenáme si T[i-1,1]}
  U[1] := U[1] + d(i,1);  {prvek T[i,1]}
  for j:=2 to B do  {počítáme prvek T[i,j]}
    begin
    Y := d(i,j) + min(U[j], U[j-1], X);
    X := U[j];
    U[j] := Y
    end
  end;

writeln(U[B]:8:2)  {výsledkem je hodnota T[A,B]}
end.
```

Dosud jsme řešili zjednodušenou úlohu určit pouze minimální možný součet délek všech přívozů, které spojují všechna vybraná místa. Vraťme se nyní k původnímu zadání úlohy, kde bylo úkolem vypsat konkrétní dvojice míst spojených přívozem v tomto optimálním řešení. K tomu použijeme druhou dvojrozměrnou tabulku V velikosti $A \times B$ prvků, kterou budeme

vyplňovat souběžně s tabulkou T. Pro $i \geq 2$ a $j \geq 2$ se hodnota $T[i,j]$ počítá jako minimum ze tří případů, které mohou nastat. Do prvku $V[i,j]$ si při tom uložíme informaci, který z těchto tří případů nastal. Je úplně jedno, jak si tuto informaci zakódujeme – můžeme použít třeba hodnoty 1, 2, 3 jako v našem ukázkovém programu. Hodnoty $V[i,j]$ pro $i = 1$ a $j = 1$ dodefinujme odpovídajícím způsobem. Po zaplnění obou tabulek dokážeme z údajů uložených v tabulce V zpětně zrekonstruovat celé řešení, tzn. určit vedení tras jednotlivých přívozů. Vypíšeme nejprve přívoz $A-B$ spojující poslední vybraná místa po obou stranách kanálu a podle hodnoty $V[A,B]$ se přesuneme do stavu $V[A-1,B]$ nebo $V[A,B-1]$ nebo $V[A-1,B-1]$. Tam postupujme obdobně, dokud nedojdeme až k přívozu 1-1, který spojuje první vybraná místa po obou stranách kanálu. Podrobněji je tento postup předveden v závěrečné programové ukázce. Také tento zpětný průchod řízený druhým polem V je naprosto ukázkový postup, jaký se často využívá při řešení úloh dynamickým programováním.

Provedená úprava algoritmu nic nezměnila na asymptotické časové složitosti výpočtu $O(AB)$. Tabulka V má stejně jako pole T velikost $A \times B$ a každý její prvek určíme také v konstantním čase. Při zpětné rekonstrukci tras všech budovaných přívozů určíme podle údajů v tabulce V každou jednu trasu přívozu v konstantním čase a jejich počet bude jistě menší než $A+B$, když se žádné dvě trasy nesmějí křížit. Paměťová složitost algoritmu je $O(AB)$, což je dáno velikostí tabulky V. Její velikost nelze jednoduše zredukovat tak, jako jsme to provedli u předchozího řešení s tabulkou T.

Na závěr si předvedeme řešení celé úlohy i s výpisem nalezených tras přívozů. Pro větší názornost vezmeme za základ výsledného programu naše původní řešení s dvojrozměrnou tabulkou T, tedy bez zmiňované paměťové optimalizace. Vzhledem k přidané druhé tabulce V bude stejně paměťová složitost programu kvadratická.

```
program Privozy3; {řešení Privozy1 doplněné o určení tras}

const Max = 1000; {počet vybraných míst na každé straně kanálu}
var L: longint;  {délka kanálu}
    S: integer;  {šířka kanálu}
    A, B: integer;  {počet vybraných míst vlevo a vpravo}
    aa, bb: array[1..Max] of longint;  {polohy vybraných míst}
    T: array[1..Max, 1..Max] of real;  {součty délek přívozů}
    V: array[1..Max, 1..Max] of byte;  {určení pozic přívozů}
       {hodnoty: 1 = snížit 1. souřadnici,
                 2 = snížit 2. souřadnici,
                 3 = snížit obě souřadnice}
    i, j: integer;
```

```
function d(i, j: integer): real;
{počítá vzdálenost i-tého místa vlevo od j-tého místa vpravo}
begin
  d:=sqrt(sqr(s) + sqr(aa[i]-bb[j]))
end; {function d}

function min(K, L, M: real; var Kde: byte): real;
{minimum ze tří reálných čísel}
{výstupní parametr Kde určuje, kolikátý parametr ji minimální}
var Q: real;
begin
  if K < L then begin Q:=K;Kde:=1 end else begin Q:=L;Kde:=2 end;
  if M < Q then begin Q:=M; Kde:=3 end;
  min := Q
end; {function min}

begin
readln(L, S, A, B);
for i:=1 to A do readln(aa[i]);
for j:=1 to B do readln(bb[j]);

T[1,1] := d(1,1);
for j:=2 to B do
  begin
  T[1,j] := T[1,j-1] + d(1,j); V[1,j] := 2
  end; {první řádek T}
for i:=2 to A do  {i-tý řádek tabulky T pro i>=2}
  begin
  T[i,1] := T[i-1,1] + d(i,1); V[i,1]:=1;
  for j:=2 to B do
    {počítáme prvek T[i,j], zároveň určíme V[i,j]}
    T[i,j]:=d(i,j) + min(T[i-1,j],T[i,j-1],T[i-1,j-1],V[i,j]);
  end;

i := A; j := B;
writeln(i, ' ', j);
while (i>1) or (j>1) do
  begin
  case V[i,j] of
  1: dec(i);
  2: dec(j);
  3: begin dec(i); dec(j) end
  end;
  writeln(i, ' ', j)
  end;
end.
```