INFORMATIKA

Plánování zakázek aneb Bořivojova prádelna

(Úlohy z MO – kategorie P, 13. část)

PAVEL TÖPFER

Matematicko-fyzikální fakulta UK, Praha

Problematika rozvrhování zakázek a plánování jejich zpracování patří mezi tradiční úlohy řešené na počítačích. V našich programátorských soutěžích pro středoškoláky se však toto téma vyskytuje poměrně zřídka. V minulém 54. ročníku Matematické olympiády – kategorie P se objevila jedna úloha z této oblasti v domácím kole a na ni navázala podobná úloha i v kole krajském. Obě úlohy a jejich autorská řešení připravili členové úlohové komise MO-P z MFF UK Praha *RNDr. Daniel Kráľ , PhD.* a *Milan Straka*. V dnešním pokračování série článků o zajímavých úlohách z programátorských soutěží se budeme věnovat podrobněji první z těchto úloh. Její zadání uvedeme v původním znění, řešení ovšem rozebereme poněkud podrobněji včetně ukázkového programu v Pascalu. V závěru článku najdete i zadání navazující druhé úlohy s krátkým návodem k řešení.

Prádelna

Bořivoj se rozhodl, že začne podnikat, a to ve velkém. Jednou v noci, poté co upadl v koupelně, dostal skvělý nápad: otevře si prádelnu. Každý, kdo přijde, si bude moci za malý obnos půjčit pračku, pokud bude nějaká volná, a vypere si své prádlo. Ihned se tedy zeptal svých přátel, zda by chtěli jeho prádelnu navštěvovat, a zjistil, že zájem je opravdu velký. Brzy ani nevěděl, kolik praček bude vůbec potřebovat, aby se dostalo na všechny zákazníky. A proto se rozhodl obrátit se na vás s prosbou o pomoc.

Soutěžní úloha: Na vstupu dostanete počet zakázek, které Bořivoj na jeden konkrétní den obdržel. U každé zakázky víte čas příchodu zákazníka a dobu, na jakou si chce pronajmout jednu pračku. Požadavky zákazníků nejsou uvedeny v žádném konkrétním pořadí. Vaším úkolem je zjistit, kolik nejméně praček bude Bořivoj potřebovat, aby si každý zákazník mohl pronajmout pračku na celou požadovanou dobu od svého příchodu. Kromě minimálního počtu praček musíte pro Bořivoje vytvořit ještě seznam, podle kterého bude posílat zákazníky k volným pračkám.

Formát vstupu: První řádka textového souboru *pradelna.in* obsahuje jediné přirozené číslo $N \leq 1000$ – počet zákazníků. Dalších N řádek obsahuje informace o jednotlivých zákaznících: na i-té z těchto řádek je uveden čas T_i, kdy chce zákazník přijít, a doba T'_i, na kterou si chce pronajmout pračku. Můžete předpokládat, že T_i a T'_i jsou celá čísla od 1 do 1 000 000 000.

Formát výstupu: První řádka textového souboru *pradelna.out* obsahuje jediné číslo P – nejmenší možný počet praček, s nimiž může Bořivoj obsloužit všechny zákazníky. Dalších N řádek bude obsahovat N čísel a_1 až a_N, přičemž a_i je číslo pračky, kterou má použít i-tý zákazník. Předpokládejte, že pračky budou očíslovány od 1 do P.

Příklad:

pradelna.in	pradelna.out
4	3
1000 1000	2
1900 900	1
1500 700	3
2000 500	2

Řešení úlohy je založeno na metodě zvané počítačová simulace. Budeme sledovat, co se s postupujícím časem odehrává v prádelně. Zajímavé jsou pro nás okamžiky, kdy některý zákazník přichází do prádelny, a okamžiky, kdy někdo dokončil praní a z prádelny odchází. Těmto důležitým okamžikům, v nichž se nějak skokově mění stav simulovaného systému, se říká události. Pro N zákazníků bude tedy existovat celkem $2N$ událostí. Časy jednotlivých událostí lehce určíme přímo ze vstupních dat. Všechny události si musíme nejprve seřadit vzestupně podle jejich času, abychom je následně mohli procházet a zpracovávat chronologicky.

Při vlastní simulaci se postupně zvyšuje hodnota simulárního času, tj. nastavení pomyslných hodin v simulovaném systému. Simulární čas neroste spojitě jako ve skutečnosti, ale mění se skokem tak, že nabývá jenom hodnot času jednotlivých událostí. Vždy se zpracují všechny události, které nastanou v příslušném čase, a poté se simulární čas opět zvýší. Čas v simulovaném systému tedy postupně nabývá jen jistých diskrétních hodnot, a proto se tato metoda výpočtu označuje jako diskrétní počítačová simulace.

Vraťme se nyní zpět k řešené úloze. Předem nevíme, kolik praček budeme potřebovat mít v simulovaném systému (tj. v prádelně). Naším cílem ale je mít jich co nejméně. Začneme proto s „prázdnou prádelnou" bez praček a pračky do ní budeme postupně přidávat vždy, když to potřeba. Při každém příchodu zákazníka se podíváme, zda máme momentálně nějakou volnou pračku, a pokud ano, pošleme ho k ní. Je-li při příchodu zákazníka volných praček více, jistě mu můžeme přidělit jednu libovolnou z nich, neboť všechny jsou rovnocenné. Pokud volnou pračku nemáme, je nutné zvýšit počet praček v prádelně. Nastal totiž okamžik, kdy chce prát zároveň více zákazníku, než kolik jsme měli dosud praček. Přicházejícího zákazníka pochopitelně pošleme k této nové pračce. Při každém odchodu zákazníka si jen poznamenáme, že jemu přidělená pračka byla uvolněna. Pro efektivní realizaci tohoto postupu si musíme udržovat seznam všech volných praček a také evidenci, který zákazník byl nebo ještě je u které pračky.

Po zpracování všech událostí, tzn. po odchodu posledního zákazníka, budeme znát minimální nezbytný počet praček a budeme mít i zaznamenáno, u které pračky byl který zákazník. Přesně to jsou požadované výstupní údaje. Nalezený výsledek je jistě správný, neboť počet praček v prádelně jsme zvyšovali pouze v okamžiku, kdy to bylo objektivně zcela nezbytné (když všechny pračky byly obsazené a přicházel další zákazník).

Zbývá zamyslet se nad tím, jak budeme zpracovávat více událostí, které nastaly v témže okamžiku. Pokud ve stejném okamžiku některý zákazník končí praní a jiný přichází do prádelny, může nově příchozí zákazník dostat přidělenu právě uvolněnou pračku. Máme-li tedy více událostí stejným časem, je třeba zpracovat nejprve všechny odchody (aby se uvolnily pračky) a až poté všechny příchody zákazníků. Nejlepší je pamatovat na to již při řazení událostí podle času a v případě shodného času rovnou umístit odchody před příchody zákazníků.

Na závěr popisu algoritmu odhadneme ještě jeho časovou a paměťo-

vou složitost. Načtení vstupních dat a vytvoření událostí má jistě časovou složitost $O(N)$. Poté je třeba všech $2N$ událostí uspořádat. Seřazení K hodnot podle velikosti lze provést nejrychleji vhodným třídicím algoritmem s časovou složitostí $O(K \cdot \log K)$, použít můžeme například třídění haldou (*heapsort*). V našem případě $K = 2 \cdot N$ a $2N \cdot \log(2N) = 2N(\log 2 + logN) = O(N \log N)$, takže tato přípravná fáze řešení má časovou složitost $O(N \cdot \log N)$. Vlastní simulace představuje jeden sekvenční průchod všemi setříděnými událostmi, každou z nich zpracujeme v konstantním čase. Časová složitost simulace je tedy $O(N)$. Časová složitost celého algoritmu je dána složitostí nejpomalejší fáze a je proto rovna $O(N \cdot \log N)$. Pro uložení všech událostí potřebujeme pole velikosti $2N$ záznamů, jejich setřídění se provede na místě v tomto poli. Při vlastní simulaci pak potřebujeme pole velikosti N na uložení seznamu právě volných praček a další pole velikosti N na evidenci přiřazení praček jednotlivým zákazníkům. Celková paměťová složitost algoritmu je proto $O(N)$.

```
program Pradelna; {MO-P 54-I-1}

{jednoduchá diskrétní počítačová simulace}

const MaxN = 1000;                {maximální počet zákazníků}

type TUdalost = record
                  Cas: longint;                 {čas události}
                  Prichod: boolean;           {true = příchod,
                                               false = odchod}
                  Zakaznik: integer          {číslo zákazníka}
                end;
     TUdalosti = array[1..2*MaxN] of TUdalost;
                                        {typ pro pole událostí}
var Udalosti: TUdalosti;                        {pole událostí}
    N: integer;                                {počet zákazníků}
    P: integer;                                   {počet praček}
    Pracka: array[1..MaxN] of integer;
                     {přiřazení pračky jednotlivým zákazníkům}
    Volne: array[1..MaxN] of integer;
                      {seznam čísel volných praček = zásobník}
    V: integer;      {počet volných praček = vrchol zásobníku}
```

```
    I, T, X: integer;

procedure Seradit(var U: TUdalosti; Zac, Kon: integer);
  {Seřadí prvky pole U typu TUdalosti v rostoucím pořadí
   podle položky Cas.
   V případě výskytu více záznamů se stejnou hodnotou
   položky Cas řadí podle hodnoty položky Prichod
   (nejprve záznamy s hodnotou false, potom s true).
   Tříděné prvky jsou v poli U umístěny v rozmezí indexů
   od Zac do Kon.
   Použitý algoritmus: Quicksort}

  var X: TUdalost;           {hodnota pro rozdělení na úseky}
      Q: TUdalost;                  {pro výměnu prvků v poli}
      I, J: integer;       {posouvané pracovní indexy v poli}

  function Mensi(A, B: TUdalost): boolean;
    {porovnává pořadí dvou událostí: true <=> A<B }
    begin
    Mensi := (A.Cas < B.Cas) or
             ((A.Cas = B.Cas) and not A.Prichod
                              and B.Prichod)
    end;

  function Vetsi(A, B: TUdalost): boolean;
             {porovnává pořadí dvou událostí: true <=> A>B }
    begin
    Vetsi := (A.Cas > B.Cas) or
             ((A.Cas = B.Cas) and A.Prichod
                              and not B.Prichod)
    end;

  begin {Seradit}
  I := Zac;
  J := Kon;
  X := U[(Zac+Kon) div 2];
  {za hodnotu X vezmeme prostřední prvek ve zkoumaném úseku}
  repeat
```

```
    while Mensi(U[I], X) do inc(I);      {U[I] < X}
    while Vetsi(U[J], X) do dec(J);      {U[J] > X}
    if I < J then            {vyměnit prvky s indexy I a J}
      begin
      Q := U[I]; U[I] := U[J]; U[J] := Q;
      inc(I); dec(J);         {posun indexů na další prvky}
      end
    else if I = J then
          {indexy I a J se sešly, oba ukazují na hodnotu X}
      begin
      inc(I); dec(J)
      {posun indexů na další prvky - kvůli ukončení cyklu}
      end
  until I > J;
  {úsek <Zac,Kon> je rozdělen na úseky <Zac,J> a <I,Kon>,
  které zpracujeme rekurzivním voláním procedury:}
  if Zac < J then Seradit(U, Zac, J);
  if I < Kon then Seradit(U, I, Kon);
  end; {Seradit}

begin
{Načtení vstupních dat a vytvoření událostí:}
readln(N);
for I:=1 to N do
  begin
  read(Udalosti[2*I-1].Cas);
  Udalosti[2*I-1].Prichod :=true;
  Udalosti[2*I-1].Zakaznik := I;
  readln(T);
  Udalosti[2*I].Cas :=Udalosti[2*I-1].Cas + T;
  Udalosti[2*I].Prichod :=false;
  Udalosti[2*I].Zakaznik := I;
  end;

{Seřazení událostí podle rostoucího času,
 v případě shody času dříve odchody:}
Seradit(Udalosti, 1, 2*N);
```

```
{Vlastní simulace dění v prádelně:}
P := 0;                                    {začínáme bez praček}
V := 0;                                  {žádná pračka není volná}
for I:=1 to 2*N do                      {zpracování I-té události}
  if Udalosti[I].Prichod then
    begin
    if V=0 then                                {není volná pračka}
      begin
      inc(P);                                {přidáme novou pračku}
      X := P                               {přidělíme ji zákazníkovi}
      end
    else                                          {je volná pračka}
      begin
      X := Volne[V];                 {obsadíme jednu volnou pračku}
      dec(V)                         {snížíme počet volných praček}
      end;
    Pracka[Udalosti[I].Zakaznik] := X
    end
  else                              {událost je odchod zákazníka}
    begin
    inc(V);                                    {uvolníme jeho pračku}
    Volne[V] := Pracka[Udalosti[I].Zakaznik]
    end;

{Výpis výsledků:}
writeln(P);
for I:=1 to N do
  writeln(Pracka[I])
end.
```

K seřazení prvků v poli *Udalosti* lze použít libovolný standardní třídicí algoritmus (procedura *Seradit*). V uvedené programové ukázce jsme pro jednoduchost použili algoritmus *quicksort*, který nám dává průměrnou časovou složitost $O(N \log N)$. Chceme-li mít zajištěnu časovou složitost $O(N \log N)$ i v nejhorším případě, můžeme místo něj použít například *heapsort*. Oba algoritmy jsou všeobecně známé a jejich popis najdete v mnoha učebnicích programování.

V krajském kole soutěže se řešitelé setkali s modifikací této úlohy. Její zadání opět uvedeme v původním znění:

Prádelní salón

Z Bořivoje se stal díky vaší pomoci úspěšný podnikatel a jeho klientela zahrnuje i bohatší a malichernější zákazníky. Pokud totiž nějaký zákazník uvidí dva různé zákazníky používat stejnou pračku, nebude už tuto prádelnu dále navštěvovat: „No považte, přece nelze prát prádlo s lidmi, kteří nemají na to, aby si zaplatili pračku sami pro sebe!“

Soutěžní úloha: Na vstupu dostanete $N \leq 10000$ – počet zákazníků, kteří navštíví Bořivojovu prádelnu během jednoho dne. U každého zákazníka je zadán čas jeho příchodu a doba, na jakou si chce pronajmout pračku (obojí jsou celá čísla mezi 1 a 1 000 000 000). Požadavky zákazníků nejsou uvedeny v žádném konkrétním pořadí.

Vaším úkolem je zjistit, kolik nejméně praček Bořivoj potřebuje, aby všichni jeho zákazníci byli zcela spokojeni. Zákazník bude spokojen, pokud si bude moci pronajmout pračku od okamžiku příchodu na dobu, kterou požaduje (je samozřejmé, že jednu pračku nemohou používat dva různí zákazníci současně), a navíc během doby, kdy bude prát, nebude žádnou pračku využívat více zákazníků po sobě.

Kromě určení minimálního počtu praček musíte pro Bořivoje vytvořit ještě seznam, podle kterého bude posílat zákazníky k volným pračkám.

Příklad: Pro 5 zákazníků, jejichž příchody a doby praní jsou

```
1000 1000
3000 2000
4500 500
1500 500
2000 2000
```

jsou potřeba alespoň 3 pračky a přiřazení praček zákazníkům například takové:

```
zákazník 1 bude u pračky 2
zákazník 2 bude u pračky 3
zákazník 3 bude u pračky 1
zákazník 4 bude u pračky 3
zákazník 5 bude u pračky 2
```

Všimněte si, že zákazníci 3 a 5 nemohou dostat stejnou pračku, protože by je viděl zákazník 2 pracovat u stejné pračky.

Řešení této varianty úlohy je velmi podobné postupu, s nímž jsme se právě seznámili. Z hlediska algoritmu jediný rozdíl spočívá v tom, že když nějaký zákazník Z odchází, uvolní se jeho pračka pro okamžité další použití jedině v případě, že v prádelně zrovna nikdo jiný není. Pokud jsou v prádelně nějací zákazníci, zůstane uvolněná pračka „zablokována“ a nesmí se použít pro nikoho dalšího tak dlouho, dokud neodejde poslední ze zákazníků přítomných v prádelně při odchodu zákazníka Z. Tento požadavek lze realizovat různými způsoby. Zkuste si s pomocí našeho návodu dokončit řešení sami. Zvolíte-li vhodný postup, časová složitost algoritmu se oproti složitosti řešení úlohy z domácího kola nezvýší. Pokud by vás zajímalo, jak elegantně a efektivně vyřešili tento problém autoři soutěžních úloh, jejich vzorové řešení najdete na Internetu na webových stránkách Matematické olympiády – kategorie P http://mo.mff.cuni.cz/ v archívu úloh (úloha 54-II-1).

Riešenie konštrukčných úloh na stredovú súmernosť pomocou Cabri geometrie

STANISLAV LUKÁČ

Prírodovedecká fakulta UPJŠ, Košice, SLOVENSKO

Úvod

Týmto článkom by sme radi nadviazali na článok [1], v ktorom boli opísané možnosti využitia programu Cabri geometria v úvodnej etape výučby osovej súmernosti. Uviedli sme v ňom systém úloh zameraný na skúmanie vlastností osovej súmernosti a osovo súmerných útvarov. Dôležitým vzdelávacím cieľom vyučovania časti geometrie zameranej na zhodné zobrazenia je aplikovanie osvojených poznatkov na riešenie konštrukčných úloh. Pri ich riešení musia študenti hľadať vzťahy medzi neznámymi prvkami a na základe nich zobrazovať útvary vyhovujúce zadaniu úlohy. Tieto činnosti

vyžadujú uplatňovanie logického a analytického myslenia, čo niektorým študentom môže spôsobovať značné problémy. Z uvedených dôvodov považujú študenti konštrukčné úlohy za veľmi náročné a ich riešenie nepatrí medzi ich obľúbené činnosti na hodinách matematiky. Vlastnosti a rôznorodé prostriedky dynamického geometrického systému Cabri geometria možno využiť aj pri riešení konštrukčných úloh.

V článku sa budeme tentokrát zaoberať konštrukčnými úlohami na stredovú súmernosť. Z bohatej ponuky príkazov a nástrojov programu Cabri geometria sme sa snažili vybrať a opísať tie, ktoré poskytujú možnosti pre hľadanie vzťahov medzi neznámymi prvkami konštrukcie, skúmanie vlastností konštrukcií a pre určovanie podmienok riešiteľnosti úloh. Cabri geometria a iné programy tohto typu prinášajú do vyučovania matematiky nové technológie vlastné dynamickým konštrukciám, ktoré umožňujú rôzne spôsoby ich využitia nielen študentom pri tvorbe a skúmaní konštrukcií, ale aj učiteľom pri príprave a realizácii výučby. Pri tvorbe dynamických konštrukcií sa pomocou príkazov programu využívajú prvky daných objektov (tzv. voľné prvky) pre konštrukciu na nich naviazaných objektov (tzv. viazané prvky). Aby viazané prvky konštrukcie mali zodpovedajúcu vlastnosť (napr. kružnica prechádzala daným bodom alebo mala určený polomer), je potrebné vybrať z ponuky správne príkazy a presne vyberať ukazovateľom na nákresni objekty podľa pokynov programu, ktoré sa zobrazujú v blízkosti ukazovateľa. Pri zmene polohy alebo iných vlastností voľných prvkov konštrukcie program automaticky prekreslí na nich odvodené objekty podľa ich vzťahu k meneným voľným prvkom. Takýmto spôsobom môžeme na nákresni postupne zobrazovať rozličné konkrétne prípady konštrukcie v závislosti od zadaných prvkov. Uvedené možnosti programu sa budeme snažiť ilustrovať pri riešení konštrukčných úloh založenom na využívaní už osvojených základných vlastností stredovej súmernosti a stredovo súmerných útvarov. Vybrané konštrukčné úlohy sme rozdelili do štyroch skupín, ktoré reprezentujú rôzne typy konštrukčných úloh nachádzajúcich sa v učebniciach matematiky.

Konštrukčné úlohy

Ako už bolo uvedené z hľadiska vyučovacích cieľov tvorí riešenie konštrukčných úloh dôležitú súčasť vyučovania matematiky. Častou chybou pri riešení konštrukčných úloh je, že študenti po neúplnom alebo nesprávnom rozbore riešenia úlohy pristupujú ku konštrukcii, v ktorej sa potom snažia špekulatívnymi spôsobmi dokresliť hľadané útvary. Ak je konštruk-

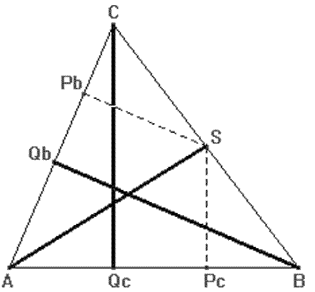
cia zostrojovaná na papieri, kde sa ešte môžu pridružiť aj chyby vyplývajúce z nepresnosti rysovania, môže byť niekedy problematické rýchlo objaviť chyby v postupe konštrukcie. Pri tvorbe konštrukcií v programe Cabri geometria by sa chyby tohto druhu mali rýchlo prejaviť po zmenách voľných prvkov konštrukcie vedúcich k zobrazeniu rôznych konkrétnych prípadov. Táto vlastnosť je v súlade aj s vyučovacím cieľom riešenia konštrukčných úloh, spočívajúcom v hľadaní všeobecného postupu riešenia konštrukčnej úlohy. Ďalším prostriedkom pre skúmanie konštrukcie a hľadanie prípadných chýb je príkaz na prehratie konštrukcie, ktorý umožňuje postupne zobrazovať jednotlivé kroky vedúce k vytvoreniu výslednej konštrukcie na nákresni.

1. Konštrukcia dvojice hľadaných bodov na základe známeho stredu súmernosti

Do prvej skupiny sme zaradili konštrukčné úlohy zamerané najčastejšie na konštrukcie trojuholníkov, ktorých dva neznáme vrcholy majú zadanú vlastnosť určujúcu útvar v rovine, pričom je zadaný stred súmernosti hľadaných bodov. Ako príklad možno uviesť konštrukciu trojuholníka s danými ťažnicami alebo úlohu: Daná je úsečka AS s veľkosťou 5 cm. Zostrojte trojuholník ABC s ťažnicou AS, ak veľkosť uhla ACB je $30°$ a dĺžka strany AB je 6 cm.

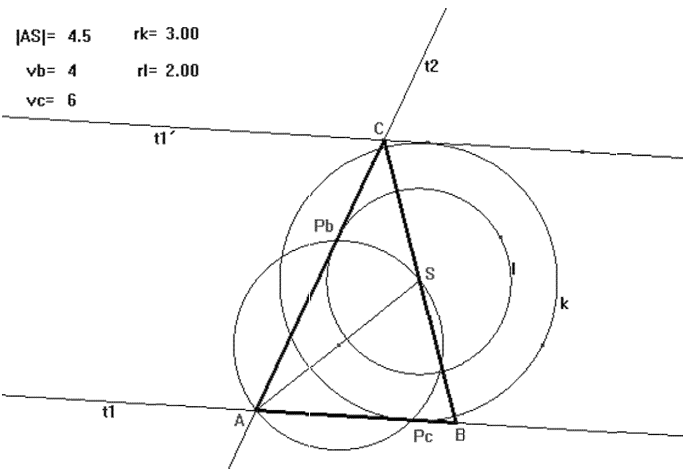
Podrobnejšie sa budeme venovať riešeniu nasledovnej úlohy: Daná je úsečka AS s veľkosťou 4,5 cm. Zostrojte všetky trojuholníky ABC s ťažnicou AS tak, aby $v_b = 4$ cm a $v_c = 6$ cm. Na obrázku 1 predstavujúcom zjednodušený náčrt trojuholníka ABC sú zvýraznené zadané prvky a zobrazené kolmé priemety bodu S na strany AC a AB. Trojuholníky BP_cS a BQ_cC sú podľa vety uu podobné s pomerom podobnosti 2. Dĺžka úsečky SP_c je potom rovná polovici dĺžky výšky v_c (3 cm). Keďže uhol SP_cB je pravý, priamka AB predstavuje dotyčnicu ku kružnici $k(S, SP_c)$. Podobnú úvahu využijeme pre trojuholníky CQ_bB a CP_bS. Priamka AC je potom dotyčnicou ku kružnici $l(S, SP_b)$. Body B a C sú súmerne združené v stredovej súmernosti určenej bodom S. Preto priesečník priamky predstavujúcej obraz jednej dotyčnice v stredovej súmernosti podľa bodu S a druhej dotyčnice určuje ďalší vrchol trojuholníka ABC.

Konštrukcia jedného trojuholníka ABC s danými vlastnosťami je zobrazená na obrázku 2. Z bodu A možno zostrojiť dve dotyčnice ku kružnici k, ktoré sú osovo súmerné podľa priamky AS. Podobne možno zostrojiť aj dve dotyčnice kružnice l prechádzajúce bodom A. Ak by sme využili dve



Obr. 1

nové dotyčnice nezobrazené na obrázku 2 pre konštrukciu hľadaného trojuholníka, získali by sme druhé riešenie úlohy predstavujúce trojuholník osovo súmerný s trojuholníkom ABC podľa priamky AS. Ďalšie dva osovo súmerné trojuholníky podľa priamky AS vyhovujúce zadaniu úlohy možno zostrojiť využitím zvyšných dvoch dvojíc dotyčníc kružníc k a l. Získali by sme tak všetky štyri riešenia úlohy. Aby však úloha pre dané vstupné údaje vôbec mala riešenie, musí Talesova (Thaletova) kružnica zostrojená nad priemerom AS pretnúť kružnice k aj l. To je splnené vtedy, ak súčasne platia nasledujúce podmienky: $r_k = v_c/2 \leqq |AS|$ a $r_l = v_b/2 \leqq |AS|$.



Obr. 2

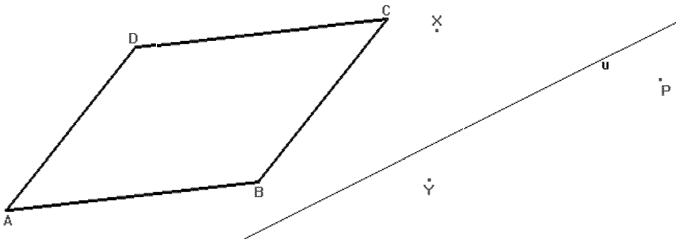
Pri riešení konštrukčných úloh sa môže stať, že študent založí konštrukciu hľadaného útvaru na nesprávnom predpoklade. Môže to byť napríklad vzťah platný medzi prvkami konštrukcie len v určitých špeciálnych prípadoch. Pomocou programu Cabri geometria možno v etape hľadania riešenia preveriť niektoré kroky riešenia na viacerých konkrétnych prípadoch, čo umožňuje rýchlo objaviť mylné úvahy. V určitom zmysle je to analogický postup ako pri testovaní algoritmov v informatike. Simuláciu algoritmu na vhodne zvolených sadách vstupných údajov síce nepovažujeme za dôkaz správnosti algoritmu, ale s veľkou pravdepodobnosťou nám pomôže odhaliť chyby v nesprávnom algoritme. Učiteľ môže rozvíjať schopnosti žiakov odhaľovať chyby v konštrukciách tým, že pripraví pre študentov výkresy s chybnými konštrukciami a nechá študentov nachádzať a opravovať tieto chyby.

2. Dokreslenie stredovo súmerného útvaru

Úlohy tohto typu sú zamerané na využitie vlastností stredovo súmerných útvarov na ich zostrojenie na základe daných prvkov. Ako príklad uvádzame nasledovnú úlohu: Dané sú nekolineárne body A, B, S. Zostrojte štvorec $MNPQ$ so stredom S tak, aby bod A ležal na priamke MN a bod B na priamke PQ. Keďže priamky MN a PQ sú navzájom stredovo súmerné podľa stredu S (priesečník uhlopriečok štvorca), obraz bodu B v stredovej súmernosti určenej bodom S a bod A určujú priamku, na ktorej leží strana štvorca MN.

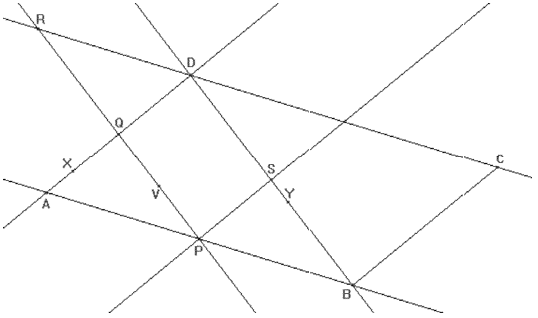
Ak máme vytvorenú konštrukciu hľadaného útvaru, možno pomocou schránky skopírovať útvar a využiť príkazy programu Cabri geometria na vytvorenie výkresu obsahujúceho dané prvky a predlohu hľadaného útvaru, ktorú môžu študenti využiť pre kontrolu svojho riešenia. Ukážeme to na tomto príklade: Z rovnobežníka $ABCD$ zostali len priamka u, na ktorej leží uhlopriečka AC, bod X ležiaci na priamke CD, bod Y ležiaci na priamke AB a stred P strany BC. Dokreslite rovnobežník $ABCD$. Opisovanú situáciu zobrazuje obrázok 3.

Stred úsečky XY a bod P určujú priamku, na ktorej leží stred rovnobežníka S. V priesečníku tejto priamky s priamkou u leží bod S, podľa ktorého je rovnobežník $ABCD$ stredovo súmerný. Rovnobežky s priamkou PS vedené cez body X, Y obsahujú strany AB a CD. V priesečníku týchto priamok s priamkou u dostávame vrcholy A, C. Zvyšné vrcholy rovnobežníka možno ľahko zostrojiť využitím stredovej súmernosti určenej stredmi P a S.



Obr. 3

Príkaz na prehratie konštrukcie umožňuje pri zobrazovaní jednotlivých krokov konštrukcie zachovať priebežný stav určený od začiatku po ľubovoľný krok konštrukcie a vymazať všetky neskôr zostrojené útvary. Tento postup možno vhodne využiť pri preformulovávaní úloh v etape prípravy na vyučovaciu hodinu. Modifikáciu zadania úlohy budeme ilustrovať na nasledovnom príklade: Dané sú nekolineárne body X, P, V. Zostrojte rovnobežník $ABCD$, ak bod X leží na priamke určenej stranou AD, bod P je stred strany AB a bod V je stred úsečky AS, kde S je priesečník uhlopriečok rovnobežníka $ABCD$. Po zakreslení daných bodov zostrojíme bod Q ako obraz bodu P v stredovej súmernosti určenej bodom V a bod R ako obraz bodu P v stredovej súmernosti určenej bodom Q. Bod Q je stredom strany AD a bod R leží na priamke určenej stranou CD. Na rovnobežke s priamkou QX vedenej cez bod P leží stred rovnobežníka S.



Obr. 4

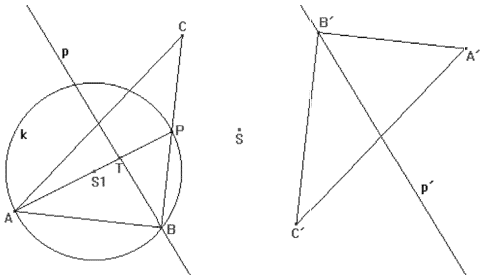
Na zostrojenie bodu S potrebujeme ďalšiu informáciu. Mohli by sme napríklad do zadania doplniť bod Y ležiaci na priamke určenej stranou CD. Potom by už na priamke RY ležali body C, D a na priamke rovnobežnej s RY vedenej cez bod Q by ležal bod S, takže by sme ho už vedeli zostrojiť. Nech však štvrtý daný bod Y leží na priamke určenej uhlopriečkou BD. Podľa vety sus sú trojuholníky PAQ a BAD podobné, preto rovnobežka s priamkou PQ vedená cez bod Y obsahuje bod S. Po zostrojení bodu S a vrcholu D ľahko dokreslíme ďalšie vrcholy rovnobežníka $ABCD$. Výsledná konštrukcia je zobrazená na obrázku 4.

3. Dokreslenie navzájom stredovo súmerných útvarov

Program Cabri geometria poskytuje príkazy na jednoduché a rýchle zostrojovanie obrazov útvarov v základných typoch zhodných zobrazení. Tieto možnosti možno vhodne využiť pri riešení konštrukčných úloh zameraných na zostrojovanie dvojice navzájom stredovo súmerných útvarov. Ako príklad uveďme nasledovnú úlohu: Dané sú body A, C, B', S'_o, kde S'_o je stred opísanej kružnice trojuholníku $A'B'C'$. Zostrojte trojuholníky ABC a $A'B'C'$, ak sú navzájom stredovo súmerné. Stačí si uvedomiť, že stred kružnice opísanej trojuholníku ABC leží na osi úsečky AC a polomery opísaných kružníc stredovo súmerných trojuholníkov sú rovnaké.

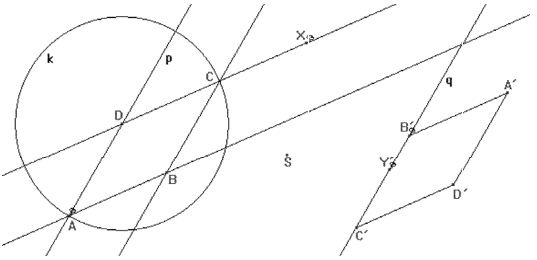
Konštrukčne zaujímavejšia je nasledujúca úloha: Daná je ťažnica AP trojuholníka ABC s pravým uhlom pri vrchole B, bod B' a priamka p' prechádzajúca bodom B', na ktorej leží ťažnica t'_b trojuholníka $A'B'C'$. Zostrojte trojuholníky ABC a $A'B'C'$, ak sú navzájom stredovo súmerné. Ak zostrojíme ťažisko trojuholníka ABC, potom vrchol B leží na priamke p rovnobežnej s p' vedenej cez ťažisko T. Bod T bol na výkrese zostrojený využitím pomocnej konštrukcie rozdeľujúcej úsečku AP v pomere 2:1, ktorú sme pre väčšiu prehľadnosť konštrukcie ukryli. Keďže uhol ABP je pravý, vrchol B leží v priesečníku priamky p a Talesovej kružnice zostrojenej nad priemerom AP. Stred súmernosti hľadaných trojuholníkov je v strede úsečky BB'. Na obrázku 5 je zostrojená len jedna dvojica hľadaných trojuholníkov ABC a $A'B'C'$.

Ako posledný v tejto skupine uvedieme príklad zo zbierky úloh [3] zameraný na konštrukciu stredovo súmerných kosoštvorcov: Dané sú body A, B', X, Y', kde X patrí priamke CD a Y' patrí priamke $B'C'$. Zostrojte kosoštvorce $ABCD$, $A'B'C'D'$, ak viete, že sú navzájom stredovo súmerné. Po zohľadnení orientácie vrcholov stredovo súmerného kosoštvorca a faktu, že v stredovej súmernosti sa zachováva rovnobežnosť, možno zostrojiť priamku p rovnobežnú s priamkou $B'Y'$ prechádzajúcu bodom A. Na



Obr. 5

tejto priamke musí ležať vrchol D. Po ďalšom hľadaní vzťahov, ktoré by mohli viesť ku konštrukcii niektorého z vrcholov alebo stredu súmernosti, sa môže ponúknuť hypotéza, že dvojíc navzájom stredovo súmerných kosoštvorcov vyhovujúcich zadaniu úlohy by mohlo existovať nekonečne veľa. Preto upevnime na výkrese dané body a zvoľme na priamke p ľubovoľne vrchol D. Teraz už možno ľahko zostrojiť kosoštvorec $ABCD$ a následne po určení stredu súmernosti S aj kosoštvorec $A'B'C'D'$. Po posúvaní bodu D po priamke p sa možno presvedčiť, že získavame rôzne dvojice navzájom stredovo súmerných kosoštvorcov vyhovujúcich zadaniu úlohy. Na obrázku 6 je zobrazená jedna dvojica navzájom stredovo súmerných kosoštvorcov.

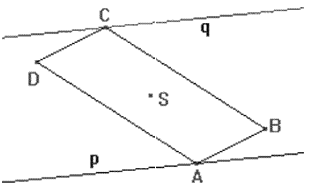


Obr. 6

Pri riešení úloh tohto typu zistenie možnosti existencie nekonečného počtu riešení nie je triviálne a využitie programu Cabri geometria môže značne uľahčiť konštrukciu hľadaných útvarov nielen študentom ale aj učiteľom pri príprave výučby.

4. Množiny všetkých bodov danej vlastnosti

Ako posledný typ sme vybrali úlohy na vyšetrovanie množín všetkých bodov danej vlastnosti. Program Cabri geometria disponuje príkazmi na zostrojovanie množín bodov danej vlastnosti a na vykresľovanie priebežných polôh bodov pri ich pohybe po nákresni (tzv. stopa útvaru). Pomocou týchto nástrojov môžu študenti nielen zobraziť hľadané útvary, ale aj využívať dynamiku konštrukcií pri hľadaní špeciálnych prípadov predstavujúcich určité obmedzenia alebo pri objavovaní vzťahov medzi hľadaným útvarom a danými prvkami. Začnime týmto príkladom: Daná je priamka p obsahujúca bod A a rôzne body S, B neležiace na priamke p. Vyšetrite množinu všetkých vrcholov C rovnobežníka $ABCD$ so stredom S, pre ľubovoľné body A. Po zostrojení bodu C prípadne celého rovnobežníka pomocou stredovej súmernosti využijeme niektorý z vyššie uvedených príkazov na vykreslenie ďalších vrcholov C z hľadaného útvaru. Získame body z priamky q rovnobežnej s priamkou p predstavujúcej obraz priamky p v stredovej súmernosti určenej bodom S. Riešenie úlohy je zobrazené na obrázku 7. Pri pohybe bodu A po priamke p však možno nájsť špeciálny prípad, kedy bod A leží na priamke BS, ktorá obsahuje aj vrcholy C, D. V tomto prípade netvoria body A, B, C, D rovnobežník a preto riešením úlohy je priamka q okrem jej priesečníka s priamkou BS. Tu nás však zaiste napadne ďalší špeciálny prípad, keď body B, S ležia na priamke rovnobežnej s priamkou p. V takomto prípade priamka BS nepretne priamku p a preto riešením úlohy bude celá priamka q.

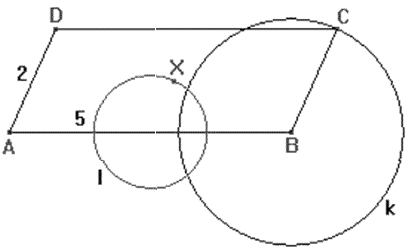


Obr. 7

Uvedenú úlohu možno modifikovať viacerými spôsobmi. Jedným variantom môže byť nasledovná úloha: Daná je kružnica $k(S, r)$ a bod A ležiaci zvonku kružnice k. Nech bod P je ľubovoľný bod kružnice k. Vyšetrite množinu všetkých bodov Y, kde $SAXY$ je rovnobežník so stredom P. Pre ľubovoľný rovnobežník $SAXY$ je bod P stredom úsečky AY. Riešením

úlohy bude preto v tomto prípade kružnica l predstavujúca obraz kružnice k v rovnoľahlosti so stredom v bode A a koeficientom 2 okrem dvoch bodov ležiacich v priesečníkoch priamky AS a kružnice l. V týchto špeciálnych prípadoch leží bod P na priamke AS a namiesto rovnobežníka dostávame úsečku.

Od priamky sme sa postupne dostali ku kružnici, objavenie ktorej kreslením na papieri môže byť pre niektorých študentov náročné. Absencia predstavivosti môže spôsobiť, že študenti nevedia nájsť hľadaný útvar a nemôžu potom pristúpiť ani k ďalšej etape riešenia úlohy zameranej na zdôvodnenie vyslovenej hypotézy a zovšeobecnenia riešenia na základe logických úvah. Ako je uvedené aj v publikácii [4], program Cabri geometria môže výrazne pomôcť študentom najmä v úvodnej etape riešenia problémov tohto typu, ak chýba predstava výsledného útvaru a hypotézy študentov nie sú podložené logickými argumentmi, ale predstavujú skôr tipovanie.



Obr. 8

Na ilustráciu možností využitia programu Cabri geometria v úvodnej etape riešenia problému by mohol slúžiť nasledujúci príklad: Dané sú body A, B vo vzdialenosti 5 cm. Vyšetrite množinu všetkých bodov X, pre ktoré štvoruholník zložený z vrcholov A, B a ich obrazov v stredovej súmernosti podľa stredu X má obvod 14 cm. Štvoruholník s vrcholmi A, B a ich obrazmi v stredovej súmernosti podľa bodu X bude rovnobežníkom. Preto strana BC bude mať dĺžku 2 cm. Zostrojíme úsečku AB s veľkosťou 5 cm a kružnicu $k(B, 2\text{ cm})$. Bod C zvolíme ľubovoľne na kružnici k. Potom zostrojíme bod X ako stred AC a vrchol D ako obraz bodu B v stredovej súmernosti určenej stredom X. Na obrázku 8 je zostrojený hľadaný útvar, ktorým je kružnica l so stredom v strede úsečky AB a polomerom 1 cm. Predstavuje obraz kružnice k v rovnoľahlosti so stredom v bode A

a koeficientom 0,5. Znova však musíme z kružnice l vylúčiť jej priesečníky s úsečkou AB. Sú to prípady, kedy bod C leží na priamke AB a body A, B, C, D netvoria štvoruholník.

Záver

V článku sme sa snažili poukázať na niektoré možnosti využitia programu Cabri geometria pri riešení konštrukčných úloh. Rôznorodé prostriedky programu môžu byť využité pre podporu všetkých etáp vyučovacieho procesu počnúc od prípravy výučby až po kontrolu a zovšeobecňovanie študentmi vytvorených riešení. Na druhej strane treba poznamenať, že dynamické geometrické programy predstavujú otvorené systémy, od ktorých nemožno očakávať, že naučia študentov riešiť rôzne typy úloh z geometrie. Jedná sa najmä o úlohy, riešenie ktorých vyžaduje určitý stupeň rozvoja abstraktného myslenia, medzi ktoré určite patrí aj väčšina konštrukčných úloh. Dynamické geometrické systémy plnia vo vyučovacom procese úlohu didaktickej pomôcky, ktorá môže byť v rukách šikovného učiteľa veľmi výkonná a môže mať dôležitú úlohu pri zvyšovaní efektívnosti výučby. Prínosom pre vyučovanie matematiky môže byť aj skutočnosť, že po niekoľkých vyučovacích hodinách potrebných na zvládnutie základov tvorby dynamických konštrukcií môžu tieto programy ponúknuť jednoduchšie a výkonnejšie prostriedky ako sú klasické rysovacie nástroje a vrátiť tak študentom radosť z rysovania a učenia sa geometrie.

Literatúra

[1] *Engel, R. – Lukáč, S.:* Skúmanie osovej súmernosti pomocou Cabri geometrie. MFI, ročník 14, čísla 5, 6.

[2] *Hecht, T. – Černek, P.:* Matematika pre 2. ročník gymnázií a SOŠ, Geometrické zobrazenia. Orbis Pictus Istropolitana, Bratislava, 1997.

[3] *Hecht, T. a kol.:* Matematika pre 2. ročník gymnázií a SOŠ, Zbierka úloh. Orbis Pictus Istropolitana, Bratislava 2003.

[4] *Seibert, J. – Slabý, A. – Trojovský, P.:* Cabri geometrie a formulace hypotéz o množinách bodů dané vlastnosti. MFI, ročník 11, číslo 7.

[5] *Smida, J. – Šedivý, J.:* Zbierka úloh z matematiky pre 1. ročník gymnázia. SPN Bratislava, 1985.

[6] *Vaníček, J.:* Geometrie na počítači: mřížové body. MFI, ročník 11, číslo 1.

[7] *Vrba, A.:* Oživlá geometrie. MFI, ročník 10, čísla 2, 3.