
MO-P 2000

Soutěž dětí a mládeže v programování
kategorie starší žáci
obvodní kolo — obvod Praha 1

Zadání a řešení úloh, výsledková listina

Příklad 1

Zadání:

Máme pět ostrovů, na každém z nich je letiště, a různé letecké spoje mezi jednotlivými ostrovy. Letecký spoj je vždy pouze mezi dvěma ostrovy. Turista chce zjistit, zda se může dostat z jednoho ostrova na některý jiný ostrov.

Napište program, který se na začátku zeptá, mezi kterými ostrovy létá letecký spoj (např. spoj 1-2, spoj 1-3, atd. – na spoj mezi jedním a tím samým ostrovem se neptejte). Dále se program zeptá na jména ostrovů, mezi kterými chce turista cestovat. Program pak vypíše, zda mezi ostrovy cestovat lze nebo nelze. Ostrovy označte 1, 2, 3, 4, 5.

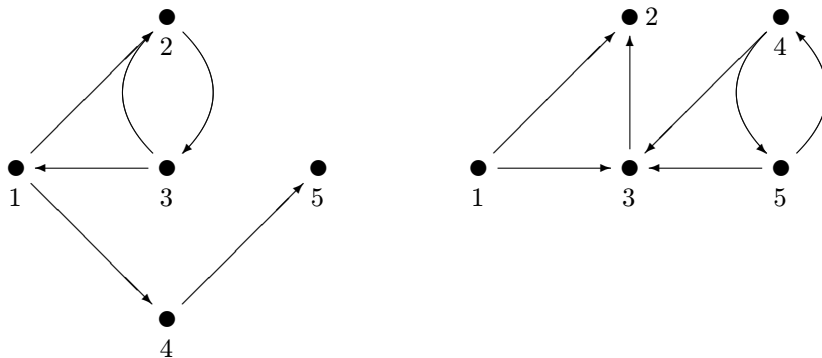
Řešení:

Jedná se o typickou úlohu na tzv. hledání cesty v grafu. Graf si můžeme představit jako množinu bodů, vrcholů v našem případě jsou to jednotlivé ostrovy), přičemž některé jsou navzájem propojeny (spojnice jsou v našem případě představovány leteckými spoji).

Tento problém se dá řešit několika způsoby, uveďme si jeden z nich, který se vyznačuje jednoduchostí a zároveň velkou rychlostí výsledného programu.

Ještě než začneme se samotným řešením, je dobré uvědomit si následující fakt: Pokud se lze z ostrova A dostat na ostrov B , potom tato správná cesta musí vést přes některý z ostrovů, na něž vede z ostrova A přímá letecká linka (a naopak, v případě, že se z žádného z těchto ostrovů nelze na ostrov B dostat, potom se nelze na B dostat ani ze samotného ostrova A). Problém lze tedy

řešit rekurzivně: z ostrova A se lze dostat na ostrov B , pokud se lze na ostrov B dostat z některého z ostrovů C_1, \dots, C_n , kde C_1, \dots, C_n jsou ostrovy, na které existuje z A přímé spojení. Předchozí odstavec možná vypadá zmateně, ale když si to promyslíte a podíváte se na obrázek 1, bude vám tato myšlenka jasná.



Obrázek 1: (vlevo:) Cesta z 1 do 5 vede přes 4, na nějž existuje z 1 přímé spojení, (vpravo:) cesta z 1 do 5 nevede, pokud z žádného z ostrovů, na něj je z 1 přímé spojení, nevede do 5 cesta

Funkce `ZjistiSpoj`, která určí, jestli z daného ostrova (určeného parametrem funkce) lze cestovat do cíle, může být tedy velice jednoduchá:

V první řadě otestuje, zda parametr není cílem. V případě, že ano, jednoduše vrátí hodnotu 1 (případně `true`) signalizující, že cesta byla nalezena. Pokud parametr není cílem, zavolá funkce sama sebe, postupně pro jednotlivé ostrovy, na něj existuje přímý spoj z právě zkoumaného ostrova. V případě, že alespoň pro jeden z těchto ostrovů bude vráceno 1 (`true`), vrátí funkce rovněž 1 (`true`). V případě, že pro všechny ostrovy bude návratová hodnota 0 (`false`), vrátí funkce hodnotu 0 (`false`), signalizující, že přes tento ostrov se do cíle dostat nelze.

Napsat takovou funkci je skutečně hračka, objevuje se však háček s jednoduchým jménem: zacyklení. Představme si situaci, kdy existuje spojení mezi ostrovy 1 a 2, dále spojení (1,3) a ještě spojení (2,1). Tyto tři spoje zadáme programu a zeptáme se, jestli je možné dostat se z ostrova 1 na ostrov 3. Tato cesta sice existuje (jedná se dokonce o přímý spoj), ale program na to nepříjde. Proč? Podle zásad našeho algoritmu zavolá nejdřív `ZjistiSpoj(1)`. Funkce `ZjistiSpoj` nejdřív otestuje, zda není v cíli (to není, $1 \neq 3$) a potom začne volat

sebe sama pro další ostrovy, do nichž existuje z 1 přímý spoj. Zjistí, že prvním takovým ostrovem je 2. Zavolá tedy `ZjistiSpoj(2)`. Ten opět provede neúspěšný test na cíl ($2 < 3$) a začne zkoumat bezprostřední sousedy. Jediným bezprostředním sousedem je ostrov 1. Následuje tedy volání `ZjistiSpoj(1)`, a dále již stále dokola volání `ZjistiSpoj(2)`, `ZjistiSpoj(1)`, `ZjistiSpoj(2)`,...

V čem spočívá problém: kdyby program uměl poznat, že z ostrova 2 se na 1 nemá cenu vracet, protože už tam jednou byl, místo dalšího volání `ZjistiSpoj(1)` by funkce `ZjistiSpoj(2)` vrátila 0 (false), neboť nenalezla cestu do cíle. Vykonávání programu by se vrátilo do funkce `ZjistiSpoj(1)`, kde by se zjistilo další letiště s přímým spojením (v našem případě 3), zavolala by se funkce `ZjistiSpoj(3)`, která by provedla tentokrát úspěšný test na cílový ostrov ($3=3$) a vrátila by 1 (true). Volající funkce `ZjistiSpoj(1)` by poté rovněž vrátila 1 (true) a tato hodnota by byla zpracována hlavním programem.

Celá záležitost se tedy vyřeší, pokud zajistím, aby funkce `ZjistiSpoj()` byla volána pouze pro dosud nenavštívené ostrovy. To lze ovšem zařídit velice snadno: Vytvořím si nové pole `Navstiveny[]`, jehož všechny hodnoty budou zpočátku nastaveny na 0 (false). Hned na začátku funkce `ZjistiSpoj()` potom otestuji, zda daný ostrov nebyl již navštíven, a v případě že ano, ihned funkci opustím. V případě, že se jedná o dosud nenavštívený ostrov, označím ho jako navštívený a provedu v předchozích odstavcích popsanou činnost.

Ve výpisech `r_01.c` a `r_01.pas` jsou uvedeny programy, které využívají výše popsaný algoritmus.

Příklad 2

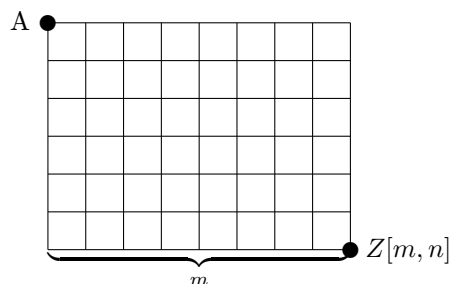
Zadání:

Máme sáček s kuličkami N barev, od každé barvy je 100 kuliček. Napište program, který bude hádat, která kulička bude v dalším kole vytažena ze sáčku. Program bude hádat barvu, která je nejpravděpodobnější – pokud bude mít více barev stejnou pravděpodobnost, tak si barvu vybere náhodně. Pak skutečně vytáhneme (bez vracení) ze sáčku jednu kuličku, programu zadáme barvu, kterou jsme vytáhli. Program bude opět hádat, která kulička bude tažena, a postup se bude opakovat. Nakonec program vypíše počet všech tahů a počet úspěšně uhodnutých barev kuliček. Tahů je maximálně tolik, kolik je kuliček v sáčku, tedy $100 * N$.

Barvy označte čísly 1 až N , ukončení programu je zadání barvy 0. Můžete předpokládat, že počet barev je nejvýše 10.

Řešení:

Jedná se o čistě programovací úlohu, kde není potřeba nad ničím příliš přemýšlet. Mírně znevýhodnění byli méně zkušené programátoři v jazyce C, ve kterém jsou všechna pole automaticky indexována od nuly a nedá se to ovliv-



Obrázek 2: Ilustrace k zadání příkladu č. 3

nit. Řešením je buď vytvoření většího pole, tedy pro barvy od 1 do 10 něco jako

```
int pocty[11];
```

Tato definice vytvoří pole s indexy 0 až 10, prvek s indexem 0 zůstane nevyužitý. Druhou možností je vytvoření pole „správné“ velikosti

```
int pocty[10];
```

v němž jsou ovšem prvky indexovány 0 až 9 a při komunikaci s uživatelem je potřeba vždy tento index zvýšit, příp. snížit o jednu, s čímž souvisí větší riziko zavlečení chyby.

Program se tedy na začátku zeptá na počet barev, potom naplní pole počtů kuliček jednotlivých barev a potom provádí cyklus opakující se až do maximálního možného počtu tahů, který přeruší v případě, že je uživatelem zadána barva 0.

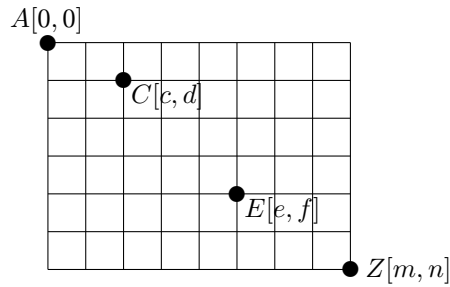
Výpisy r_02.c a r_02.pas ukazují implementaci tohoto problému.

Příklad 3

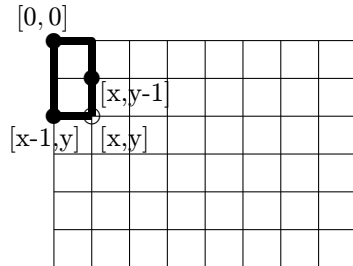
Zadání:

Procházka po čtverečkové síti velikosti m a n je cesta z bodu A do bodu Z , kdy smíme jít pouze po hranách čtverečků, a to jen doprava nebo dolů (viz obr. 2).

- určete počet cest z A do Z na síti o rozměrech m a n .
- určete počet cest z A do Z , pokud musíte projít přes alespoň jeden z bodů C a E . Bod A má souřadnice $[0, 0]$, bod Z má souřadnice $[m, n]$, bod C má souřadnice $[c, d]$, bod E má souřadnice $[e, f]$. Vstup odpovídá obr.



Obrázek 3: Ilustrace k zadání příkladu č. 3



Obrázek 4: Cesty do $[x,y]$ jdou buď přes $[x-1,y]$, nebo $[x,y-1]$

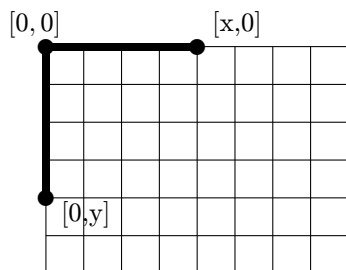
3, tedy bod C leží na některých procházkách z A do E a bod E leží na některých procházkách z C do Z .

Řešení:

Toto je skutečně pěkný příklad, který poskytuje velké množství způsobů řešení. Ukážeme si „programátorský“ přístup k tomuto problému, přestože se nejedná o nejrychlejší způsob řešení.

Uvědomme si následující fakt: Pokud potřebuji znát počet cest, kterými se lze dostat do bodu $[x, y]$ ($= P$) a znám počet cest, kterými se lze dostat do bodu $[x - 1, y]$ ($= P_1$) a počet cest, kterými se lze dostat do bodu $[x, y - 1]$ ($= P_2$), stačí tato dvě čísla sečíst ($P = P_1 + P_2$). Každá z cest, která končí v $[x, y - 1]$, se dá totiž jediným možným způsobem protáhnout do $[x, y]$, podobně se dá každá z cest končících v $[x - 1, y]$ protáhnout jediným možným způsobem do $[x, y]$. Jinudy, než přes $[x - 1, y]$, nebo $[x, y - 1]$ se do $[x, y]$ nedá dostat. Celá situace je ilustrována na obr. 4.

Předchozí úvaha samozřejmě platí pouze pro $x > 0, y > 0$, avšak je zřejmé, že pokud $x = 0$, nebo $y = 0$, je cesta, kterou se můžeme do $[x, y]$ dostat, jediná. Tato situace je ilustrována na obr. 5.



Obrázek 5: Existuje jediná cesta do bodu s nulovou x-ovou nebo y-ovou souřadnicí

Nyní tedy není problémem navrhnout funkci, která bude brát jako parametry souřadnice cílového bodu $[x, y]$ a pomocí rekurzivního volání sebe sama jednou s parametry $[x-1, y]$, podruhé s parametry $[x, y-1]$ a následného součtu získaných hodnot získá výsledný počet cest. V případě, že jeden z jejích parametrů bude roven nule, vrátí hodnotu 1.

V tomto okamžiku máme vymyšlen správný algoritmus a mohli bychom tedy skončit. Podívejme se ale ještě jednou na chování námi vytvořeného programu. Řekněme, že funkce zjišťující počet cest z bodu $[0, 0]$ do bodu $[x, y]$ bude vypadat takto:

Funkce(x, y)
Pokud $x=0$ nebo $y=0$, vrať 1
Jinak vrať $Funkce(x-1, y) + Funkce(x, y-1)$

Stručně si rozeberme, jak bude vypadat výpočet po volání $Funkce(4, 2)$:
 $Funkce(4, 2)$ zavolá

- $Funkce(3, 2)$, která zavolá
 - $Funkce(2, 2) \dots$
 - $Funkce(3, 1) \dots$
- $Funkce(4, 1)$, která zavolá
 - $Funkce(3, 1) \dots$
 - $Funkce(4, 0)$ (okamžitě vrátí 1)

Nevypisuji všechny volání $Funkce()$, již z těchto několika řádků je vidět, že $Funkce(3, 1)$ je volána dvakrát. Dvakrát tedy probíhá tentýž výpočet hodnoty

této funkce. Pokud bychom si podrobněji rozepsali pořadí volání `Funkce()`, zjistili bychom, že se nejedná o ojedinělý případ, naopak, `Funkce()` je s některými dvojicemi parametrů volána ještě mnohem víckrát. Nebylo by možné určovat počet cest do každého bodu jenom jednou? Ušetřili bychom tím v případě větších hodnot m a n spoustu času. Pokud se ještě jednou podíváme na způsob, jakým se zjišťují počty cest do jednotlivých bodů, zjistíme, že taková možnost existuje. Připravme si pole o rozměrech $[0..m][0..n]$. Pokud zařídíme, aby v každém prvku $[x][y]$ tohoto pole byl počet cest, kterými se do bodu $[x][y]$ mohu dostat, bude v prvku $[m][n]$ hledaný počet cest.

První řádek a sloupec tohoto pole (tj. prvky se souřadnicemi typu $[x][0]$ a $[0][y]$) vyplníme na začátku hodnotou 1. To odpovídá již dříve zmíněné skutečnosti, že do bodů s těmito souřadnicemi existuje z bodu $[0,0]$ vždy jen jediná cesta (viz obr. 5). Nyní vyplňujeme pole po řádcích, do prvku $[x][y]$ dosadím součet hodnot v políčkách $[x-1][y]$ a $[x][y-1]$. Jelikož postupují po řádcích zleva doprava, odshora dolů, mám jistotu, že hodnoty v těchto políčkách jsou již správně spočítány, a proto bude i hodnota v $[x][y]$ dobře spočítána.

Popis tohoto postupu vypadá složitě, ale po prohlédnutí vzorových řešení `r_03.pas` nebo `r_03.c` a vyzkoušení funkčnosti na polích menších rozměrů by neměl být problém hlavní myšlenku pochopit.

Jenom okrajově zmíním třetí, čistě matematický postup: počet cest z $[0,0]$ do $[m,n]$ je roven $\binom{m+n}{n} = \frac{(m+n)!}{m!n!} = \frac{(m+n) \cdot \dots \cdot (m+1)}{n \cdot \dots \cdot 1}$. Tato rovnost převádí celý problém na určení hodnoty faktoriálu. Jedná se o nejrychlejší metodu zjištění správného výsledku, tato metoda je však vázána na znalost některých matematických vzorců a navíc nemusí být vždy použitelná.

Co se týká vyřešení druhé části problému, vypůjčíme si opět matematickou teorii. Pokud chci zjistit, kolik cest z $[0,0]$ do $[m,n]$ prochází přes alespoň jeden z bodů C , E , budu postupovat následujícím způsobem: nejdřív zjistím počet cest, které procházejí bodem C (bez ohledu na to, zda zároveň procházejí i bodem E) – toto číslo označím P_c . Dále zjistím počet cest jdoucích přes E (bez ohledu na to, zda zároveň procházejí bodem C) – toto číslo označím P_e . A konečně počet cest, které procházejí jak bodem C , tak i bodem E , označím P_{ce} . Pokud nyní sečtu P_c a P_e , dostanu počet cest jdoucích přes C nebo přes E a ještě něco navíc. V tomto součtu jsou totiž dvakrát započteny cesty, které jdou přes C i přes E (proč?). Pokud od tohoto součtu odečtu hodnotu P_{ce} , dostávám požadovaný počet cest, tedy $P = P_c + P_e - P_{ce}$.

Zbývá vyřešit jedinou otázku, jak spočítat, kolik cest z $[0,0]$ do $[m,n]$ vede přes $[c,d]$. Každá cesta, která vede z $[0,0]$ do $[c,d]$, může pokračovat libovolnou z cest vedoucích z $[c,d]$ do $[m,n]$. Přitom počet cest z $[0,0]$ do $[c,d]$ je roven hodnotě `Funkce(c,d)`, počet cest z $[c,d]$ do $[m,n]$ je roven hodnotě `Funkce(m-c,n-d)`. Celkový počet cest z $[0,0]$ do $[m,n]$ vedoucích přes $[c,d]$ je

tedy $P_c = \text{Funkce}(c,d) * \text{Funkce}(m-c,n-d)$. Obdobně počet cest vedoucích z $[0,0]$ do $[m,n]$ přes $[e,f]$ je $P_e = \text{Funkce}(e,f) * \text{Funkce}(m-e,n-f)$ a počet cest vedoucích přes oba body je $P_{ce} = \text{Funkce}(c,d) * \text{Funkce}(e-c, f-d) * \text{Funkce}(m-e,n-f)$. Dosazením do původního vzorce dostáváme výsledek

$$P = \text{Funkce}(c,d) * \text{Funkce}(m-c,n-d) + \\ + \text{Funkce}(e,f) * \text{Funkce}(m-e,n-f) - \\ - \text{Funkce}(c, d) * \text{Funkce}(e-c,f-d) * \text{Funkce}(m-e,n-f)$$

Jelikož se jedná o čistě matematický problém, není toto řešení ve vzorových programech uvedeno.

Na závěr:

Celkem mě překvapilo, že se někteří z vás ze začátku vrhli na úlohu číslo 1, která rozhodně nebyla nejjednodušší a zaměstnala vás na dost dlouho, takže jste neměli dost času na řešení podstatně jednodušší úlohy číslo 2. To se také projevilo v nízkých bodových ziscích některých z vás

Některým z vás by rozhodně prospělo nejdřív si pořádně všechno rozmyslet a klidně nakreslit nebo napsat na papír, a teprve potom začít programovat. Není nic horšího, než když dopíšete program, a s hrůzou zjistíte, že vůbec neděá to, co jste po něm chtěli. Kdybyste si předem rozmysleli, jak bude váš program při výpočtu postupovat, a nejdřív si tento postup vyzkoušeli na nějakých datech, mohli byste lépe odhalit chyby, které se v návrhu vyskytují a buď provést v návrhu patřičné úpravy, nebo se do psaní nefunkčního programu vůbec nepouštět.

Na druhou stranu mě příjemně překvapilo, že se nevyskytli soutěžící, kteří by si „ani nefukli“. I ti, kteří mají nízké bodové zisky, napsali většinou ceé programy, které bohužel ne vždy fungovaly bez chyb — chyby byly však většinou zaviněny nekvalitním návrhem, jak bylo popsáno v předchozím odstavci, a proto lze věřit, že osvojením lepších postupů při vývoji se vaše programy podstatně zepší.

Mnoho úspěchů při psaní dalších programů vám za ty, kdo se na organizaci tohoto kola podíleli (Jakub Fischer, Pavel Míka, Jan Vodička) přeje

Jan Vodička

Výpisy programů

Poznámka: za správnost ručím pouze u C-čkových programů. Pascal už si pamatuji jenom mlhavě, a proto je možné, že v Pascalovských programech je nějaká syntaktická (ale nikoliv logická) chyba. Omlouvám se programátorům v Basicu, na ten už moje síly nestačí, a proto dobrá rada: naučte se Pascal, nebo ještě raději C.

r.01.c

```
#include <stdio.h>

#define OSTROVY 5

int navstiveny[OSTROVY+1];
int spoj[OSTROVY+1][OSTROVY+1];

int ZjistoSpoj(int odkud, int cil)
{
    int i;

    /* V pripade, ze jsem v cili, vratim 1 */
    if (odkud==cil)
        return 1;
    /* V pripade, ze jsem na tomto ostrove jiz byl, vratim 0 */
    if (navstiveny[odkud])
        return 0;
    /* Jinak ho rovnou oznacim, abych se sem uz nevracel */
    navstiveny[odkud]=1;
    /* Projdi vsechny ostrovy */
    for (i=1;i<=5;i++)
        /* Pokud se jedna o jiny ostrov */
        if (i!=odkud)
            /* Pokud existuje prymy spoj */
            if (spoj[odkud][i])
                /* Pokud z ostrova i lze cestovat do cile */
                if (ZjistoSpoj(i,cil))
                    return 1;
    /* Vyzkousel jsem vsechny ostrovy a nikde neuspel, vratim tedy 0 */
    return 0;
}

int main(void)
{
```

```

int from, to;
int znovu;
int i,j;

/* Prestoze tuto inicializaci vetsina prekladacu provede
automaticky, neni vubec od veci ji tady uvest */

for (i=0;i<=OSTROVY;i++)
    navstiveny[i]=0;
for (i=0;i<=OSTROVY;i++)
    for (j=0;j<=OSTROVY;j++)
        spoj[i][j]=0;

do {
    printf("Zadej spoj ve tvaru x-y (0-0=konec):");
    do {
        znovu=0;
        scanf("%i-%i",&from,&to);
        if (from<0 || to<0 || from>OSTROVY || to >OSTROVY)
        {
            printf("Povolene rozmezi je 1 az %i\n",OSTROVY);
            znovu=1;
        }
        spoj[from][to]=1;
    } while (znovu);
} while(from != 0 && to != 0);

printf("Zadej hledane spojeni ve tvaru x-y :");
do {
    znovu=0;
    scanf("%i-%i",&from,&to);
    if (from<=0 || to<=0 || from>OSTROVY || to >OSTROVY)
    {
        printf("Povolene rozmezi je 1 az %i\n",OSTROVY);
        znovu=1;
    }
} while (znovu);
if (ZjistiSpoj(from,to))
    printf("Cesta %i -> %i existuje\n",from,to);
else

```

```

    printf("Cesta %i -> %i neexistuje\n",from,to);
    return 0;
}

```

r_01.pas

```

Program r_01;

```

```

const ostrovy=5;

```

```

Var

```

```

    navstiveny : array[0..5] of boolean;
    spoj : array[0..5][0..5] of boolean;

```

```

Function ZjistoSpoj(odkud : Integer; cil : Integer) : Integer;

```

```

Var

```

```

    i : Integer;

```

```

Begin

```

```

{ V pripade, ze jsem v cili, vratim 1 }

```

```

If (odkud=cil) Then

```

```

    Begin

```

```

        ZjistoSpoj := 1;

```

```

        Exit;

```

```

    End { If };

```

```

{ V pripade, ze jsem na tomto ostrove jiz byl, vratim 0 }

```

```

If (navstiveny[odkud]<>0) Then

```

```

    Begin

```

```

        ZjistoSpoj := 0;

```

```

        Exit;

```

```

    End { If };

```

```

{ Jinak ho rovnou oznacim, abych se sem uz nevracel }

```

```

navstiveny[odkud]:=1;

```

```

{ Projdi vsechny ostrovy }

```

```

for i:=1 to ostrovy do

```

```

    { Pokud se jedna o jiny ostrov }

```

```

    If (i<>odkud) Then

```

```

        { Pokud existuje primy spoj }

```

```

        If (spoj[odkud][i]<>0) Then

```

```

            { Pokud z ostrova i lze cestovat do cile }

```

```

        If (ZjistoSpoj(i, cil)<>0) Then
            Begin
                ZjistoSpoj := 1;
                Exit;
                { Vyzkousel jsem vsechny ostrovy a nikde neuspel,
vratim tedy 0 }
            End { If };
ZjistoSpoj := 0;
Exit;
End; { ZjistoSpoj }

Var
    from : Integer;
    to_1 : Integer;
    znovu : boolean;
    i : Integer;
    j : Integer;
Begin
{ Prestoze tuto inicializaci vetsina prekladacu provede automaticky, }
{ neni vubec od veci ji tady uvest }
for i:=0 to ostrovy do
navstiveny[i]=false;
for i:=0 to ostrovy do
for j:=0 to ostrovy do
spoj[i][j]=false;

Repeat
    Write('Zadej spoj ve tvaru x y (0 0=konec):');
    Repeat
        znovu:=false;
        Read(from, to_1);
        If (from<0) Or (to_1<0) Or (from>OSTROVY) Or (to_1>OSTROVY) Then
            Begin
                Writeln('Povolene rozmezi je 1 az ', OSTROVY);
                znovu:=true;
            End;
        spoj[from][to_1]:=true;
    Until Not (znovu);
Until Not (((from<>0) And (to_1<>0)));

```

```

Write('Zadej hledane spojeni ve tvaru x y :');
Repeat
    znovu:=false;
    Read(from, to_1);
    If (from<=0) Or (to_1<=0) Or (from>OSTROVY) Or (to_1>OSTROVY) Then
        Begin
            Writeln('Povolene rozmezi je 1 az ', OSTROVY);
            znovu:=true;
        End;
Until Not (znovu);
If (ZjistiSpoj(from, to_1)<>0) Then
    Writeln('Cesta ', from, ' -> ', to_1, ' existuje');
Else
    Writeln('Cesta ', from, ' -> ', to_1, ' neexistuje');
End.

```

r.02.c

```

#include <stdlib.h>

#define BAREV 10

int pocty[BAREV+1];
int kandidati[BAREV+1];

int main(void)
{
    int i,j;
    int druhu;
    int tahu,spravne;
    int tah,tip;
    int stejne;
    int max;

    tahu=0;
    spravne=0;

    printf("Zadej pocet barev (1 az %i):\n",BAREV);
    while(1)

```

```

{
    scanf("%i",&druhu);
    if (druhu<1 || druhu>BAREV)
        printf("Povolene rozpeti je 1 az %i\n",BAREV);
    else
        /* Pokud je vstup ve spravnem rozpeti, vyskoc z cyklu */
        break;
}
/* Inicializace pole poctu */
for (i=1;i<=druhu;i++)
    pocty[i]=100;
for (i=0;i<100*druhu;i++)
{
    stejne=0;
    max=0;
    for (j=1;j<=druhu;j++)
    {
        /* Pokud jsem narazil na dosud nejvyssi pocet, upravim max */
        if (max<pocty[j])
        {
            max=pocty[j];
            kandidati[0]=j;
stejne=1;
        }
        /* Pokud je tento pocet pouze roven maximu, pridam tuto barvu
        mezi kandidaty na losovani */
        else if (max==pocty[j])
        {
            kandidati[stejne]=j;
stejne++;
        }
    }
}
/* Tento zpusob ziskani nahodneho cisla neni uplne idealni ...
nicmene pro nase ucely plne postacuje */

tip=kandidati[random() % stejne];
printf ("Muj tip: %i\n",tip);
printf ("Cos vytahl?\n");

while(1)

```

```

{
    scanf("%i",&tah);
    if (tah<0 || tah>druhu)
        printf("Neplacej nesmysly (mozne rozpeti 1 az %i, 0 \
pro konec)\n",druhu);
    else if (pocety[tah]<=0 && tah>0)
        printf("Tato barva uz se v sacku nevyskytuje\n");
    else
        break;
}

if (tah==0)
    break;
if (tah==tip)
    spravne++;
    tahu++;
    pocety[tah]--;
}
printf ("Celkovy pocet tahu: %i, z toho spravne uhodnuto: %i\n", \
    tahu,spravne);
}

```

r_02.pas

Program r_02;

const barev=10;

Var

```

pocety : array [1..barev] of integer;
kandidati : array [1..barev] of integer;
i : Integer;
j : Integer;
druhu : Integer;
tahu : Integer;
spravne : Integer;
tah : Integer;
tip : Integer;
stejne : Integer;
max : Integer;

```

```

Begin
tahu:=0;
spravne:=0;
Writeln('Zadej pocet barev (1 az ', BAREV, '):');
While (True) Do
    Begin
        Read(druhu);
        If ((druhu<1) Or (druhu>BAREV)) Then
            Writeln('Povolene rozpeti je 1 az ', BAREV, '));
        Else
            break;
        End { While };

{ Inicializace pole poctu }
for i:=1 to druhu do
    pocty[i]:=100;

for i:=1 to 100*druhu do
    Begin
        stejne:=0;
        max:=0;
        for j:=1 to druhu do
            Begin
                { Pokud jsem narazil na dosud nejvyssi pocet, upravim maximum }
                If (max<pocty[j]) Then
                    Begin
                        max:=pocty[j];
                        kandidati[0]:=j;
                        stejne:=1;
                    End { If }
                Else
                    { Pokud je tento pocet pouze roven maximum, pridam tuto }
                    { barvu mezi kandidaty na losovani }
                    If (max=pocty[j]) Then
                        Begin
                            kandidati[stejne]:=j;
                            stejne:=Succ(stejne);
                        End { If };
                    End { For };
            End { For };

```



```

tip:=kandidati[random(stejne)];
Writeln('Muj tip: ', tip);
Writeln('Cos vytahl?');
While (True) Do
    Begin
        Read(tah);
        If ((tah<0) Or (tah>druhu)) Then
            Writeln('Neplacej nesmysly (mozne rozpeti 1 az ', druhu, ',
0 pro konec)');
        Else
            If ((pocety[tah]<=0) And (tah>0)) Then
                Writeln('Tato barva uz se v sacku nevyskytuje');
            Else
                break;
        End { While };
    If (tah=0) Then
        break;
    If (tah=tip) Then
        spravne:=spravne+1;
    tahu:=tahu+1;
    pocety[tah]:=pocety[tah]-1;
    End { For };
Writeln('Celkovy pocet tahu: ', tahu, ', z toho spravne uhodnuto: ',
spravne);
End.

```

r.03.c

```

#define ROZMER 100

int pocet[ROZMER+1][ROZMER+1];

int cest(int m,int n)
{
    int i,j;
    for (i=0;i<=ROZMER;i++)
    {
        pocet[0][i]=1;
        pocet[i][0]=1;
    }
    for (i=1;i<=m;i++)

```

```

    for (j=1;j<=n;j++)
        /* Tato mozna ponekud zvlastni podminka je tu kvuli pretecenim
        ... pokud soucet pretece rozmery intu, funkce vrati -1 */
        if (pocet[i-1][j]+pocet[i][j-1]>pocet[i-1][j])
            pocet[i][j]=pocet[i-1][j]+pocet[i][j-1];
        else
            return -1;
    return pocet[m][n];
}

int main(void)
{
    int m,n;
    int celkem;

    printf("Zadej rozmery site ve tvaru m,n:\n");
    while(1)
    {
        scanf("%i,%i",&m,&n);
        if (m<0 || n<0 || m>ROZMER || n>ROZMER)
            printf("Mozne rozpeti rozmeru je 0 az %i\n",ROZMER);
        else
            break;
    }

    if ((celkem=cest(m,n))!=-1)
        printf("Bohuze, pocet cest je prilis veliky\n");
    else
        printf ("Pocet moznych cest z [0,0] do [%i,%i] je %i.\n",m,n, \
            celkem);
}

```

r.03.pas

Program r03;

const rozmer=100;

Var

pocet : array[0..rozmer,0..rozmer] of longint;

```

Function cest( m : Integer; n : Integer) : Integer;
Var
    i : Integer;
    j : Integer;

Begin
for i:=0 to rozmer do
    Begin
        pocet[0,i]:=1;
        pocet[i,0]:=1;
    End { For };
for i:=1 to m do
    for j:=1 to n do
        Begin
            { Tato mozna ponekud zvlastni podminka je tu kvuli preteceni }
            { pokud soucet pretece rozmery longintu, funkce vrati -1 }
            If (pocet[i-1][j]+pocet[i][j-1]>pocet[i-1][j]) Then
                pocet[i][j]:=pocet[i-1][j]+pocet[i][j-1];
            Else
                Begin
                    cest := -1;
                    Exit;
                End { Else };
            End { For };
        End { For };
    cest := pocet[m][n];
End; { cest }

Var
    m : Integer;
    n : Integer;
    celkem : longint;

Begin
Writeln('Zadej rozmery site ve tvaru m n:');
While (True) Do
    Begin
        Read(m, n);
        If (((m<0) Or (n<0)) Or (m>ROZMER)) Or (n>ROZMER)) Then
            Writeln('Mozne rozpeti rozmeru je 0 az ', ROZMER);
    End

```

```

Else
    break;
End { While };
celkem=cest(m,n);
If (celkem=-1) Then
    Writeln('Bohužel, pocet cest je prilis veliky');
Else
    Writeln('Pocet moznych cest z [0,0] do [' , m, ', ', n, '] je ',
        celkem, '.');
End.

```

Výsledková listina

Pořadí	Jméno	Body												Celkem
		1			2			3			3b			
1.	Václav Potoček	6	2,5	1	6	2,5	1	6	3	1	3	1,5	0	33,5
2.	Petr Nohavica	3	0	1	1	1	1	6	3	1	3	1,5	0	21,5
3.	Tomáš Zvala	4	3	1	3	1,5	0	-	-	-	-	-	-	12,5
4.-5.	Pavel Sorejs	1	0	0	6	2	1	-	-	-	-	-	-	10
4.-5.	Jan Vater	3	2	1	3	0	1	-	-	-	-	-	-	10
6.	Jan Hlavsa	0	0	0	5	3	1	0,5	0	0	-	-	-	9,5
7.	Martin Bak	0,5	0	0	3	1,5	0,5	-	-	-	-	-	-	7,5
8.	Jakub Vaniš	0	0	0	3	1,5	0,5	-	-	-	-	-	-	5
9.	Pavel Pokorný	1	1	1	0,5	0,5	0	0,5	0	0	-	-	-	4,5
10.-11.	Dominik Peklo	0	0	0	3	1	0	0	0	0	-	-	-	4
10.-11.	Jan Mikolajczyk	3	1	0	-	-	-	-	-	-	-	-	-	4
12.	Michal Bernhardt	1	0	0	0	0	0	0	0	0,5	-	-	-	1,5
13.	Petr Nohavica	0	1	0	0	0	0	-	-	-	-	-	-	1