Informatika – navazující magisterské studium
Přijímací zkouška z informatiky – 2018 – varianta A

Každá úloha je hodnocena maximálně 25 body.
Všechny své odpovědi zdůvodněte!

1. Postavte na stůl do řady vedle sebe šest stejných skleniček tak, aby čtyři z nich stály dnem nahoru a dvě dnem dolů.
a) Kolika různými způsoby lze takto skleničky rozmístit?
b) Úkolem nyní je z popsaného výchozího rozmístění skleniček dosáhnout situace, kdy budou všechny skleničky v poloze dnem dolů. K tomu máte k dispozici libovolný počet tahů, v každém z nich můžete otočit vždy přesně tři vedle sebe stojící skleničky. Určete, pro která počáteční rozestavění skleniček má tato úloha řešení.

2. V uzlech binárního vyhledávacího stromu jsou uložena celá čísla. Máte za úkol vypustit z tohoto stromu uzel se zadanou hodnotou (pokud se tato hodnota ve stromě nachází). Popište algoritmus pro řešení tohoto úkolu, zdůvodněte jeho správnost a odvoďte asymptotickou časovou složitost. Stačí slovní popis algoritmu, nepište program.

3. Navrhněte deterministický konečný automat nad abecedou {0, 1}, který přijímá všechna slova zakončená alespoň třemi stejnými znaky. Například slova 111, 001111, 000111000 automat přijme, zatímco slova 11, 11100, 11011 nepřijme. Přechodovou funkci automatu zapište ve tvaru tabulky a automat znázorněte ve tvaru přechodového diagramu. Navrhněte co nejjednodušší automat, tzn. takový, který bude mít minimální počet stavů.

4. Je dán následující program (obě zadání v Pascalu a v C jsou ekvivalentní):

```
program AA;
var I, J, P, N: integer;
begin
  read(N);
  P:=0;
  for I:=0 to N do
    for J:=I to N do
      if J mod 2 = 0 then
        P:=P + 1;
  write(P)
end.
```

```
#include <stdio.h>
void main(void)
{
  int i, j, p, n;
  scanf("%i", &n);
  p=0;
  for(i=0; i<=n; i++)
    for(j=i; j<=n; j++)
      if (!(j%2)) p++;
  printf("%i", p);
}
```

a) Určete, jaký výsledek obdržíme při výpočtu se vstupní hodnotou N=201.

b) Určete všechny takové vstupní hodnoty N, pro které výpočet skončí s výsledkem 2500.

c) Určete nejmenší vstupní hodnotu N, pro niž je výsledkem výpočtu čtyřciferné číslo.

Informatika – navazující magisterské studium
Přijímací zkouška z informatiky – 2018 – varianta B

Každá úloha je hodnocena maximálně 25 body.
Všechny své odpovědi zdůvodněte!

1. a) Čtverečkovou mřížku velikosti 3×12 políček chceme celou pokrýt dvanácti obdélníky o velikosti 1×3 políčka. Určete, kolika různými způsoby to můžeme udělat. Obdélníky můžeme libovolně natáčet.

b) Odvoďte obecný vztah pro počet různých pokrytí mřížky velikosti $3 \times N$ políček pomocí N obdélníků o velikosti 1×3 políčka.

2. Souvislý neorientovaný graf je zadán počtem vrcholů a výčtem hran. Máte za úkol nalézt kostru tohoto grafu (jednu libovolnou, pokud jich existuje více). Popište algoritmus pro řešení tohoto úkolu, zdůvodněte jeho správnost a odvoďte asymptotickou časovou složitost. Stačí slovní popis algoritmu, nepište program.

3. Navrhněte deterministický konečný automat nad abecedou {0, 1}, který přijímá všechna slova sudé délky, která obsahují více než dva znaky 1. Například slova 1011, 1111, 00011100 automat přijme, zatímco slova 111, 1100, 11011 nepřijme. Přechodovou funkci automatu zapište ve tvaru tabulky a automat znázorněte ve tvaru přechodového diagramu. Navrhněte co nejjednodušší automat, tzn. takový, který bude mít minimální počet stavů.

4. Je dán následující program (obě zadání v Pascalu a v jazyce C jsou ekvivalentní):

```
program BB;
var a, b: integer;
begin
  b := 0;
  while b <= 2000 do
    begin
    a := 0;
    while a <= b do a:=a+1;
    b:=b+a;
    end;
  writeln(b)
end.
```

```
main() /* BB */
{
  int a, b;
  b = 0;
  while (b <= 2000)
  {
    a = 0;
    while (a <= b) a++;
    b += a;
  }
  printf("%d", b);
}
```

a) Určete, kolikrát se vykoná tělo vnějšího while-cyklu.

b) Určete výslednou hodnotu proměnné *b*.

Řešení přijímací zkoušky z informatiky – 2018 – varianta A

1. Označme skleničku stojící dnem dolů znakem 0 a skleničku postavenou dnem nahoru znakem 1. Každé rozmístění skleniček pak můžeme popisovat uspořádanou šesticí tvořenou nulami a jedničkami.

a) Dvě převrácené skleničky ze šesti můžeme zvolit (6 nad 2) = **15** způsoby. Všech 15 možných rozmístění skleniček vyhovujících zadání úlohy můžeme vypsat také výčtem: 001111, 010111, 011011, 011101, 011110, 100111, 101011, 101101, 101110, 110011, 110101, 110110, 111001, 111010, 111100.

b) Bylo by velmi pracné zkoušet pro každé z 15 možných výchozích rozmístění skleniček, zda z něj lze dosáhnout stavu 000000. Vyjdeme proto od cílového stavu 000000, kdy jsou všechny skleničky dnem dolů, a budeme zkoušet provádět všechny možné tahy. Tak postupně získáme všechna dosažitelná rozmístění šesti skleniček a z nich vybereme ta, v nichž stojí dvě skleničky dnem dolů a čtyři dnem nahoru. Opakující se rozmístění skleniček nevypisujeme:

po 1. tahu: 111000, 011100, 001110, 000111
po 2. tahu: 100100, 110110, 111111, 010010, 011011, 001001
po 3. tahu: 101010, 100011, 110001, 010101
po 4. tahu: 010010, 101101

Úloha má **tři řešení**, vyhovují situace **110110**, **011011** a **101101**. Tyto situace obsahující dvě 0 a čtyři 1 (a žádné jiné situace této vlastnosti) jsou dosažitelné ze stavu 000000, a proto také právě z nich je posloupností povolených tahů dosažitelné cílové rozmístění skleniček 000000.

Jiný postup řešení: V řadě šesti skleniček lze pouze čtyřmi způsoby zvolit trojici sousedních. Nemá smysl uvažovat posloupnosti tahů, v nichž se nějaká volba otáčené trojice skleniček opakuje (účinek takových dvou tahů by se vzájemně vyrušil), stačí tedy sledovat posloupnosti tahů délky nejvýše čtyři. Spočítejme nejprve, kolik tahů je třeba provést, aby se oproti výchozí situaci změnila poloha přesně čtyř skleniček. Lichým počtem tahů se vždy změní poloha lichého počtu skleniček, takže to musí být buď 2 nebo 4 tahy. V případě dvou tahů musí mít obě zvolené trojice otáčených skleniček přesně jednu skleničku společnou (překrývají se v ní) – takové volby existují dvě a vedou k situacím **011011** a **110110.** V případě čtyř tahů se provedou všechny čtyři různé možné tahy, což vede k vyhovující výsledné situaci **101101.** Úloha má tedy **tři řešení.**

2. V BVS nejprve vyhledáme zadanou hodnotu. Postupujeme do kořene stromu směrem k listům a na základě porovnání hledané hodnoty s hodnotou uloženou v uzlu BVS jdeme vždy do levého nebo do pravého syna. Pokud hodnotu nenajdeme, nemůžeme žádný uzel vypustit. Když ji najdeme, rozlišíme tři případy podle počtu synů tohoto uzlu. Je-li to list, jednoduše ho zrušíme. Má-li náš uzel jednoho syna, uzel zrušíme a jeho syna ve stromu připojíme místo zrušeného uzlu. Má-li dva syny, uzel zrušit nemůžeme. Jeho hodnotu ale vypustíme a nahradíme ji maximální hodnotou z levého podstromu (nebo symetricky minimální hodnotou z pravého podstromu). Maximální hodnotu v levém podstromu najdeme co nejvíce vpravo – sestupujeme od kořene podstromu stále doprava, dokud to jde. Po zkopírování hodnoty ještě vypustíme ten uzel, v němž se původně nacházela (což je snadné, jistě nemá pravého syna, takže je to buď list, nebo má jenom jednoho syna).

Asymptotická časová složitost popsaného algoritmu je nejvýše rovna výšce BVS. Nejprve jsme sestupovali stromem od kořene k listům, dokud jsme nenašli uzel s vypouštěnou hodnotou. Od něj jsme pak v některých případech postupovali dále směrem k listům stromu, dokud jsme nenašli uzel se správnou náhradní hodnotou. Počet kroků algoritmu je proto nejvýše roven výšce stromu, v mnoha případech je ovšem nižší (buď nemusíme hledat náhradní hodnotu, nebo ji najdeme dříve). Binární strom s N uzly má v nejlepším a také v průměrném případě výšku O(log N), v nejhorším případě může mít výšku až O(N). Proto i asymptotická časová složitost popsaného algoritmu je v nejhorším a v průměrném případě O(log N), zatímco v nejhorším případě je O(N).

3. Pomocí stavů automatu musíme počítat, kolik stejných znaků za sebou (a jakých) jsme přečetli ze vstupního slova. Končí-li slovo ve chvíli, kdy má automat přečtené alespoň tři stejné znaky za sebou, slovo přijme. Ve stavu A nebylo přečteno nic, ve stavech B, C byla přečtena jedna 0 resp. 1, ve stavech D, E byly přečteny dva znaky 0 resp. 1, ve stavech F, G byly přečteny nejméně tři znaky 0 resp. 1 souvisle po sobě. Minimalitu počtu stavů ověříme redukcí sestrojeného konečného automatu.

		0	1	
→	A	B	C	počáteční stav, nic nepřečteno
	B	D	C	slovo končí jednou 0
	C	B	E	slovo končí jednou 1
	D	F	C	slovo končí dvěma 0
	E	B	G	slovo končí dvěma 1
←	F	F	C	slovo končí alespoň třemi znaky 0
←	G	B	G	slovo končí alespoň třemi znaky 1

4. Pro každé i od 0 do N se do P přičte počet sudých čísel ležících v intervalu $<i, N>$. Rozlišíme dva případy podle toho, zda N je sudé nebo liché. Výsledná hodnota P pro N sudé je:
$P = (N/2+1) + 2(N/2 + ... + 2 + 1) = (N/2+1)^2$.
Pro N liché dostaneme jen mírně změněný vztah: $P = ((N-1)/2+1)+2((N-1)/2+...+2+1) = ((N-1)/2+1)^2$.

a) Úlohu řešíme přímým dosazením do odvozeného vzorce pro lichá N, dostaneme výsledek P=**10201**.

b) Musíme uvažovat možnost sudého i lichého N:
$(N/2+1)^2=2500$, $N/2+1=50$, $N=98$
$((N-1)/2 + 1)^2=2500$, $(N-1)/2 + 1=50$, $N=99$
Výsledek 2500 tedy získáme pro vstupní hodnoty **98** a **99**.

c) Nejmenší čtyřciferné číslo, které je kvadrátem, je $1024 = 32^2$ (platí $31^2=961$). Výsledku 1024 dosáhneme pro N sudé 62 a N liché 63 (což odvodíme stejným způsobem jako v řešení úlohy b). Menší z nich je 62, takže **62** je nejmenší vstupní hodnota se čtyřciferným výsledkem.

Řešení přijímací zkoušky z informatiky – 2018 – varianta B

1. Počet různých způsobů pokrytí mřížky o velikosti 3 x *N* políček označíme *P(N)*. Hledáme hodnotu *P(12)*. V prvním sloupci mřížky umístíme obdélník 1x3 buď svisle, a pak za ním následuje *P(11)* možností pokrytí zbytku mřížky, nebo vodorovně (druhý a třetí nutně vodorovně pod něj) a za nimi následuje *P(9)* dalších možností. Celkově tedy *P(12) = P(11) + P(9)*, **obecně *P(N) = P(N-1) + P(N-3)* pro všechna *N* > 3. Zřejmě platí *P(3) = 2, P(2) = 1, P(1) = 1***. Výsledný počet *P(12)* snadno dopočítáme podle uvedeného vzorce ve dvanácti krocích:
1 1 2 3 4 6 9 13 19 28 41 **60**

Jiný postup řešení pro *N*=12: Předpokládejme, že mřížka je umístěna vodorovně. Obdélníky 1x3 v ní mohou být umístěny buď svisle, nebo vodorovně – ale v tom případě jedině vždy tři nad sebou. Rozlišíme možné případy podle počtu trojic vodorovných obdélníků:
- žádná vodorovná trojice, 12 svislých 1 možnost
- jedna vodorovná trojice, 9 svislých 10 možností
- dvě vodorovné trojice, 6 svislých (8 nad 2) = 28 možností
- tři vodorovné trojice, 3 svislé (6 nad 3) = 20 možností
- čtyři vodorovné trojice, žádný svislý 1 možnost

Celkem tedy dostáváme 1 + 10 + 28 + 20 + 1 = **60** možností.

2. Kostru neorientovaného souvislého grafu můžeme nalézt prohledáváním grafu do hloubky nebo do šířky. Zvolíme libovolný výchozí vrchol a z něj spustíme prohledávání, při kterém si označujeme již navštívené vrcholy, abychom je nezpracovávali vícekrát. Do vytvářené kostry přitom zařazujeme právě ty procházené hrany, které vedou do ještě nenavštíveného vrcholu. Ze souvislosti plyne, že nakonec navštívíme všechny vrcholy. Do každého z nich (kromě prvního) přijdeme po nějaké hraně poprvé a těchto *N*-1 hran tvoří nalezenou kostru. Z postupu přímo plyne, že do kostry nikdy nezařadíme hrany, které by tvořily cyklus, takže zvolených *N*-1 hran opravdu tvoří kostru původního grafu o *N* vrcholech. Asymptotická časová složitost závisí na tom, jaké použijeme datové struktury. Při vhodně volbě lze dosáhnout složitosti O(*N*+*M*), kde *N* je počet vrcholů a *M* je počet hran grafu. Existuje i jiný postup řešení pomocí faktorových množin.

3. Pomocí stavů automatu musíme počítat počet jedniček přečtených ze vstupu a evidovat paritu délky vstupního slova. Ve stavech A, B nebyla přečtena žádná 1, ve stavech C, D byla přečtena jedna 1, ve stavech E, F byly přečteny dvě 1, ve stavech G, H více než dvě 1 (přesný počet už není zajímavý). Minimalitu počtu stavů ověříme redukcí sestrojeného konečného automatu.

	0	1	
→ A	B	C	přečtena žádná 1, sudá délka
B	A	D	přečtena žádná 1, lichá délka
C	D	E	přečtena jedna 1, lichá délka
D	C	F	přečtena jedna 1, sudá délka
E	F	G	přečteny dvě 1, sudá délka
F	E	H	přečteny dvě 1, lichá délka
G	H	H	přečteny více než dvě 1, lichá délka
← H	G	G	přečteny více než dvě 1, sudá délka

4. Při každém průchodu vnějšího while-cyklu se nejprve do proměnné *a* dosadí hodnota *b*+1 a poté se tato hodnota *a* přičte k proměnné *b*. Hodnota proměnné *b* se tedy celkově změní na 2*b*+1. Proměnná *b* proto po jednotlivých iteracích nabývá hodnot 1, 3, 7, 15, …, obecně po provedení *i*-té iterace vnějšího cyklu má hodnotu 2^i-1. Zajímá nás, kdy poprvé překročí hodnotu 2000. Platí, že 2^{10}-1=1023, 2^{11}-1=2047, tzn. dojde k tomu po jedenácté iteraci vnějšího while-cyklu.
a) **11**
b) **2047**