A

Jméno _____

V následujících kvízových otázkách je správně někdy jen jedna, někdy více variant odpovědí. Správně zodpovězená otázka je taková, která určí správně všechny varianty odpovědí a je hodnocena uvedeným počtem bodů. Pokud některá varianta je určena chybně, otázka je hodnocena 0 body. V otázkách s tvořenou odpovědí je pouze správná odpověď hodnocena uvedeným počtem bodů. Než tvořenou odpověď vepíšete do archu, připravte si ji dobře nanečisto na jiném papíru.

1. (5 b.)

a)- kódy všech znaků začínají stejným symbolem,
b)+ hloubka T je 5 (když hloubka kořene je 0),
c)+ dva znaky mají délku kódu rovnou hlubce stromu,
d)+ T má 6 listů,
e)+ jediný znak má kód délky 3.

Četnosti znaků v textu jsou dány tabulkou. Pro statický Huffmanův kód znaků a pro odpovídající Huffmanův strom T platí, že

a	*b*	*c*	*d*	*e*	*f*
22	11	3	50	44	6

2. (5 b.)

a)- nutně obsahuje ε-přechody,
b)+ má více než 10 stavů,
c)+ má 4 koncové stavy,
d)- v každém stavu, který není koncový, má smyčku (= přechod do téhož stavu),
e)+ po odstranění smyček z přechodového diagramu A_1 vznikne acyklický graf.

Pro vzorek $P = accac$ nad abecedou $\{a, b, c\}$ je standardním postupem sestrojen nedeterministický automat A_1, pomocí nějž lze v textu vyhledávat řetězce, které mají od P Hammingovu vzdálenost rovnu nejvýše 3. Pro A_1 platí

3. (5 b.)

Je dán NKA A_1 byl převeden na DKA A_2 použitím standardního algoritmu převodu NKA na DKA. Žádné další úpravy se s A_2 neprováděly. Do A_2 se vloudily chyby. Je zapotřebí provést opravy

a) označit stav 0 jako koncový,
b) doplnit přechod δ(13, *c*) = 02,
c) označit stav 13 jako koncový,
d) doplnit přechod δ(02, *b*) = 13,
e) přidat stav 123 s prázdnými přechody.

A_1	*a*	*b*	*c*	
0		1, 3		
1	2			F
2	0, 2		3	F
3		1		

A_2	*a*	*b*	*c*	
0		13		
13	2	1		
2	02		3	F
1	2			F
02	02		3	F
3		1		

4. (5 b.)

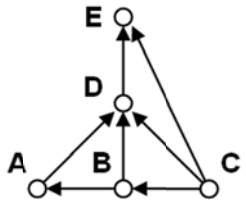
a) *B* může mít více stavů než *A*
b) *B* je deterministický
c) *B* může mít více koncových stavů než *A*
d) diagram *B* je vždy acyklický graf
e) *B* i *A* mají stejný počet koncových stavů

K danému nedeterministickému konečnému automatu A, který obsahuje alespoň jeden ε–přechod) konstruujeme pomocí standardní konstrukce k němu ekvivalentní automat *B* bez ε–přechodů. Platí:

5. (5 b.)

a) $E \le D \le A \le B \le C$
b) $A \le B \le C \le D \le E$
c) $C \le B \le A \le D \le E$
d) $C \le B \le D \le E \le A$
e) $A \le C \le B \le D \le E$

V topologickém uspořádání grafu na obrázku platí pro jednotlivé vrcholy (tranzitivní) relace



6. (10 b.)

a) $m + n - 2$
b) $mn + m + n - 2$
c) $mn - 2$
d) $mn + 2m + 2n - 2$
e) $2mn + m + n - 2$

Strom T_n ($n \ge 3$) s n uzly propojíme s jiným stromem T_m ($m \ge 3$) s m uzly tak, že z každého uzlu T_n vedeme hranu do každého uzlu T_m. Kolik hran má výsledný graf?

A

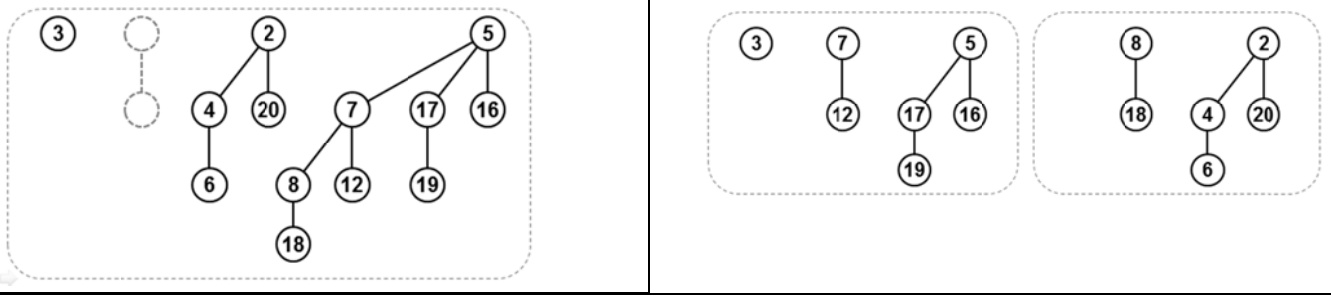
Jméno _____

7. (5 b.)
V jaké třídě složitosti je funkce h v následujícím programu vzhledem k velikosti pole p (budeme ji značit n), když víme, že v proměnné N je vždy velikost pole p?

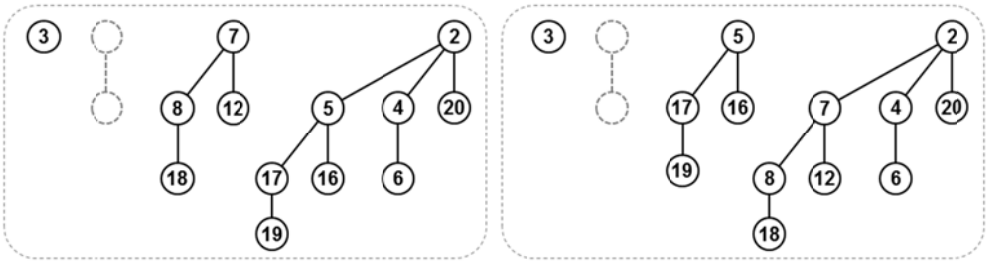
a) $O(n)$
b) $O(\log(n))$
c) $\Omega(1)$
d) $\Omega(\log(\log(n)))$
e) $\Omega(n^2)$
f) $\Theta(n)$
g) $\Theta(1)$
h) $O(n \cdot \log(n))$

```
int h (int p[], int N, int i)
{
      int j,a,s;
      for( j=0, a=N<<1, s=j;
           (j<=N) && (a!=0);
           (p[j]>i)?(j-=s):(j+=s), a>>1 ) {
             s = (s==1)?1:(s>>1);
             if (p[j] == i) return j;
      }
      return -1;
}
```

8. (10 b.)
Když spojíme pomocí operace Merge dvě binomiální haldy na obrázku získáme jedinou výslednou binomiální haldu. Nakreslete ji.



Další dvě možné varianty řešení, stačí kterákoli z uvedených tří:



9. **(5 b.)**

$\boldsymbol{T}(\boldsymbol{x}) =$

$$\begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} -3/4 \\ 1/2 \end{pmatrix}$$

Určete transformaci roviny, která je složená z následujících zobrazení v uvedeném pořadí. Nejprve je vzor $\boldsymbol{x}$ posunut o tři jednotky doleva (v záporném směru osy x) a o jednotku nahoru (v kladném směru osy y). Potom je výsledek zmenšen na poloviční velikost. Transformaci zapište ve tvaru $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z}$, kde A je matice a $\boldsymbol{z}$ je sloupcový vektor.

10. (5 b.)

a) $\Theta(n)$
b) $O(n^2)$
c) $\Theta(n^2)$
d) $\Omega(n^2 \cdot \log(n))$
e) $O(n \cdot \log(n))$

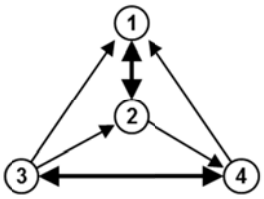
Z binární haldy obsahující n^2 prvků, jejíž kořen obsahuje nejmenší hodnotu z celé haldy, odstraníme n nejmenších prvků. Asymptotická složitost této akce je

11. (10 b.)
Najděte přechodovou matici pro návštěvnost webu se čtyřmi adresami daného obrázkem. Šipky znázorňují možnosti přechodů mezi jednotlivými adresami. Najděte přechodovou matici $\boldsymbol{P}$ pro algoritmus Page Rank pro tento web (5 b.). Určete sloupcový $\boldsymbol{w}$ vektor popisující stacionární režim přechodové matice tohoto webu (5 b.).

A

Jméno _____

Matice $\boldsymbol{P}$:	vektor $\boldsymbol{w}$:
$\begin{pmatrix} 0 & 1/2 & 1/3 & 1/2 \\ 1 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/2 \\ 0 & 1/2 & 1/3 & 0 \end{pmatrix}$	$\begin{pmatrix} 9/28 \\ 10/28 \\ 3/28 \\ 6/28 \end{pmatrix}$

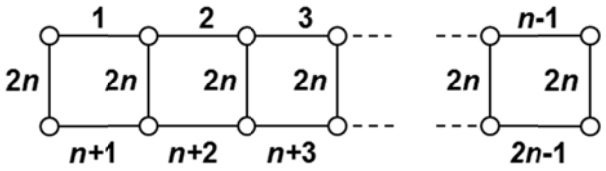


12. (5 b.)

V Kruskalově algoritmu reprezentujeme datovou strukturu Union-Find pomocí orientovaných stromů. Předpokládáme, že při sjednocování stromů zařazujeme vždy menší strom pod kořen většího a nepoužíváme žádné další heuristiky pro zvýšení efektivity. Určete, jaká je hloubka konečného stromu v této struktuře po nalezení minimální kostry daného grafu (kořen má hloubku 0), a nakreslete strom pro $n = 5$.

Hloubka pro obecné $n \geq 3$:

Strom:

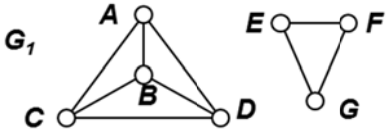


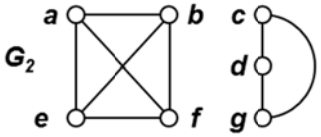
13. (5 b.)

Určete počet izomorfizmů mezi danými dvěma grafy. Pozor, nejsou souvislé.

Počet izomorfizmů:

144



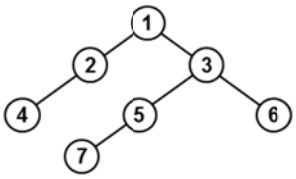


14. (5 b.)

Správné varianty (stačí kterákoli)

Černé:	Červené:
1 2 5 6	3 4 7
1 4 5 6	2 3 7
1 2 3 4 6 7	5
1 2 3 4 5 6	7

Napište, jak je nutno obarvit uzly daného stromu, pokud to má být červenočerný strom.



15. (15 b.)

Uvažujte následující program napsaný v pseudokódu:

Poznámka: metoda `G.neighbors_of(y)` vrací seznam všech vrcholů, které jsou spojeny hranou s vrcholem `y` v grafu `G`.

```
1)    Vertices visited = empty;
2)    procedure search(Graph G, Vertex start_vertex )
3)    {
4)          while (to_visit.number_of_elements() != 0) {
5)               if (v not in visited) then {
6)                    visited.add(v);
7)                    for all Vertex x in G.neighbors_of(v) {
8)                    }
9)               }
10)     }
11) }
```

a) b) c) d) e) . Rozhodněte, která z následujících tvrzení platí:

A **Jméno** _____

a) Přidáním řádku „`Queue to_visit = empty;`“ za řádek 1),
řádku „`to_visit.push(start_vertex);`“ za řádek 3),
řádku „`Vertex v = to_visit.pop();`“ za řádek 4 a
řádku „`to_visit.push(x);`“ za řádek 7)
vznikne následující procedura, která prohledává graf do hloubky.

```
Vertices visited = empty;
Queue to_visit = empty;
procedure search(Graph G, Vertex start_vertex ) {
  to_visit.push(start_vertex);
  while (to_visit.number_of_elements() != 0) {
    Vertex v = to_visit.pop();
      if (v not in visited) then {
        visited.add(v);
         for all Vertex x in G.neighbors_of(v)
           to_visit.push(x);
      }
   }
}
```

b) Přidáním řádku „`Stack to_visit = empty;`“ za řádek 1),
řádku „`to_visit.push(start_vertex);`“ za řádek 3),
řádku „`Vertex v = to_visit.pop();`“ za řádek 4 a
řádku „`to_visit.push(x);`“ za řádek 7)
vznikne následující procedura, která prohledává graf do hloubky.

```
Vertices visited = empty;
Stack to_visit = empty;
procedure search(Graph G, Vertex start_vertex ) {
  to_visit.push(start_vertex);
  while (to_visit.number_of_elements() != 0) {
    Vertex v = to_visit.pop();
    if (v not in visited) then {
       visited.add(v);
       for all Vertex x in G.neighbors_of(v)
         to_visit.push(x);
     }
   }
}
```

c) Přidáním řádku „`search(G,x);`“ za řádek 7) a
smazáním řádků 4) a 9) vznikne následující procedura, která prohledává graf do hloubky.

```
Vertices visited = empty;
procedure search(Graph G, Vertex start_vertex ) {
  if (v not in visited) then {
     visited.add(v);
     for all Vertex x in G.neighbors_of(v)
      search(G,x);
  }
}
```

d) Přidáním řádku „`int a = 0;`“ za řádek 1),
řádku „`Priority_Queue to_visit = empty;`“ za řádek 1),
řádku „`to_visit.push(a++, start_vertex);`“ za řádek 3),
řádku „`Vertex v = to_visit.pop();`“ za řádek 4 a

řádku „`to_visit.push(a++,x);`“ za řádek 7)
vznikne následující procedura, která prohledává celý graf do hloubky.
Poznámka: Metoda `push` má v tomto případě dva argumenty. První je hodnota klíče a druhý jsou data. Metoda `pop` vrací vždy ta data, která byla vložena do prioritní fronty s nejnižší hodnotou klíče. Po provedení operace `pop` je tento klíč z prioritní fronty odstraněn.

```
Vertices visited = empty;
int a = 0;
Priority_Queue to_visit = empty;
procedure search(Graph G, Vertex start_vertex ){
  to_visit.push(a++,start_vertex);
  while (to_visit.number_of_elements() != 0) {
    Vertex v = to_visit.pop();
     if (v not in visited) then {
       visited.add(v);
       for all Vertex x in G.neighbors_of(v)
         to_visit.push(a++,x);
     }
   }
}
```

e) Přídáním řádku „`int a = 0;`“ za řádek 1),
řádku „`Priority_Queue to_visit = empty;`“ za řádek 1),
řádku „`to_visit.push(a--, start_vertex);`“ za řádek 3),
řádku „`Vertex v = to_visit.pop();`“ za řádek 4 a
řádku „`to_visit.push(a++,x);`“ za řádek 7)
vznikne procedura, která prohledává celý graf do hloubky.
Poznámka: Metoda `push` má v tomto případě dva argumenty. První je hodnota klíče a druhý jsou data. Metoda `pop` vrací vždy ta data, která byla vložena do prioritní fronty s nejnižší hodnotou klíče. Po provedení operace `pop` je tento klíč z prioritní fronty odstraněn.

```
Vertices visited = empty;
int a = 0;
Priority_Queue to_visit = empty;
procedure search(Graph G, Vertex start_vertex ) {
  to_visit.push(a--,start_vertex);
  while (to_visit.number_of_elements() != 0) {
    Vertex v = to_visit.pop();
    if (v not in visited) then {
      visited.add(v);
      for all Vertex x in G.neighbors_of(v)
         to_visit.push(a--,x);
    }
  }
}
```