Zdůvodněte, proč funkce $f(n) = n \cdot \log(n)^{-1} \cdot n^{1/2}$ roste rychleji než funkce $g(n) = n$.

Zdůvodněte, proč funkce $f(n) = n^{3/2} \cdot \log(n)$ roste pomaleji než funkce $g(n) = n^2$.

Zdůvodněte, proč funkce $4^{\lg(\lg(n))}$ roste rychleji než funkce $\lg^3(n)$. Symbolem lg značíme logaritmus o základu 2.

Zdůvodněte, proč funkce $n \cdot \lg(n)$ roste alespoň stejně rychle nebo rychleji než než funkce $\lg(n!)$. Symbolem lg značíme logaritmus o základu 2.

Uspořádejte do neklesající posloupnosti podle řádu růstu uvedené funkce poměnné n. Funkce, které zařadit nedokážete, do posloupnosti nevkládejte. Symbolem lg značíme logaritmus o základu 2.

$2^{\lg(\lg(n))}$ $n \cdot \lg(n)$ n^2 $n!$ $(\lg(n))!$ $(3/2)^n$ $\lg(n!)$ $4^{\lg(n)}$ $\sqrt{\lg(n)}$ $\left(\sqrt{2}\right)^{\lg(n)}$

Uspořádejte do neklesající posloupnosti podle řádu růstu uvedené funkce poměnné n. Funkce, které zařadit nedokážete, do posloupnosti nevkládejte. Symbolem lg značíme logaritmus o základu 2.

$\left(\sqrt{2}\right)^{\lg(n)}$ n^2 $n!$ $(\lg(n))!$ $(3/2)^n$ $\lg(n!)$ $2^{\lg(\lg(n))}$ $4^{\lg(n)}$ $\sqrt{\lg(n)}$ $n \cdot \lg(n)$

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

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

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?

```
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;
}
```

Předpokládejte, že máte k dispozici neorientovaný graf G = (V, E), který je reprezentován seznamem hran. Seznam hran není nijak uspořádán a přístup k jeho jednotlivým prvkům je pouze sekvenční (k prvkům nelze přistupovat pomocí indexu). Určete, jaká je za těchto okolností asymptotická složitost algoritmů BFS a DFS.

V jaké třídě složitosti je funkce X v následujícím programu vzhledem k velikosti pole p (budeme ji značit n a odpovídá hodnotě proměnné N)?

```
int N;
int * p;

void X (void)
{     int i,j,t;
      for (i = N; i > 0; i--) {
            t = 0;
            for(j = 1; j < i; j++)
                  if (p[j-1] > p[j]) {
                        p[j]   = p[j-1] - p[j];
                        p[j-1] = p[j-1] - p[j];
                        p[j]   = p[j-1] + p[j];
                        t = 1;
                  }
            if (t == 0) break;
      }
}
```

V jaké třídě složitosti je funkce `push` v následujícím programu vzhledem k velikosti pole `p` (budeme ji značit n)? Předpokládejte, že funkce `malloc` (naalokuje datovou strukturu velikosti argumentu a vrací na tuto strukturu ukazatel) a `free` (uvolní z paměti nealokovanou datovou strukturu, kterou dostane v argumentu) pracují v konstantním čase vzhledem k velikosti pole `p`. Dále předpokládejte, že velikost proměnné `N` vždy před a po provedení funkce `push` koresponduje s velikosti pole `p`. Proměnná `i` je vždy v intervalu `-1` až `N`.

```
int N = 0;
int i = -1;
int * p;
void push (int key)
{
      if (++i >= N) {
            int m = N*2+1;
            int * t = malloc(m*sizeof(int));
            int j;
            for(j = 0; j<N; j++, t[j] = p[j]);
            if (N != 0) free(p);
            N = m;
            p = t;
      }
      p[i] = key;
}
```

Popište jednotlivé reprezentace orientovaného grafu v paměti počítače, které znáte. Pro každou možnou dvojici reprezentací R1, R2 určete, jaká je asymptotická složitost převodu grafu z reprezentace R1 do R2.

Najděte a nakreslete nesouvislý neorientovaný graf s nejmenším možným počtem uzlů, jehož matice incidence I má dvakrát více sloupců než má řádků.

V seznamu je uložena množina všech hran grafu, každá hrana je dvojice <uzel, uzel>. Víme, že graf má N uzlů, že je nesouvislý a že obsahuje komponentu K, která má více než N/2 uzlů. Máme vytvořit nový seznam obsahující (ve stejném formátu) právě všechny hrany komponenty K.
Popište, jak co nejefektivněji budete tuto úlohu řešit, jaké použijete datové struktury a jaká bude složitost vašeho řešení. Pořadí hran v obou seznamech není předepsáno a může být libovolné.

Najděte a nakreslete nesouvislý neorientovaný graf s nejmenším možným počtem uzlů, jehož matice incidence I má dvakrát více sloupců než má řádků.

Najděte a nakreslete co nejmenší nesouvislý neorientovaný graf, jehož matice incidence I obsahuje v každém řádku právě tři jedničky.

Je dán graf G = (V, E). Platí $|V| = n$, $|E| \in \Theta(n \cdot \mathrm{sqrt}(n))$. Graf je reprezentován maticí incidence. Určete asymptotickou složitost algoritmu BFS v závislosti na n, za předpokladu, že algoritmus při zjišťování informací o grafu využívá pouze danou reprezentaci G.

Je dán graf G = (V, E). Platí $|V| = n$, $|E| \in \Theta(n \cdot \log(n))$. Graf je reprezentován seznamem hran v náhodném pořadí. Určete asymptotickou složitost algoritmu BFS v závislosti na n, za předpokladu, že algoritmus při zjišťování informací o grafu využívá pouze danou reprezentaci G.

Když má daný graf n uzlů a $\Theta(n^2)$ hran, potom asymptotická složitost rekurzivního algoritmu DFS (prohledávání do hloubky) je $\Theta(n^2)$, za předpokladu, že během provádění algoritmu máme přístup v konstantním čase ke každému právě zpracovávanému uzlu a ke každé právě zpracovávané hraně. Určete co nejpřesněji, jaká bude asymptotická složitost rekurzivního DFS, pokud doba přístupu ke každému uzlu bude bude konstantní a doba přístupu ke každé hraně bude ve třídě $\Theta(\log(n^2))$.

Když má daný graf n uzlů a $\Theta(n^2)$ hran, potom asymptotická složitost rekurzivního algoritmu DFS (prohledávání do hloubky) je $\Theta(n^2)$, za předpokladu, že během provádění algoritmu máme přístup v konstantním čase ke každému právě zpracovávanému uzlu a ke každé právě zpracovávané hraně. Určete, jaká bude asymptotická složitost rekurzivního DFS, pokud doba přístupu ke každému uzlu bude ve třídě $\Theta(n)$ a doba přístupu ke každé hraně bude ve třídě $\Theta(n^2)$.

Když má daný graf n uzlů a $\Theta(n^2)$ hran, potom asymptotická složitost rekurzivního algoritmu DFS (prohledávání do hloubky) je $\Theta(n^2)$, za předpokladu, že během provádění algoritmu máme přístup v konstantním čase ke každému právě zpracovávanému uzlu a ke každé právě zpracovávané hraně. Určete co nejpřesněji, jaká bude asymptotická složitost rekurzivního DFS, pokud doba přístupu ke každému uzlu bude konstantní a doba přístupu ke každé hraně bude úměrná logaritmu počtu hran.

Když má daný graf n uzlů a $\Theta(n^2)$ hran, potom asymptotická složitost rekurzivního algoritmu DFS (prohledávání do hloubky) je $\Theta(n^2)$, za předpokladu, že během provádění algoritmu máme přístup v konstantním čase ke každému právě zpracovávanému uzlu a ke každé právě zpracovávané hraně. Určete, jaká bude asymptotická složitost rekurzivního DFS, pokud doba přístupu ke každému uzlu bude úměrná počtu uzlů a doba přístupu ke každé hraně bude úměrná počtu hran.

Když má daný graf n uzlů a $\Theta(n \cdot \log n)$ hran, potom asymptotická složitost algoritmu BFS (prohledávání do šířky) je $\Theta(n \cdot \log n)$, za předpokladu, že během provádění algoritmu máme přístup v konstantním čase ke každému právě zpracovávanému uzlu a ke každé právě zpracovávané hraně. Určete, jaká bude asymptotická složitost BFS, pokud doba přístupu ke každému uzlu bude ve třídě $\Theta(n^{3/2})$ a doba přístupu ke každé hraně bude ve třídě $\Theta(n^{1/2})$.

Když má daný graf n uzlů a $\Theta(n \cdot \log n)$ hran, potom asymptotická složitost algoritmu BFS (prohledávání do šířky) je $\Theta(n \cdot \log n)$, za předpokladu, že během provádění algoritmu máme přístup v konstantním čase ke každému právě zpracovávanému uzlu a ke každé právě zpracovávané hraně. Určete, jaká bude asymptotická složitost BFS, pokud doba přístupu ke každému uzlu bude ve třídě $\Theta(n^{1/2})$ a doba přístupu ke každé hraně bude ve třídě $\Theta(n^{3/2})$.

Když má daný souvislý graf n uzlů a m hran, potom asymptotická složitost algoritmu BFS (prohledávání do šířky) je $\Theta(m)$, za předpokladu, že během provádění algoritmu máme přístup v konstantním čase ke každému právě zpracovávanému uzlu a ke každé právě zpracovávané hraně. Jaká bude asymptotická složitost tohoto algoritmu, pokud jediná reprezentace grafu, kterou budeme mít k dispozici, bude matice sousednosti tohoto grafu?

V grafu G s n vrcholy ($n \geq 4$) vede cesta mezi každými dvěma vrcholy. Určete, která z následujících tvrzení platí:
a) G je souvislý , b) G je strom , c) G má alspoň n+1 hran, d) G má alespoň $\lfloor n/2 \rfloor$ hran e) G je úplný graf

Matice B vznikne tak, že matici sousednosti A neorientovaného grafu G vynásobíme samu se sebou, $B = A \cdot A$. Předpokládejme, že označení vrcholů G je shodné s indexací řádků A. Určete, která z následujících tvrzení platí:

a) Hodnota prvku $B[i][i]$ je rovna stupni vrcholu I,
b) hodnota prvku $B[i][i]$ je vždy sudé číslo,
c) hodnota prvku $B[i][j]$ pro $i \neq j$ je rovna vzdálenosti vrcholů i a j,
d) buď $B = A$ nebo $B = A+A$,
e) v B může existovat nulový prvek.

Pravidelný graf stupně k je definován jako graf, v němž mají všechny vrcholy stupeň k. Souvislý pravidelný rovinný graf G stupně 3 je nakreslen tak, že každá stěna (včetně vnější neomezené stěny) G sousedí s právě čtyřmi hranami. Pokud lze počet stěn grafu určit, určete jej co nepřesněji nebo uveďte co nejlepší odhad, pokud to určit nelze, zdůvodněte proč. (V rovinném nakreslení grafu se žádné dvě hrany neprotínají, stěna je každá maximální oblast roviny neobsahující žádnou hranu.)

Matice incidence I neorientovaného grafu G s alespoň šesti uzly obsahuje v každém řádku právě dvě jedničky. Která z následujících tvrzení platí?

a) I obsahuje v každém soupci právě dvě jedničky,
b) G má sudý počet uzlů,
c) G je určitě souvislý,
d) G může být nesouvislý,
e) všechny uzly G mají stupeň 2.

Graf G s n ($n \geq 3$) uzly obsahuje jako svůj podgraf úplný graf K_{n-1} s $n-1$ uzly. Platí v tomto případě některé z uvedených tvrzení?

a) G je souvislý,
b) G má alespoň $n-2$ hran,
c) G má alespoň $n+2$ hran,
d) G má nejvýše $\lfloor n \cdot \log(n) \rfloor$ hran,
e) G může obsahovat vrchol stupně 1.

Z úplného grafu K_{200} odebereme 200 hran tak, aby výsledný graf zůstal souvislý. Jaký je součet stupňů uzlů výsledného grafu?

Úplný graf K_n ($n \geq 3$) propojíme s jiným úplným grafem K_m ($m \geq 3$) tak, že z každého uzlu K_n vedeme hranu do každého uzlu K_m. Kolik hran má výsledný graf pro $n = 20$, $m = 21$?

Graf W_n vznikne tak, že každý uzel kružnice C_n spojíme hranou s dalším přidaným uzlem x (W_n má tedy n+1 uzlů). Určete, kolik maximálně a kolik minimálně různých cest délky 3 v grafu W_n může vést vede mezi dvěma libovolnými uzly původní kružnice C_n, pokud $n \geq 20$. Cesty nemusí být disjunktní.

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

Jaký je součet stupňů všech vrcholů grafu G, když víme, že má 1991 vrcholů, 25541 hran a nejvyšší stupeň vrcholu v grafu je 1887?

Kolik různých cest vede mezi dvěma různými vrcholy v úplném neorientovaném grafu s 6 vrcholy?

Kolik různých cest spojuje všechny vzájemně různé vrcholy stupně 1 v grafu G, když víme, že G je souvislý neorientovaný graf s 1999 vrcholy, 1998 hranami a 7 vrcholy stupně 1?

Jaký je součet stupňů všech vrcholů grafu G, když víme, že má 2369 vrcholů, 26359 hran a nejvyšší stupeň vrcholu v grafu je 1963?