Programování pro matematiky
6. cvičení - Spojové seznamy

Peter Kovács

Doporučuje si promyslet řešení každého z úkolů, pro vaše vlastní ujasnění učiva. Navíc pokud některé z úvah sepíšete můžete získat body

Každý úkol obsahuje za názvem maximální počet bodů, které lze za úkol získat. Vaším úkolem je si vybrat úkoly, které chcete řešiť. Klidně všechny. Úkoly, které jste si vybrali sepište do jednoho souboru a odvzdejte do recodexu. Za celý úkol můžete získat maximálne **3 body**. Po odvezdání ohodnotím každou odevzdanou úlohu. Body sečtu a přidelím vám $max(body, 3)$. Úkoly řešte sami. Pokud spolupracujete v skupině dostanete své body podělené počtem lidí v skupině.

Za úkoly je možné uděliť pouze celočíselné body. Pokud získáte neceločíselný výsledek, bude zaokrouhlen nadol.

Sčítání polynomů (1 bod):

Popište v pseudokódu a okomentujte algoritmus pro sčítávání dvou polynomů v řídké reprezentaci. Polynom je uložen v spojovím seznamu, pričem hodnota každého prvku seznamu je dvojice (exponent, koeficient).

Řešení: Zadáno také k implementaci na praktickém cvičení. Řešení nebude zveřejněno.

Implementace funkcí spojového seznamu (max 1 bod):

Doimplementujte chybějíci funkce třídy *LinkedList* v kódu z cvičení. Za každou správně implementovanou funkci můžete získat 0.25 bodu. Funkce jsou obvzkle na 2-3 řádky. Testovací vstupy jsi musíte vytvořit sami. Dbejte na to, aby ste nezapomněli na žádný okrajový případ.

Řešení: Implementaci naleznete v zdrojáku na webu

Implementace funkcí obousměrného spojového seznamu (max 1 bod):

Doimplementujte chybějíci funkce třídy *BidirectionalLinkedList* v kódu z cvičení. Za každou správně implementovanou funkci můžete získat 0.25 bodu. Funkce jsou obvzkle na 2-3 řádky. Testovací vstupy jsi musíte vytvořit sami. Dbejte na to, aby ste nezapomněli na žádný okrajový případ.

Řešení: Implementaci naleznete v zdrojáku na webu

Třídení spojového seznamu (1 body):

Popište jakým způsobem by jsme mohli co najefektivneji utřídit jednosměrný spojový seznam. Můžeme použít jednom $O(1)$ pomocné paměti. Analyzujte časovou složitost algoritmu.

Řešení: Použijeme merge sort. Nalezneme střed pomocí find_middle, to nám rozpadne seznam na dva. Na oba pak spustíme merge sort a pak je slejeme. Na každé úrovni musí find_middle projít dokopy na všech úsecích O(n) prvků. Slévaní trvá O(n) na každé úrovni a úrovní je O(log n). Lze popsat i nerekurzivně postupně slévám úseky (nejprv úseky délky 1, pak 2, 4 ...), bude náročnejší na implementaci.