Zkoušková úloha Vlak

Datové struktury

Jako vždy je volba datových struktur zcela zásadní součástí řešení. Naštěstí jich nebude mnoho, projdeme si je jednu za druhou a pak nad nimi sestavíme již velmi jednoduchý rekurzivní postup řešení.

Lokomotiva má identifikátor 0, vagóny mají identifikátory 1..*n*, žádnou další speciální reprezentaci pro vagóny nebo lokomotivu nebudeme potřebovat, identifikátory nám pouze poslouží jako indexy.
Budeme potřebovat několik polí indexovaných -- ve shodě s předchozí poznámkou -- od 1 do *n*, díky lokomotivě se nám bude hodit indexování od 0 do *n*.
Zaprvé uložíme váhy vagónů do pole **carriageMass**. Složka **carriageMass[1]** představuje váhu vagónu s identifikátorem 1, atd. (nikoli váhu prvního vagónu ve vlaku).

Dvourozměrné pole logických hodnot **canFollow** bude registrovat, zda dva vagóny mohou být zapojeny za sebou. **canFollow[i][j]** bude mít hodnotu **true** právě tehdy, když vagón **j** může být zapojen za vagónem **i**. Významný je také první řádek tothoto pole, **canFollow[0][j]** bude mít hodnotu **true** právě tehdy, když vagón **j** může být zapojen za lokomotivou. Sloupec s indexem **0**, tj složky **canFollow[i][0]**, bude pro nás bezvýznamný.

Aktuálně zkoumaný vlak budeme reprezentovat polem **train**, v jehož složce s indexem **0** bude hodnota **0** představující lokomotivu a v dalších složkách budou identifikátory jeho jednotlivých vagónů. Pole deklarujeme s délkou *n*+1 (lokomotiva plus nejvýše *n* vagónů). Kupodivu, jak se ukáže dále, nebudeme potřebovat ani přídavnou proměnnou udávající délku aktuálního vlaku, tu zastane parametr rekurzivní funkce. Výhodné je však udržovat si průběžně hodnotu hmotnosti aktuálně zkoumaného vlaku v proměnné **trainMass**.

Vždy, když budeme zkoušet připojit další vagón k aktuálnímu vlaku, musíme vědět, které vagóny jsou ještě k dispozici, proto budeme udržovat logické pole **carriageFree**, v němž bude průběžně stále platit **carriageFree[i] == false**, právě když vagón už nebude k dispozici, to jest už bude zapojen někde v aktuálně zkoumaném vlaku.

S probranými prostředky bychom se již mohli vydat do hledání nejtěžšího vlaku. Úloha žádá na výstupu jediný optimální vlak a my během hledání budeme (bezpochyby) nalézat stále těžší a těžší vlaky, nejtěžší dosud nalezený vlak si budeme pamatovat v poli **bestTrain**, které bude organizováno stejně jako pole **train**, a k němu přidáme dvě proměnné **bestTrainMass** a **bestTrainLength** se zřejmým významem.

Rekurze

Zapojme do prázdného vlaku nejprve lokomotivu. Tím získáme počáteční triviální vlak, který se dále budeme pokoušet prodloužit. Lokomotiva je na pozici nula, další případné vagóny budou obsazovat další pozice. Postup, který bychom mohli snadno formulovat a využít i mimo počítačové zpracování, je tento:

1. Zkontroluj, zda aktuální vlak je těžší než nejdělší dosud nalezený
   a pokud ano, zaregistruj ho jako nejtěžší.
2. Pokud je délka vlaku rovna počtu vagónů v depu,
   vytiskni nejdelší vlak a ukonči program.
3. Projdi všechny volné vagóny (ty, které nejsou zapojeny v aktuálním vlaku)
   a s každým proveď následující akci:
   Pokud lze vagón připojit za poslední vůz (vagón nebo lokomotivu) aktuálního vlaku,
   připoj tento vagón do vlaku,
   opakuj rekurzivně celý postup od bodu 1,
   odpoj tento vagón z vlaku,

Parametrem rekurzivního volání bude aktuální pozice ve vlaku, kam se pokoušíme připojit další volný vagón. Tato pozice udává délku celého aktuálního vlaku včetně lokomotivy, proto si ji nemusíme pamatovat zvlášť, jak jsme již poznamenali výše.

Poznámky
Nalezené řešení má být minimální v lexikografickém pořadí identifikatorů vagónů. O to se nemusíme příliš starat, stačí, když se budeme snažit připojovat jednotlivé volné vagóny vždy v lexikografickém pořadí jejich identifikátorů, to jest -- zcela triviálně -- v pořadí od 1 do *n*. První nalezený nejtěžší vlak pak bude ovšem i prvním nejtěžším v lexikografickém pořadí identifikatorů vagónů a další vlaky stejné hmotnosti již registrovat nebudeme.

Drobnou zajímavostí je, že v implementaci nemusíme prozkoumaný vagón z vlaku nijak explicitně odstraňovat, když se navracíme do vyšších úrovní rekurze. Vždy totiž pracujeme s vlakem jen do místa, které je určeno parametrem rekurze (aktuální délkou vlaku), takže obsah pole **train** na dalších pozicích nijak neovlivňuje výpočet a můžeme v něm prozkoumané vagóny ponechávat.

Každé zaregistrování dosud nejtěžšího nalezeného vlaku je úměrné délce vlaku, nedokážeme to udělat v konstantním čase. Nevíme totož, kdy byl naposled nejtěžší vlak zaregistrován, a tudíž ani nevíme, nakolik se shoduje s aktuálním vlakem, a proto musíme pokaždé do pole s optimálním vlakem zkopírovat celý aktuální vlak od začátku až po aktuální pozici.

Z teoretického hlediska představuje úloha jednoduché procházení stromem všech možných vlaků. Lokomotiva představuje kořen stromu, jeho potomky jsou představovány všemi vagóny, které mohou být zařazeny za lokomotivou. Pro každý uzel *u* kromě kořene tohoto stromu platí, že jeho bezprostředním potomkem může být jen uzel *v* reprezentující vagón, který může být zařazen za vagónem *u* a který se nevyskytuje na cestě z kořene do uzlu *u*. Řešení této úlohy pak není nic jiného než systematické generování právě tohoto stromu se současnou registrací vah cest z kořene do jednotlivých listů.