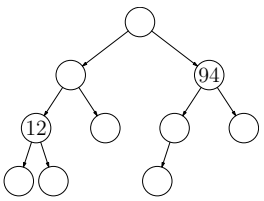
1 (True or False) and Justify [20 bodov]

1. Existuje algoritmus, ktorý ľubovoľné n-prvkové pole, v ktorom je len 47 rôznych hodnôt, usporiada v čase $O(n)$.
2. Rozmiestnenie prvkov v halde nezávisí od poradia, v akom sme ich vkladali.
3. Rozmiestnenie prvkov v binárnom vyhľadávacom strome nezávisí od poradia, v akom sme ich vkladali.
4. O usporiadanom poli s n^2 prvkami vieme v čase $\Theta(\log n)$ zistiť, či obsahuje daný prvok x.
5. Najväčší prvok v binárnom vyhľadávacom strome je aj v pre-order, aj v in-order, aj v post-order zápise posledný.
6. V halde s minimom v koreni pre ľubovoľné prvky x, y platí: ak $x > y$, tak x je aspoň tak hlboko ako y.
7. Každý (aj nevyvážený) binárny vyhľadávací strom s n prvkami má hĺbku $\Omega(\log n)$.
8. Ak v obci s n domami nie sú žiadne dva susedné od seba viac ako 100 m, tak na obec stačí $\lceil n/11 \rceil$ zastávok.

2 Pohľad zvnútra: BST [5+5 bodov]



Na obrázku je binárny vyhľadávací strom obsahujúci 10 navzájom rôznych prvkov.
a) Vyfarbite všetky vrcholy, kde sa môže nachádzať prvok s hodnotou 47.
b) V strome nie je prvok s hodnotou 42. Dokreslite všetky možnosti, kde môže pribudnúť nový vrchol, keď prvok 42 do tohto stromu vložíme.

3 Pohľad zvnútra: Podobné k sebe [15 bodov]

V poli $A[0..n-1]$ máme prvky neznámeho typu. Máme funkciu, ktorá nám pre dvojicu prvkov v konštantnom čase povie, či sa podobajú. Podobnosť je symetrická, ale nie nutne tranzitívna. Úlohou je preusporiadať toto pole tak, aby platilo, že každé dva po sebe idúce prvky sa na seba podobajú (alebo zistiť, že sa to nedá).
Dokážte, že *každý* algoritmus riešiaci túto úlohu musí mať časovú zložitosť $\Omega(n \log n)$.

4 Pohľad zvnútra: multiset [15 bodov]

Dostali ste knižnicu, v ktorej je implementovaný `ordered_set`. To je dátová štruktúra, v ktorej viete reprezentovať usporiadanú množinu: efektívne robiť operácie `insert(x)` (vlož `x` ak tam ešte nie je), `erase` (vymaž `x` ak tam je) a `lower_bound(x)` (nájdi najmenší prvok $\geq$ `x`).
Pomocou tejto dátovej štruktúry implementujte `multiset`: dátovú štruktúru pre množinu s násobnými prvkami. Tá by mala podporovať (aspoň) operácie `insert(x)`, `erase_one(x)`, `erase_all(x)` (vymaž jeden výskyt `x`, resp. vymaž všetky výskyty `x`) a `count(x)` (zisti, koľkokrát obsahuje `x`).
Ideálne riešenie okrem volaní metód `ordered_set`u spraví pri každej operácii len $O(1)$ iných krokov výpočtu. Ak takéto riešenie neviete nájsť, nájdite najlepšie, aké viete – prinajhoršom nejako implementujte `multiset` úplne bez použitia `ordered_set`u.

5 Pohľad zvonka: stíhanie mŕtvych čiar [(7+7+6) bodov + bonus]

Janku čaká n povinností. O každej z nich vie čas t_i (v sekundách), ktorý jej zaberie, a deadline d_i (v sekundách odteraz, $d_i \geq t_i$), dokedy to musí byť hotové. Činnosť na povinnostiach môže ľubovoľne prerušovať – ak má povinnosť, ktorá trvá 100 sekúnd, môže spraviť 47.7 sekundy teraz a zvyšných 52.3 neskôr.
a) Navrhnite pažravý algoritmus, ktorý zistí, či vie Janka stihnúť všetky povinnosti. Odhadnite jeho časovú zložitosť.
b) Dokážte správnosť vášho pažravého algoritmu (napr.: „Určite nič nepokazím, ak ... lebo ...“)
c) Ťažšia úloha: Každá povinnosť má aj čas začiatku $z_i \leq d_i - t_i$, odkedy sa vôbec dá na nej pracovať.
Upravte algoritmus z časti a) tak, aby riešil aj túto úlohu. (6 bodov za čokoľvek polynomiálne, bonus za efektívne)

6 Pohľad zvonka: problém vreca [4 × 5 bodov + bonus]

Každý, kto už skúšal narvať kopu vecí do batohu či kufra, vie, že nezáleží len na našej nosnosti, ale aj na objeme vecí. Náš zlodej má pred sebou n vecí. Každá vec má celočíselný objem v_i (v decilitroch), celočíselnú hmotnosť m_i (v stovkách gramov) a cenu c_i. Zlodej má so sebou vrece, do ktorého sa zmestia veci s celkovým objemom nanajvýš V. Zlodej v ňom zvláda odniesť veci s celkovou hmotnosťou nanajvýš M. Nájdite najdrahšiu množinu vecí, ktoré zvláda odniesť.
Príklad: pre $V = 700$, $M = 650$, $n = 3$ a veci s $(v_i, m_i, c_i) = (500, 20, 40), (100, 35, 42), (200, 430, 47)$ vezme veci 2 a 3.

a) Napíšte (ako pseudokód alebo kus programu) rekurzívny algoritmus skúšajúci všetky možnosti ako vybrať veci do vreca. (Začne napr. tým, že pre n-tú vec postupne rekurzívne vyskúša možnosti „vezmem ju“ a „nevezmem ju“.)
b) Pridajte do predchádzajúceho algoritmu memoizáciu tak, aby vznikol algoritmus s časovou zložitosťou polynomiálnou od počtu vecí n. Odhadnite jeho časovú zložitosť.
c) Uveďte ekvivalentný algoritmus, ktorý túto úlohu rieši pomocou dynamického programovania.
d) Dokážte alebo vyvráťte správnosť pažravého algoritmu:
Pre každú vec vypočítame jej výhodnosť $\varphi_i = c_i \,/\, \max(v_i/V, m_i/M)$ a následne do vreca po jednej vkladáme veci, pričom zakaždým spomedzi vecí, ktoré ešte pripadajú do úvahy, vyberieme tú s najväčšou výhodnosťou.
e) Za bonusové body: Ktorý z algoritmov z častí b) a c) by ste si v praxi vybrali a prečo?