



Stof

- slides college 4
- boek 2.4.1, 2.4.2
- boek 2.5

Opgaven

1. We bekijken de array-representatie van een heap met 15 items. Wat is het rijtje van indexen behorend bij een preorder traversal van de heap? En voor inorder en postorder?
2. Geef een voorbeeld waaruit blijkt dat een preorder traversal van een min-heap niet perse de keys in niet-dalende volgorde geeft.
3. Geef een voorbeeld waaruit blijkt dat een postorder traversal van een min-heap niet perse de keys in niet-stijgende volgorde geeft.
4. Geef een voorbeeld waaruit blijkt dat de bottom-up heap constructie een andere min-heap levert dan wanneer we een-voor-een toevoegen.
5. We definiëren en bekijken de 3-min-heaps. (Voor de terminologie: een binaire boom is een 2-boom.)
 - (a) Wat is een 3-boom?
 - (b) Wat is een complete 3-boom?
 - (c) Wat is de hoogte van een complete 3-boom, in termen van grote-Oh?
 - (d) Wat is de 3-min-heap eigenschap?
 - (e) Beschrijf de procedure `3downMinHeap`.
 - (f) Beschrijf de procedure `3upMinHeap`.
 - (g) In het algemeen kun je denken aan een k -heap. Met n items vinden we voor $k = 1$ de sliert-heap, min of meer een lijst, en voor $k = n$ de heap van slechts diepte 1. Welke k levert de beste (dus kleinste) tijdscomplexiteit voor de bubbel-operaties?
6. Hoe ziet een worst-case input-rijtje ter lengte n voor een min-heap eruit? Laat zien dat de tijdscomplexiteit van zo'n rijtje in $\Omega(n \log(n))$ is.

7. Geef per stap aan hoe de priority queue eruit ziet bij het uitvoeren van de volgende operaties - we beginnen met een lege priority queue:

`insertItem(5, a), insertItem(4, b), insertItem(7, i), insertItem(1, d), removeMin(),
insertItem(3, j), insertItem(6, l), removeMin(), removeMin(),
insertItem(8, g), removeMin()`

8. Hoe kunnen we het ADT van de gewone queue implementeren, gebruikmakend van een priority queue en daarbij een integer variabele?
9. Pas stap voor stap selection-sort toe op de volgende input-rij:

[22, 15, 36, 44, 10, 3, 9, 13, 29, 25]

10. Pas stap voor stap insertion-sort toe op dezelfde input-rij.
11. Geef een voorbeeld van een worst-case rijtje met n elementen voor insertion sort. Laat zien dat de tijdscomplexiteit van insertion sort voor zo'n rijtje in $\Omega(n^2)$ is.
12. We bekijken de input-rij

[3, 9, 8, 6, 2, 4, 1]

- (a) Sorteert de rij met heap-sort als volgt:

- bouw met de bottom-up heap constructie een max-heap,
- geef daarvan de array-representatie,
- geef vervolgens (in de array-representatie of met plaatjes) stap voor stap de sorteerstappen

- (b) Sorteert de rij met priority-queue sort waarbij de priority-queue geïmplementeerd wordt door een min-heap:

- bouw met de bottom-up heap constructie een min-heap,
- geef daarvan de array-representatie,
- geef vervolgens stap voor stap de sorteerstappen van de vorm: verwijder de kleinste van de heap, stop die achteraan in het rijtje dat we als output gaan geven.

13. Wat is de worst-case tijdscomplexiteit voor het toevoegen van n items (met key en element) aan een initieel lege ongesorteerde lijst?
14. We beginnen met een initieel lege hash-table van grootte 11, en we gebruiken de hash functie $h(i) = (3i + 5) \bmod 11$. Voeg (in deze volgorde) de volgende keys toe:

12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5

waarbij collision wordt opgelost met chaining.

15. Wat krijg je als resultaat van de vorige opgave, met als enige verschil dat nu collision wordt opgelost met lineair probing?
16. We bekijken hashing met lineair probing. Een mogelijkheid (zie de slides) is om verwijderde elementen te representeren met een speciale marker (bijvoorbeeld 'available'). Het kan ook anders: je kan ook de array herarrangeren, zo dat het lijkt alsof het verwijderde element er nooit in heeft bestaan. Hoe zie de operatie voor verwijderen er dan uit?
17. Waarom is een hash table niet geschikt om een geordend dictionary te implementeren?

18. (Dit is ongeveer opgaven R-2.19 – R-2.22.)

We beginnen met een lege hash table ter grootte 11 en voegen toe:

12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5

(in deze volgorde), met hash functie $h(k) = (2 \cdot k + 5) \bmod 11$ waarbij collision wordt opgelost op de volgende manieren:

- (a) met chaining,
 - (b) met linear probing,
 - (c) met quadratic probing (dit gaat op een gegeven moment mis omdat er geen lege plek wordt gevonden),
 - (d) met double hashing, met als tweede hash functie $h'(k) = 7 - (k \bmod 7)$.
19. (Dit is ongeveer opgave R-2.23.)
Geef de pseudo-code voor het toevoegen van een item aan een hash table waarbij we quadratic probing gebruiken om collisions op te lossen. We gaan ervan uit dat verwijderde items vervangen zijn door een speciaal object A (van het juiste type).