Algorithmen & Datenstrukturen

Übungsstunde 5

Michelle Honegger - CAB G59 - 20.10.2024

Miniquiz 5

- Passwort: **mergesort10**
- 8 Fragen
- 8 Minuten
- 0.125 Punkte pro richtige Antwort

Programm

- Miniquiz
- Rückmeldung Serie 3
- Besprechung Aufgaben
- Theory Recap
- Prüfungsaufgabe

Rückmeldung Serie 3

- Verwendung von Hint/Theorem welches gegeben wird soll gekennzeichnet werden
- Irgendwo muss stehen was wir beweisen
- Bei Codesnippet Laufzeitabschätzung am besten mit Summen arbeiten
- Rot ist meine Farbe 😖

Besprechung Miniquiz

$5e^{2\ln(n)} = \Theta(e^{10\ln(n)})$

- Wahr
- Falsch

Suppose you have some algorithm A and let an increasing function $T(n)$ be its runtime on an input of size n. Assume you know that $T(n) \leq T(n-1) + 100n$ and that $T(1) = 10$. Then $T(n) \leq O(n^2)$.

- Wahr
- Falsch

Every comparison-based sorting algorithm needs to perform at least $\Omega(n \log(n))$ swaps.

- Wahr
- Falsch

Suppose we apply *insertion sort* to the array $A_n = [n, n-1, n-2, \ldots, 3, 2, 1]$.

(E.g., $A_5 = [5, 4, 3, 2, 1]$)

Let $s(n)$ be the number of swap operations that are performed before the array is fully sorted (in ascending order).

Wählen Sie eine Antwort:

- a. $s(n) \geq \Omega(n^2)$
- b. $s(n) \leq O(n)$

Which of the following sorting algorithms have the invariant that: after j steps, the first j elements are sorted?

(A step refers to one iteration of the outer for loop for each sorting algorithm).

Wählen Sie eine oder mehrere Antworten:

- [ ] a. Bubble sort
- [ ] b. Selection sort
- [ ] c. Insertion sort
- [ ] d. Merge sort

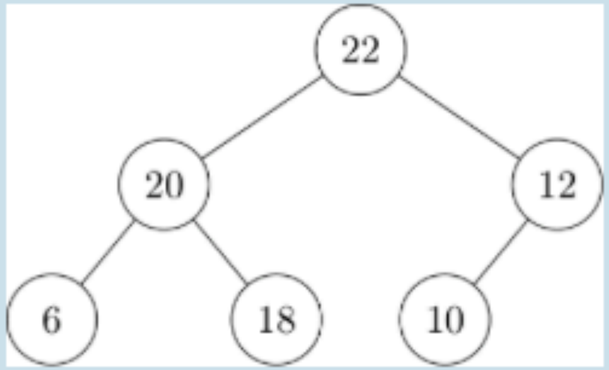
Asymptotically (in Θ-notation), *merge sort*, *heap sort*, and deterministic *quick sort* all have the same worst-case runtime.

- Wahr
- Falsch

Suppose all keys in a max-heap are unique. Then the node with the minimum key is always the left-most leaf.

- Wahr
- Falsch

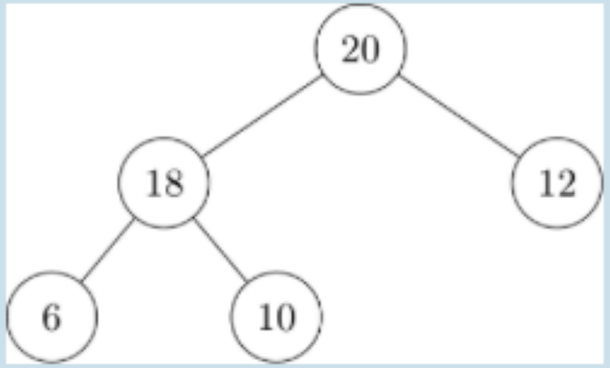
Consider the following max-heap,



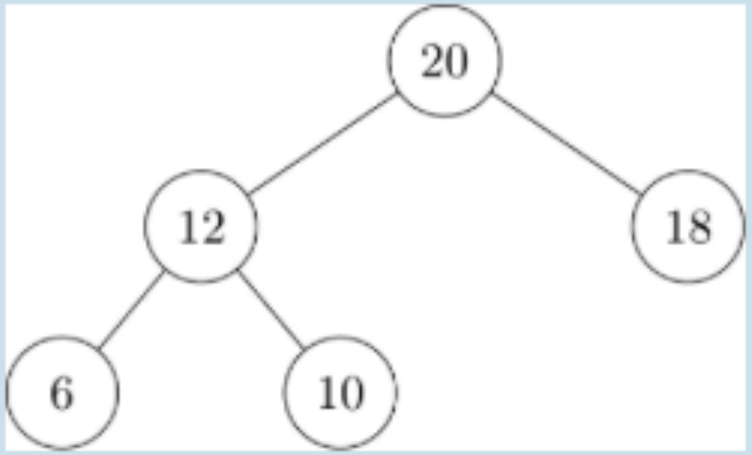
After executing one `ExtractMax` operation, what is the correct max-heap?

Wählen Sie eine Antwort:

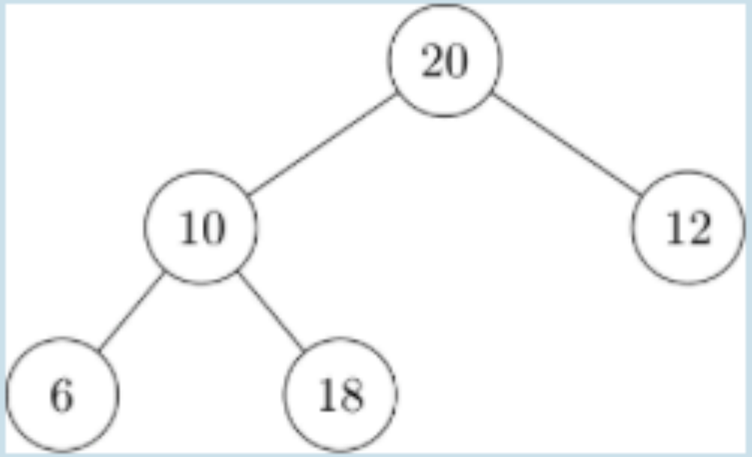
a.



b.



c.



Theory Recap

Inhalt

- Sortieralgorithmen Teil 2
- Abstrakte Datentypen
- Datenstrukturen

Quicksort

- Pivotelement p wählen, korrekte Position von p finden, durch einordnen (grösser?, kleiner?) von restlichen Elementen
- in-place
- $O(n \cdot \log(n))$
- rekursiv implementiert

```
int partition(int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low-1);
    for (int j=low; j<high; j++){
        if (arr[j] < pivot){
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i+1];
    arr[i+1] = arr[high];
    arr[high] = temp;

    return i+1;
}

void sort(int arr[], int low, int high){
    if (low < high){
        int pi = partition(arr, low, high);
        sort(arr, low, pi-1);
        sort(arr, pi+1, high);
    }
}
```

Heap-Sort

- Siehe Video: https://youtu.be/Q1yi1eaqN7I?si=390cnFE6lLfS8Qc6
- in-place
- $O(n \cdot \log(n))$
- rekursiv implementiert

```java
public static int[] heapSort(int[] arr) {
    for(int i = arr.length/2; i >= 0; i--) {
        restoreHeapCondition(arr, i, arr.length);
    }
    for(int i = arr.length-1; i >= 1; i--) {
        int tmp = arr[0];
        arr[0] = arr[i];
        arr[i] = tmp;
        restoreHeapCondition(arr, 0, i);
    }

    return arr;
}

public static int[] restoreHeapCondition(int[] arr, int i, int n) {
    int j;
    while(2*i < n) {
        j = 2*i;
        if(j+1<n) {
            if(arr[j] < arr[j+1]) {
                j++;
            }
        }
        if(arr[i] >= arr[j]) {
            break;
        }
        int tmp = arr[i];
        arr[i] = arr[j];
        arr[j] = tmp;
        i = j;
    }
    return arr;
}
```

Abstrakte Datentypen

- **Heap**: Schneller Zugriff auf Maximum/Minimum
- **Liste**: feste Reihenfolge der Objekte
  - **Array**: wird benutzt, wenn Anzahl Elemente bekannt, viele nützliche Funktionen in Java
  - **(doubly) - Linked List**: Elemente bestehen aus Schlüssel und Pointer, der auf das nächste bzw. nächste und vorherige Element zeigt -> Spezielle Art von Liste
    - .insert(x), .poll(), .get(i)

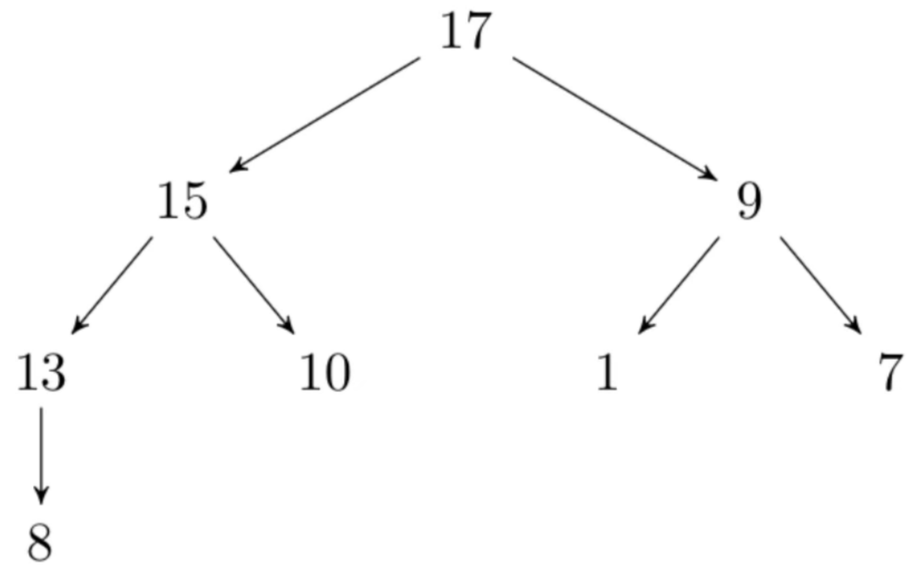
Abstrakte Datentypen

- **Stack**: LIFO mit Operationen push(x), pop(), top() - mit linked-list implementiert
- **Queue**: FIFO mit enqueue(x), dequeue() - mit doubly-linked-list implementiert
- **Priority Queue**: wird mit Heap implementiert

Prüfungsaufgaben

i) Draw a Max-Heap that contains the keys $3, 6, 1, 5, 7, 2$.

ii) Draw the Max-Heap obtained from the following Max-Heap by performing the operation DELETE−MAX once.



Consider the following recursive function that takes as an input a natural number m that is a power of two (that is, $m = 2^k$ for some nonnegative integer k).

Algorithm 3 $g(m)$

```
if m > 1 then
    g(m/2)
    g(m/2)
    g(m/2)
    g(m/2)
    for i = 1, ..., m^2 do
        f()
else
    f()
```

Let $T(m)$ be the number of calls of the function f in $g(m)$.

i) Give a recursive formula for $T(m)$.

ii) Write $T(m)$ in $\mathcal{O}$-notation in terms of m (as tight and simplified as possible).

Exercise Recommendations

Aufgabenblatt 5

- 5.1 Bonus
- 5.2 skippen wenn sonst schon viel zu tun
- 5.3 Bonus
- 5.4 Bonus
- 5.5 Kann man sich mal durchdenken und allenfalls implementieren

Leetcode of the Week

- 143. Reorder List
- https://leetcode.com/problems/reorder-list/

Peergrading

- Aufgabe 4.4
- Grading Scheme in Lösungen beachten