Exam for „Algorithms I“

WS 2022/23 / February 23, 2023

First name: ____________________

Second name: ____________________

Matriculation number: ____________________

task	max. points	points achieved
1	6	
2	7	
3	6	
4	8	
5	8	
6	5	
Σ	40	

Important information

- Duration of the exam: **60 minutes**.
- Tools: You are allowed to use a handwritten sheet of paper (size DIN A4). Both sides of the sheet of paper can be handwritten.
- Write with an indelible pen. Do not write in red paint.
- Check the exam you have been given for completeness: **6 tasks** on 8 pages.
- Enter your name and matriculation number in the appropriate fields on each sheet.
- Write your solutions in the spaces provided. If there is not enough space in a field, use the back of the corresponding sheet and indicate this on the front. If there is still not enough space, you can ask the supervisor for additional sheets of paper.

Task 1. (6 Points)
Which of the following statements hold (f and g are arbitrary functions on $\mathbb{N}$)?

1. $(n-1)^2 \in o(n^2)$
2. $\mathcal{O}(n) = \mathcal{O}(n^2)$
3. $o(n) = o(2n)$
4. $f \in \Theta(g)$ if and only if $g \in \Theta(f)$
5. If $f \in o(g)$ then also $f \in \mathcal{O}(g)$.
6. $2^{\sqrt{n}} \in \mathcal{O}(n^{\log n})$

You do not have to prove your answers.

Solution:

1. not correct
2. not correct
3. correct
4. correct
5. correct
6. not correct

Task 2. (7 Points)

Consider the following recursive sorting algorithm, where A is an array of n integers. The function swap(A, i, j) swaps the elements at positions i and j in the array A.

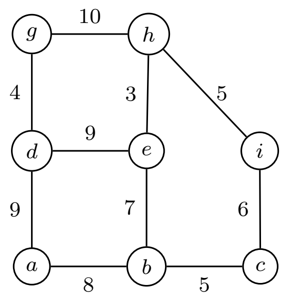
```
1.  function slow-sort(A[1...n] : array of integers)
2.  if n = 1 then return(A)
3.  else
4.      slow-sort(A[2...n])
5.      if A[1] > A[2] then
6.          swap(A, 1, 2)
7.          slow-sort(A[2...n])
8.      endif
9.  return(A)
10. endif
11. endfunction
```

1. Explain, why this is a correct sorting algorithm.
2. Give a recursive equation for the worst-case running time of the algorithm.
3. Show that the worst-case running time of the algorithm is $\Omega(2^n)$.

Solution:

1. By induction, after slow-sort($A[2 \ldots n]$) in line 4, the subarray of A from position 2 to position n is sorted. In particular, the smallest element in the array is at position 1 or 2. If $A[1] \leq A[2]$ then A is sorted and A is correctly returned in line 9. If $A[1] > A[2]$ then after swap($A, 1, 2$), the smallest array element is at position 1. Then, after slow-sort($A[2 \ldots n]$) in line 7, the array is sorted and returned in line 9.
2. $T(n) = 2 \cdot T(n-1) + c$ for a constant c.
3. We have $T(n) \geq 2 \cdot T(n-1)$ for $n \geq 2$. By induction, we show that $T(n) \geq 2^{n-1}$ for all $n \geq 1$: For $n = 1$ we have $T(n) = 1 \geq 2^{1-1}$. For $n > 1$ we obtain inductively $T(n) \geq 2 \cdot T(n-1) \geq 2 \cdot 2^{n-2} = 2^{n-1}$.

Task 3. (6 Points)
Compute a spanning subtree of maximal weight using Kruskal's algorithm for the following graph.

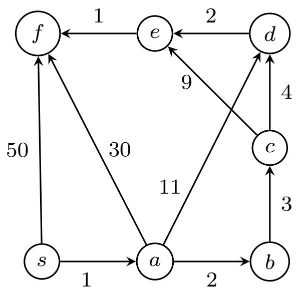


Show the edge selected in each step or indicate if no edge is selected in a step.

Solution: Let T be current set of selected edges. Initially we have $T = \emptyset$

1. Select edge $\{g, h\}$: $T = \{\{g, h\}\}$
2. Select edge $\{a, d\}$: $T = \{\{g, h\}, \{a, d\}\}$
3. Select edge $\{d, e\}$: $T = \{\{g, h\}, \{a, d\}, \{d, e\}\}$
4. Select edge $\{a, b\}$: $T = \{\{g, h\}, \{a, d\}, \{d, e\}, \{a, b\}\}$
5. Edge $\{b, e\}$ is not selected.
6. Select edge $\{c, i\}$: $T = \{\{g, h\}, \{a, d\}, \{d, e\}, \{a, b\}, \{c, i\}\}$
7. Select edge $\{b, c\}$: $T = \{\{g, h\}, \{a, d\}, \{d, e\}, \{a, b\}, \{c, i\}, \{b, c\}\}$
8. Select edge $\{h, i\}$: $T = \{\{g, h\}, \{a, d\}, \{d, e\}, \{a, b\}, \{c, i\}, \{b, c\}, \{h, i\}\}$
9. Edge $\{d, g\}$ is not selected.
10. Edge $\{e, h\}$ is not selected.

Task 4. (8 Points)
Use Dijkstra's algorithm to compute all shortest paths starting at node s in the graph below. Show the values of the program variables B, R, U, p, D after each iteration of the main **while**-loop of Dijkstra's algorithm.



Solution:

1. After iteration 1:
   $B = \{s\}$, $R = \{a, f\}$, $U = \{b, c, d, e\}$
   $p[a] = s$, $p[f] = s$, $p[x] =$ nil for other nodes x
   $D[s] = 0$, $D[a] = 1$, $D[f] = 50$, $D[x] = \infty$ for other nodes x

2. After iteration 2:
   $B = \{s, a\}$, $R = \{b, d, f\}$, $U = \{c, e\}$
   $p[a] = s$, $p[f] = a$, $p[b] = a$, $p[d] = a$, $p[s] = p[c] = p[e] =$ nil
   $D[s] = 0$, $D[a] = 1$, $D[f] = 31$, $D[b] = 3$, $D[d] = 12$, $D[c] = D[e] = \infty$

3. After iteration 3:
   $B = \{s, a, b\}$, $R = \{c, d, f\}$, $U = \{e\}$
   $p[a] = s$, $p[f] = a$, $p[b] = a$, $p[c] = b$, $p[d] = a$, $p[s] = p[e] =$ nil
   $D[s] = 0$, $D[a] = 1$, $D[f] = 31$, $D[b] = 3$, $D[c] = 6$, $D[d] = 12$, $D[e] = \infty$

4. After iteration 4:
   $B = \{s, a, b, c\}$, $R = \{d, e, f\}$, $U = \emptyset$
   $p[a] = s$, $p[f] = a$, $p[b] = a$, $p[c] = b$, $p[d] = c$, $p[e] = c$, $p[s] =$ nil
   $D[s] = 0$, $D[a] = 1$, $D[f] = 31$, $D[b] = 3$, $D[c] = 6$, $D[d] = 10$, $D[e] = 15$

5. After iteration 5:
   $B = \{s, a, b, c, d\}$, $R = \{e, f\}$, $U = \emptyset$
   $p[a] = s$, $p[f] = a$, $p[b] = a$, $p[c] = b$, $p[d] = c$, $p[e] = d$, $p[s] =$ nil
   $D[s] = 0$, $D[a] = 1$, $D[f] = 31$, $D[b] = 3$, $D[c] = 6$, $D[d] = 10$, $D[e] = 12$,

6. After iteration 6:
   $B = \{s, a, b, c, d, e\}$, $R = \{f\}$, $U = \emptyset$
   $p[a] = s$, $p[f] = e$, $p[b] = a$, $p[c] = b$, $p[d] = c$, $p[e] = d$, $p[s] =$ nil
   $D[s] = 0$, $D[a] = 1$, $D[f] = 13$, $D[b] = 3$, $D[c] = 6$, $D[d] = 10$, $D[e] = 12$,

7. After iteration 7:
   $B = \{s, a, b, c, d, e, f\}$, $R = \emptyset$, $U = \emptyset$
   $p[a] = s$, $p[f] = e$, $p[b] = a$, $p[c] = b$, $p[d] = c$, $p[e] = d$, $p[s] = \text{nil}$
   $D[s] = 0$, $D[a] = 1$, $D[f] = 13$, $D[b] = 3$, $D[c] = 6$, $D[d] = 10$, $D[e] = 12$,

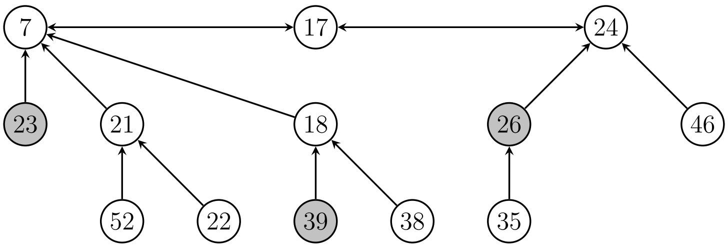
In table form (Step 0 is optional):

Node	s	a	b	c	d	e	f
Step 0	0	∞	∞	∞	∞	∞	∞
$p(x)$	nil	nil	nil	nil	nil	nil	nil
Step 1	**0**	1	∞	∞	∞	∞	50
$p(x)$	nil	s	nil	nil	nil	nil	s
Step 2	**0**	**1**	3	∞	12	∞	31
$p(x)$	nil	s	a	nil	a	nil	a
Step 3	**0**	**1**	**3**	6	12	∞	31
$p(x)$	nil	s	a	b	a	nil	a
Step 4	**0**	**1**	**3**	**6**	10	15	31
$p(x)$	nil	s	a	b	c	c	a
Step 5	**0**	**1**	**3**	**6**	**10**	12	31
$p(x)$	nil	s	a	b	c	d	a
Step 6	**0**	**1**	**3**	**6**	**10**	**12**	13
$p(x)$	nil	s	a	b	c	d	e
Step 7	**0**	**1**	**3**	**6**	**10**	**12**	**13**
$p(x)$	nil	s	a	b	c	d	e

The distances are **bold** for tree nodes B, normal for boundary nodes R and ∞ for unknown nodes U.

Task 5. (8 Points)

The following Fibonacci heap is given:

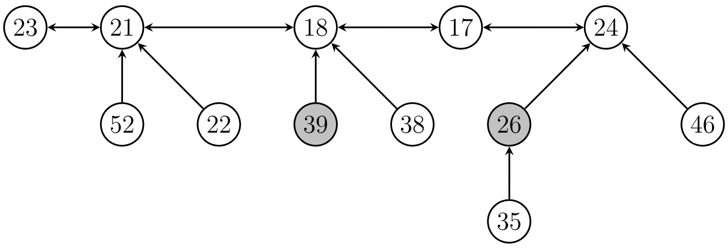


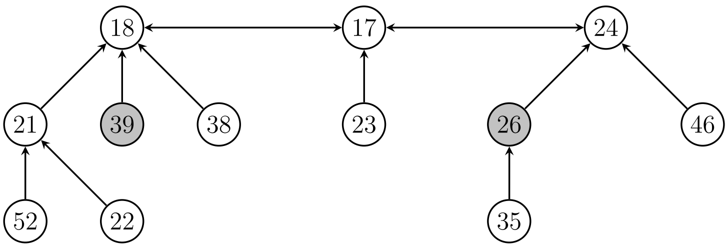
Perform the following operations in that order:

1. delete-min
2. decrease-key(node with key 52, 9)
3. decrease-key(node with key 35, 16)
4. delete-min

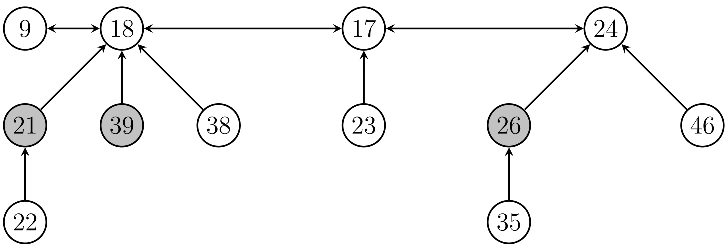
Solution:

1. delete-min (in two steps)



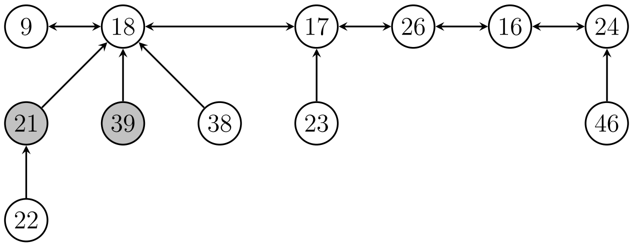


2. decrease-key(node with key 52, 9)

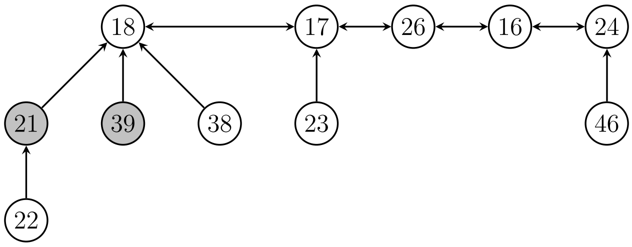


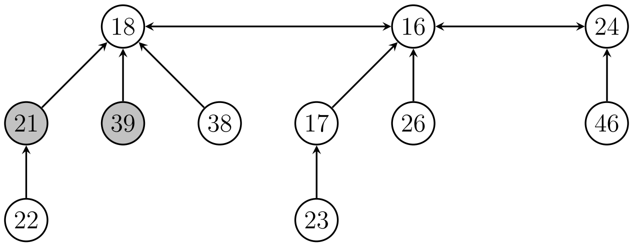
Solution:

3. decrease-key(node with key 35, 16)



4. delete-min (in two steps)





Task 6. (5 Points)

Four matrices A_1, A_2, A_3, A_4 are given and the goal is to compute the product $A_1A_2A_3A_4$. The dimensions of the matrices are as follows:

- A_1: 10×10
- A_2: 10×1
- A_3: 1×6
- A_4: 6×5

Find the optimal bracketing so that the number of scalar multiplications is minimized.

Solution:

- cost for A_1A_2 (10×1): 100
- cost for A_2A_3 (10×6): 60
- cost for A_3A_4 (1×5): 30
- optimal cost for $A_1A_2A_3$ (10×6): $\min(100 + 60, 60 + 600) = 160$, which is realized by $(A_1A_2)A_3$
- optimal cost for $A_2A_3A_4$ (10×5): $\min(60 + 300, 30 + 50) = 80$, which is realized by $A_2(A_3A_4)$
- optimal cost for $A_1A_2A_3A_4$ (10×5): $\min(160+300, 100+30+50, 80+500) = 180$, which is realized by $(A_1A_2)(A_3A_4)$.