Exercises for Algorithms and Data Structures Lecture 10

Question 1.
Design a greedy algorithm for the allocation problem. Does your greedy algorithm always give an optimal solution?

Question 2. (exercise 1.2.2 and 9.1.5 from the book by Levitin.)
New World puzzle There are four people trying to cross an unstable bridge; they all start on the same side. You have 17 minutes to get them all across. It's night, and they have a flashlight. At most two people cross the bridge at the same time. When people cross the bridge, whether it's one or two people, they should have the flashlight with them. The flashlight must be carried back and forth;It can't be thrown. Person 1 needs one minute to cross the bridge, person 2 two minutes, person 3 five minutes and person 4 ten minutes. If two people cross together, they will walk at the speed of the slowest walker..

a. Think of a solution for the example above. **b.** Now think of the general case, where we have $n > 1$ people, with times to cross being $t_1, t_2, \ldots, t_n$. All other conditions stay the same you can assume that $t_1 \leq t_2 \leq \cdots \leq t_n$. Come up with a greedy algorithm for this problem and calculate the total time it will take to cross the bridge if you use that algorithm. Check whether the greedy algorithm always produces an optimal solution. If yes, show this. If not, provide an instance with the smallest amount of people where it does not work.

Question 3. (exercise 9.1.8 from the book by Levitin.)

Bachet's weightproblem find an optimal row of n weights $\{w_1, w_2, \ldots, w_n\}$ Which makes it possible with a balance to weigh any object with the weight of an integer between 1 and W, for the largest possible number W, assuming

a. the weights can only be placed on the free scale of the balance (i.e. where the object to be weighed is not located). This means that with n positive integers you must make the biggest possible contiguous sequence of numbers bu adding them up in all possible ways. The goal is to find the optimal row to build up 'weights' in a greedy way.

b. (for the enthousiasts) the weights can be placed on both scales. Now you can also subtract, by putting some weights on the left scale of the balance, and others on the right scale. For example, you can have an object with weight 2 using the weights $w_1 = 1$ and $w_2 = 3$ by placing the object with weight w_1 on one scale and the object with weight w_2 on the other.

Question 4. (exercise 9.3.1.a,b,c from the book by Levitin.)

explain in which way (*if any*) Dijkstra's algorithm or the underlying graph needs to be modified to solve the following problems **a.**

decide the shortest paths starting from a single node in a directed weighted graph **b.** decid the shortest pad *between two given nodes* in a directed or undirected weighted graph. (this variation is called the *single-pair shortest-path problem.*)

c. decide the shortet paths *to* a given node from every other node in a a directed or undirected weighted graph. (this variation is called the *single-destination shortest-paths problem.*)

note: for Dijkstra's algorithm, look at the sheets in the lectures and below. In the algorithm on the sheets only the shortest distances are found, but the shortest paths themselves are easily found with a simple change. Save the branch (v^*, v) where the labels are changed (or change the old branch "to" v), if path $[v]$ is changed to path $[v^*]+$ weight(v^*, v). (see the version of Dijkstra's algorithm below.) The path through v^*, with the final branch (v^*, v), is then shorter than the previously found path from s to v through nodes from the original U. when the node v is chosen to be added to U (v now has the lowest path worth and is indicated in the algorithm as v^*), the candidate branch becomes definitive. This branch then is a part of the shortest path from s to v. when the algorithm is done al these branches together form the exact tree existing from all the shortest paths. In this book the node of origin is remembered, not the branches. This comes down to the same thing

Question 5. (Opgave 9.3.2 from the book by Levitin.)
Solve the next two instances of the shortest path problem from node a. Use dijkstra's algorithm like the example on the slides, by labeling the candidate nodes with their candidate distances in the graph, and changing their label if needed. Also design the tree of shortest paths, by marking aside of the choice v^* also marking the branch on the shortest path and connects v^* with the original U

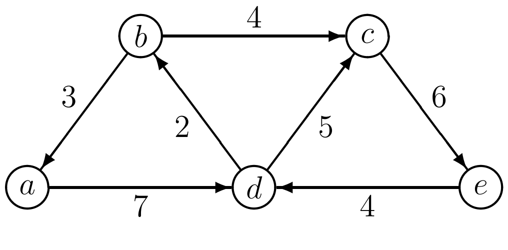
Show the progress of the algorithm with a table like the following:

(for **a.**):

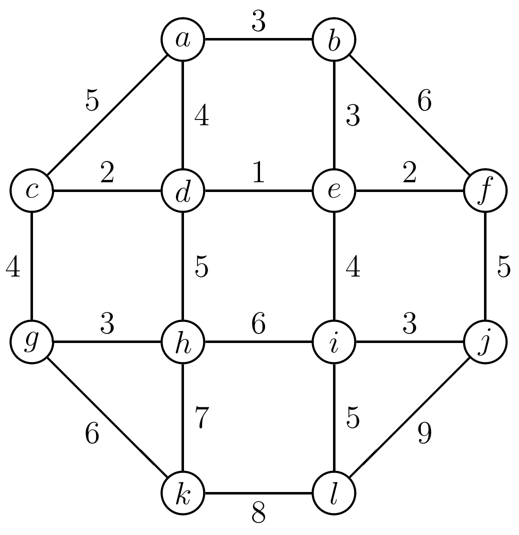
a	b	c	d	e	Actie
0	∞	∞	∞	∞	...
		...			...

Look at the sheets for an example in using the table.

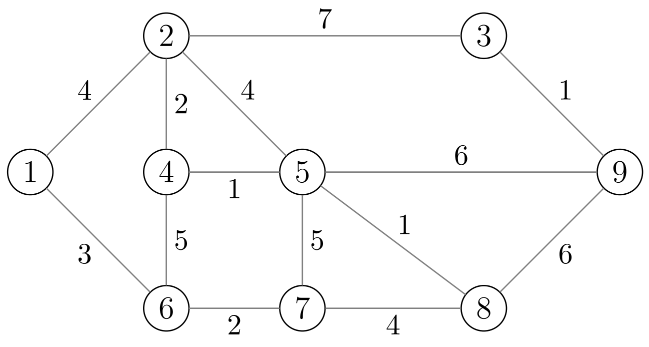
a.



b.



Question 6. use Dijkstra's algorithm on the example graph below. find the (lengths of the) shortest paths from node 1.



Question 7. Give an example case that show that dijkstra's algorithm does not always work for a weighted graph with negative weights. (show a weighted grapsh with at least one negative weight, where Dijkstra's algorithm can not find the shortest path to all nodes from a certain starting node. A graph with 3 nodes is big enough.

Het algoritme van Dijkstra

```
// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop $s$
// uitvoer: array dat de lengtes van de kortste paden vanuit $s$ bevat;
// na afloop is pad[$v$] = de lengte van een kortste pad van $s$ naar $v$
// genereert ook de takken van de kortste paden boom

**for** $v \in V$ **do**
        pad[$v$] := $\infty$;
**od**
pad[$s$] := 0;
$U := \emptyset$;
**while** ( $U \neq V$ ) **do**
        vind knoop $v^* \in V \setminus U$ met pad[$v^*$] minimaal;
        $U := U \cup \{v^*\}$;
        **for** alle knopen $v$ aangrenzend aan $v^*$ **do**
              **if** pad[$v^*$]+ gewicht($v^*, v$) < pad[$v$] **then**
                    pad[$v$] := pad[$v^*$]+ gewicht($v^*, v$);
                    *nieuwe kandidaattak* voor $v$: ($v^*, v$)
              **fi**
        **od**
**od**
```