

Introduction to algorithms and data structures

The IT University of Copenhagen

16. January 2001

This examination assignment consists of 3 exercises with a total of 13 subexercises. Subexercise are given equal weight in the grading. You have 4 hours to answer all 13 subexercises. Remember to write the pagenumber, your name and your “studienummer” on each page of your written answers. The complete assignment consists of 8 numbered pages.

CLR refers to “Introduction to Algorithms” by Cormen, Leiserson and Rivest, 18th print, 1997.

For exercises that ask for effective algorithms, the asymptotic complexity of the specified solution will be taken into account when grading. Exercises asking for time complexity must be answered using O -notation with least possible asymptotic growth.

Exercise 1

This exercise deals with O -notation, binary search and amortisation. In the exercise it is assumed that a table with n entries starts in entry 0 and ends in entry $n - 1$.

a) Let $p1$ be a procedure with the time complexity $O(n^2)$, and let $p2$ be a procedure with the time complexity $O(n^3)$. Let $p3$ be a procedure, which first makes a call to $p1$ and then a call to $p2$ and apart from this nothing else. What is the time complexity of $p3$?

Answer a) $O(n^3)$

Consider the following procedures:

F1(n)

1. $i \leftarrow 1$

F2(n)

1. **while** $n > 1$
2. **do** $n \leftarrow n - 1$

F3(n)

1. **while** $n > 1$
2. **do** $n \leftarrow \lfloor n/2 \rfloor$

F4(n)

1. **while** $n > 1$
2. **do for** $i \leftarrow 1$ **to** n
3. **do** $j \leftarrow 1$

4. $n \leftarrow n - 1$

b) Assume that the procedures above are only called with a positive integer $n > 0$. What is the time complexity expressed in n for each of the four procedures above?

Answer b)

F1: $O(1)$.

F2: $O(n)$.

F3: $O(\lg n)$.

F4: $O(n^2)$.

Let S be a set of n different integers. Let A be an array with n entries representing the integers in S . Assume that the integers in S are represented in A in a sorted order, i.e. $A[i] < A[j]$, if $0 \leq i < j \leq n - 1$. Let $\text{COUNT}(x, y)$ be a procedure, with the integer parameters x and y , where we assume $x < y$. The procedure $\text{COUNT}(x, y)$ returns the number of integers in S which are both greater than x and less than y . Example: If $A = \langle 5, 10, 33, 49, 66, 75, 82, 95, 100 \rangle$, then $\text{COUNT}(47, 85)$ return the number 4.

c) Describe how $\text{COUNT}(x, y)$ can be implemented with a time complexity of $O(\log n)$.

Answer c) We can do a binary search after the smallest number larger than x and the largest number less than y . In this way we find indices i and j , where $0 \leq i \leq j \leq n$. The count of numbers between x and y is then $j - i$.

Let B be an array consisting of n integers. Initially B is initialised such that each entry in the array is either 0 or 1. That is, $B[i] = 0$ or $B[i] = 1$ for any entry i . Consider the following procedure:

$\text{AMOR1}(i)$

1. **if** $0 \leq i \leq n - 1$ and $B[i] = 1$
2. **then** $B[i] = 0$
3. $\text{AMOR}(i + 1)$

Example: Let $B = \langle 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1 \rangle$. That is $B[0] = 1$, $B[1] = 0$, $B[2] = 1$, $B[3] = 1$, $B[4] = 0$. After a call of AMOR1(2), B is updated such that $B = \langle 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1 \rangle$.

d) Assume that B is initialised as described above and that B is solely updated through a sequence of calls of AMOR1. What is the time complexity of one such call of AMOR1? What is the complexity of n such calls of AMOR1? Justify both answers.

Answer d) A single call to Amor1 can at most take $O(n)$ time if all the entries in B are 1. n calls to Amor1 can at most take $O(n)$ time, since each entry in B at most are changed once.

Let C be an array of n integers. Initially all entries of C contains 1. That is, $C[i] = 1$ for any i . Consider the following procedure:

AMOR2(i, j)

1. **if** $0 \leq i \leq n - 1$ and $0 \leq j \leq n - 1$ and $C[i] \geq C[j] > 0$
2. **then for** $k \leftarrow 1$ **to** $C[j]$
3. **do** $C[i] \leftarrow C[i] + 1$
4. $C[j] \leftarrow 0$

Example: Assume that C has been updated through a sequence of calls of AMOR2 and $C = \langle x, y, 0, z, 0, 0, w \rangle$, where $x, y, z, w > 0$. That is $C[0] = x$ and $C[1] = y$. Assume $x > y$. Then, after a call of AMOR2(0,1), $C = \langle v, 0, 0, z, 0, 0, w \rangle$, where $v = x + y$. That is, the least integer y has been added to the largest integer x which has taken the time $O(y)$.

e) Assume that C is initialised as described above, and is solely updated by a sequence of calls to AMOR2. What is the time complexity of a call of AMOR2? What is the complexity of n calls of AMOR2? Justify both answers.

Answer e) A single call to Amor2(i, j) takes $O(C[j]) = O(n)$ time at the most. Consider the i 'th entry as a box with $C[i]$ elements. Amor2(i, j) then combines box i and j by moving the contents of the smallest box i over in the largest box j . The new box is now at least twice as big as he smallest box. Of course an element is moved no more than $\lg n$ times. The complexity of n calls to Amor2 are therefore at most $O(n \lg n)$.

1 Exercise 2

This exercise deals with directed, weighted graphs and algorithms for finding shortest paths in such graphs.

In all subexercises we assume that the input graph is given as an adjacency-list representation, cf. CLR pp 465-466. We also assume that edge weights are positive integers. For the rest of this exercise we let n denote the number of nodes in the input graph, while m denotes the number of edges.

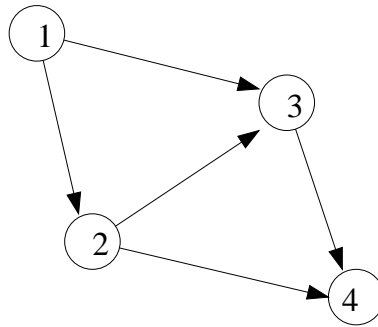


Figure 1: Graph with weights.

a) Give the adjacency-list representation for the graph in figure 1.

Answer a) See figure 2.

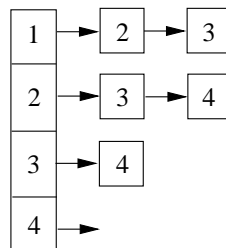


Figure 2: Adjacency-list representation of figure 1

b) Let s and t be two nodes in the directed graph G . Describe an effective algorithm for finding shortest path from a given node s to a given node t in G . Give the time complexity of the algorithm provided that m is $O(n)$.

Answer b) We can use Dijkstras algorithm. The time complexity are $O(m \lg n) = O(n \lg n)$.

c) Describe an algorithm, that finds the length of a shortest path from s to t , that goes through a given node w . That is, the length of the shortest path from s to t , where w is on the path. Give the time complexity of your solution.

Answer c) Find the shortest path from s to w and from w to t and add the lengths. This can be done by two calls to Dijkstra and therefore takes time $O(n \lg n)$.

Consider a rooted, binary tree T with the root r . For an edge (u, v) in T , v is a child of u . To each edge (u, v) in the tree T we associate a graph $G(u, v)$. These graphs constitute the components of a common, large graph $G = (V, E)$ with n nodes, which we call the component-graph. In each subgraph $G(u, v)$ there are two special nodes $s(u), t(v) \in V$ that can be in common with other subgraphs. We assume that there is a directed path from $s(u)$ to $t(v)$. For each pair of edges (u, v) og (v, w) in T , the node $t(v)$ equals the node $s(v)$ in $G(u, v)$ and $G(v, w)$ respectively. No other nodes in the subgraphs are in common with other subgraphs.

Assume that the adjacency-list representation of G also contains information of which subgraph an edge belongs to.

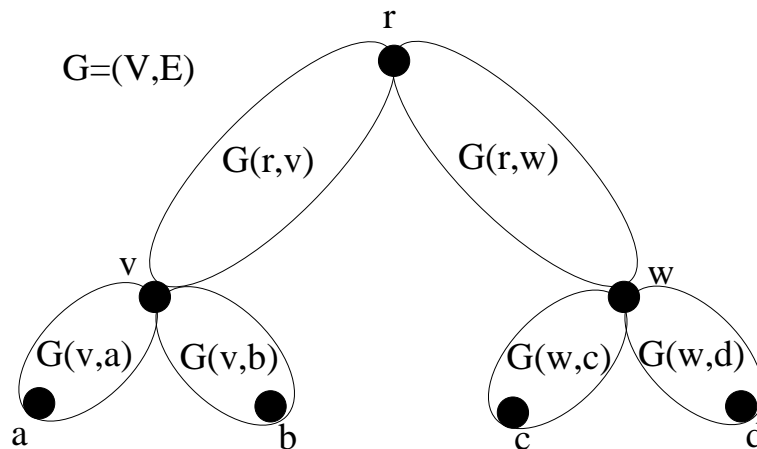


Figure 3: Example of component-graph

d) Assume that all subgraphs $G(u, v)$ have at most $O(\log n)$ nodes and edges. Describe an algorithm that finds the shortest path from $s(r)$ to all other nodes in G in time $O(n \log \log n)$.

Answer d) For each component $G(u, v)$ we calculate the shortest paths from $s(u)$ to all other nodes in $G(u, v)$. The components are calculated downwards in order to calculate the shortest paths in G correctly. The number of components are $O(n/\log n)$ and the number of nodes in each component are $O(\log n)$. Therefore the time complexity are $O(\frac{n}{\log n}(\log n \cdot \log \log n)) = O(n \log \log n)$.

Exercise 3

a) Illustrate the action of $\text{HEAPIFY}(A, 1)$ for the heap A in figure 4. Do it in the same way as in figure 7.2 in CLR page 143.

Answer a) First 7 and 3 are swapped. Next 3 are swapped with 5 and the algorithm terminates.

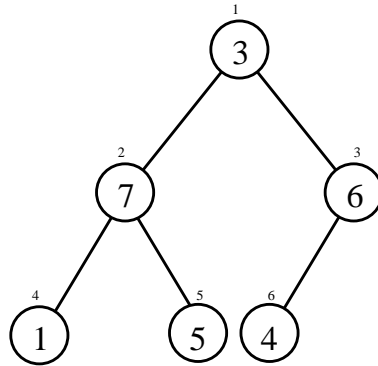


Figure 4: The heap A before the call to the HEAPIFY .

In the little country Algostan it has been decided to build a computer system called **Amandos**. **Amandos** should become a database with information about the population. In the first version of the system the database supports the following three operations:

$\text{INIT}(n)$: Initialise the database, so that it can handle information of a maximum of n persons.

$\text{INSERT}(IDnr, Salary)$: Insert a person with the identification number $IDnr$ in the database. The person should be associated with an information on the annual salary, which is $Salary$. It is assumed that the database does not previous, to the update, contain a person with the identification number $IDnr$.

$\text{ERASELOWESTSALARY}()$: Returns the identification number for a person with the lowest salary, and erases the person from the database.

b) Describe how the first version of **Amandos** can be constructed, so that $\text{INIT}(n)$ takes time $O(n)$, and the two other operations takes time $O(\log n)$, where n is the maximum number of persons to be registered in the database.

Answer b) We can use a red-black tree T ordered by CPR-number. Let s denote the maximal number of elements and r be the number of elements in T .

INIT(n): Set $s \leftarrow n$ and $r \leftarrow 0$

INSERT($CPRnr, Salary$): If $r + 1 > s$ print error. Otherwise insert into T and set $r \leftarrow r + 1$.

ERASELOWESTSALARY(): Find the least element and delete it. Set $r \leftarrow r - 1$.

All operations takes $O(\log n)$ time.

The next version of the database should be extended with the following operations:

SEARCH($IDnr$): Decide wheter or not a person with the identification number $IDnr$ is registered in the database. If there is a person with this identification number, this persons annual salary is returned.

c) Describe how the database can be extended to support the operation SEARCH in time $O(\log n)$, keeping the times from b) for the other operations.

Answer c) SEARCH($CPRnr$): Do a search in T using TREE-SEARCH as defined in CLR.

In the last version of the database it should also include the following operation

SALARYBETWEEN(a, b): Return true, if the database includes a person with an annual salary between kroner a and kroner b . I.e. a salary x , where $a \leq x \leq b$. If not false is returned.

d) Describe how the database can be extended to support the operation SALARYBETWEEN in time $O(\log n)$, keeping the times from c) for the other operations.

Answer d) SALARYBETWEEN(a, b): Calculate the following:

$a_u \leftarrow \text{TREE-SUCCESSOR}(a)$

$b_l \leftarrow \text{TREE-PREDECESSOR}(b)$

Return true if $a \leq a_u \leq b$ or $a \leq b_l \leq b$. Otherwise return FALSE