Department of Informatics

Prof. Dr. Michael Böhlen
Binzmühlestrasse 14
8050 Zurich
Phone: +41 44 635 4333
Email: boehlen@ifi.uzh.ch

University of Zurich[UZH]

Informatics II — Final Exam
Spring 2017 — 01.06.2017

Name: _____ Matriculation number: _____

Advice

If you attend the module *Informatik II*, you have 90 minutes to complete the exam. If you attend the modules, *Informatik IIa* and *IIb*, you have 120 minutes overall to complete both exams.

The following rules apply for the written part of the exam:

- Answer the questions on the exam sheets or the backside. Mark clearly which answer belongs to which question. Additional sheets are provided upon request. If you use additional sheets, put your name and matriculation number on each of them.
- Check the completeness of your exam (18 numbered pages).
- Use a pen in blue or black colour for your solutions. Pencils and pens in other colours are not allowed. Solutions written in pencil will not be corrected.
- Stick to the terminology and notations used in the lectures.
- For the exam Informatik IIb, only the following items are allowed:
  - One A4 sheet (2-sided) with your personal handwritten notes, written by yourself. Sheets that do not conform to this specification will be collected.
  - A foreign language dictionary is allowed. The dictionary will be checked by a supervisor.
  - No additional items are allowed, notably calculators, computers, PDAs, smart-phones, audio-devices or similar devices may not be used. Any cheating attempt will result in a failed test (meaning 0 points).
- Put your student legitimation card ("Legi") on the desk.

Signature:

Correction slot — Please do not fill out the part below

Exercise	1	2	3	4	Total
Points Achieved					
Maximum Points	15	15	15	15	60

Exercise 1 — 15 Points

Chain messages are those that, in the body or subject of the message, ask the recipient to send the message to all his/her direct connections.

Assume a social network whose users can send a message to people they are directly connected with. E.g., in the network illustrated in Figure 1, *Tina* can send a message to *Alice*, *Graham* and *Tom* but she can't send a message to *Eric*.

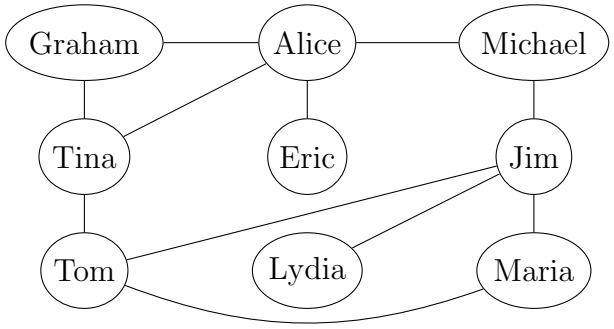


Figure 1: Example Social Network

On date s, user U creates a chain message with ending date e and sends it to his/her direct connections. Nobody is allowed to send the message after the ending date e has passed. A message is received the day it is sent. Each recipient sends the message to all his/her connections the day after the one when s/he receives it. Each recipient sends the message only the first time s/he receives it, otherwise the message is ignored.

Example: A user U creates a chain message with ending date May 31^{st}. He sends it to his/her direct connections on May 11^{th}. His/Her direct connections receive it on May 11^{th} and send it to all their connections on May 12^{th}. The last day when recipients are allowed to send this message is May 31^{st}.

1.1 [3 points] Given the social network illustrated in Figure 1, assume that Graham creates a chain message with ending date June 20^{th} and sends it to his connections on June 18^{th}. Complete the following table with the names of the users that match the cases described.

Cases	**Names**
Users who receive the message for the first time on June 19^{th}	Michael, Tom, Eric
Users who receive the message but are not allowed to send it	Maria, Jim
Users who never receive the message	Lydia

1.2 [3 points] Specify two data structures that can be used to represent the graph of the social network in Figure 1. What are the space requirements of each data structure? Assume a very large network with millions of users where every user has about 100 direct connections. Explain which of the two data structures minimizes the space requirements for such a network.

Two data structures can be used to represent a graph: (i) adjacency lists, with space requirements $\Theta(|V| + |E|)$ and (ii) adjacency matrix, with space requirements $\Theta(|V|^2)$.

The number of edges of the given asymptotically big network can be expressed as a function of number of nodes: $|E| = 100|V|$. So, the space requirement of adjacency lists is $\Theta(|V| + 100|V|) = \Theta(|V|)$ which is symptotically lower than the space requirement of adjacency matrixes $\Theta(|V|^2)$.

1.3 [9 points] Given a network `N`, on date `s`, user `U` creates a chain message with ending date `e`. Use C or pseudocode to describe an algorithm that prints all the users who receive the chain message but are not allowed to send it to their direct connections.

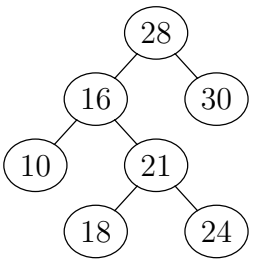
This question can be answered with a modified BFS. The distance `dist` of the starting node is set to $s-1$ and each node is printed if its distance `dist` equal e when it is visited for the first time. If a node is visited with distance equal e then its children are not extended.

```
1  Algorithm: Modified_BFS(N,U, s, e)
2  foreach v ∈ N.V − {U} do
3      v.col = W; v.dist = ∞;
4  U.col = G; U.dist = s-1;
5  InitQueue(Q);
6  Enqueue(Q,U);
7
8  while Q ≠ ∅ do
9      v = Dequeue(Q);
10     if v.dist == e then
11         print v;
12     else
13         foreach u ∈ v.adj do
14             if u.col==W then
15                 u.col = G;
16                 u.dist = v.dist+1;
17                 Enqueue(Q,u);
18     v.col = B;
19 return u.dist;
```

Name: Matriculation number:

Exercise 2 — 15 Points

2.1 [3 points] Let T be a binary search tree without duplicates. The lowest common ancestor of two nodes n_1 and n_2 is the node in T with the largest depth that has both n_1 and n_2 as descendants. Each node is a descendant of itself.



Given the binary search tree illustrated above, determine the lowest common ancestor of the following pairs of nodes:

- 18 and 24: 21
- 24 and 10: 16
- 24 and 16: 16

2.2 [8 points] Assume a binary search tree `T` with root `r` and integers as keys. Each node has pointers to its children but no pointer to its parent. Use C to define the datatype of a node of `T`. Then, use C or pseudocode to describe an algorithm that finds the lowest common ancestor of the nodes `n1` and `n2` of `T`.

Node definition:

```
1 struct node {
2   int val;
3   struct node* left;
4   struct node* right;
5 };
```

../code/bst.c

Recursive solution:

```
1 struct node* lca_rec(struct node* root, struct node* n1, struct node* n2) {
2   if (root==NULL) return NULL;
3   if (root->val > n1->val && root->val > n2->val) {
4     return lca_rec(root->left, n1, n2);
5   }
6   if (root->val < n1->val && root->val < n2->val) {
7     return lca_rec(root->right, n1, n2);
8   }
9   return root;
10 }
```

../code/bst.c

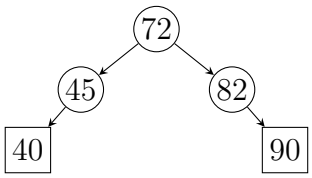
Iterative solution:

```
1 struct node* lca(struct node* root, struct node* n1, struct node* n2) {
2   while(root!=NULL) {
3     if (root->val > n1->val && root->val > n2->val) root=root->left;
4     if (root->val < n1->val && root->val < n2->val) root=root->right;
5   }
6   return root;
7 }
```

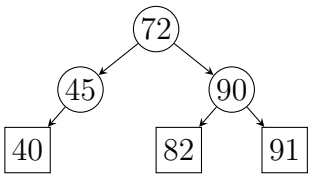
../code/bst.c

2.3 [4 points] When the value **91** is inserted in the red-black tree of Figure 2a, the cases and operations applied are illustrated in Table 1. The result tree is illustrated in Figure 2b. Insert the value `42` in the red-black tree of Figure 2b. Draw the result tree and fill in Table 1 with the cases and operations applied.

Note: Black nodes are denoted with a circle and red nodes are denoted with a square.



(a) Initial red-black tree



(b) Insertion of **91**

Figure 2

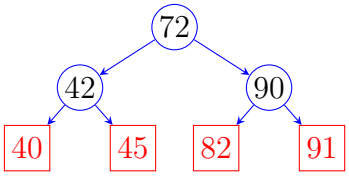


Table 1: Insert cases and operations

Case	Operation	Arguments
Case 3 mirrored	assign color	90, black
	assign color	82, red
	left rotate	82
Case 2	left rotate	40
Case 3	assign color	42, black
	assign color	45, red
	right rotate	45

Name: Matriculation number:

Exercise 3 — 15 Points

3.1 [3 points] Consider the following hash table `HT` with a linear probing scheme of the form $h(k,i) = (h'(k)+i) \bmod 10$, where $h'(k) = k \ mod \ 7$. Assume each slot in `HT` is of type `struct elem {int key; int status}`. Draw the state of the hash table after inserting the values **1, 4, 12, 27, 7, 11, 6** in the given order.

HT:

	key	**status**
0	7	OCC
1	1	OCC
2		EMP
3		EMP
4	4	OCC
5	12	OCC
6	27	OCC
7	11	OCC
8	6	OCC
9		EMP

3.2 [5 points] Use C or pseudocode to describe an algorithm `HTinsert(HT,k)` that inserts key `k` into the hash table `HT` of task 3.1.

```
1  int hprime(int key) {return key % 7;}
2  int h(int key, int i) {return (hprime(key) + i) % 10;}
3
4  int HTinsert(struct elem HT[], int n, int key) {
5    int i, probe;
6
7    i=−1;
8    do {
9      i++;
10     probe = h(key, i);
11   } while (!(i≥n || HT[probe].status!=OCC));
12   if (i≥n) return −1;
13   HT[probe].status = OCC;
14   HT[probe].key = key;
15   return probe;
16 }
```

../code/hash–linear–probe.c

3.3 [4 points] Assume an unsorted array of integers `A[0...n-1]` and a hash table `HT1`. Use C or pseudocode to describe an algorithm that uses `HT1` to print each distinct value of the `A`, i.e., duplicates must be printed only once. For example, if you are given the array,

1	4	7	6	2	3	4	1	1	9

your algorithm should print: **1 4 7 6 2 3 9**

Note: Consider the following two functions as known:

- `HTinsert(HT1, k)`, inserts key `k` into hash table `HT1` and
- `HTsearch(HT1, k)`, searches for `k` in `HT1` and returns the location where `k` is stored or -1 if `k` does not exist in `HT1`.

```
1 Algorithm: printNoDuplicates(A, n)
2 for i = 0 to n − 1 do
3     if HTsearch(HT, k) == -1 then
4         HTinsert(HT, A[i]);
5         print(A[i]);
```

3.4 [3 points] Assuming uniform hashing and load factor $a < 1$, what is the asymptotic complexity of your algorithm in Task 3.3? Justify your answer.

The algorithm scans the elements of the array. If a value is not in the hash table, it is printed and is added to the hash table. Given that checking if a value is in the hash table takes $O(1)$ and given that all elements of the array are accessed once, the complexity of the algorithm is $O(n)$.

Name: Matriculation number:

Exercise 4 — 15 Points

A company manager needs to recruit new people. There are n applicants that are interested in joining this company. Given two arrays $s[n]$ and $v[n]$ and an applicant a, $s[a]$ corresponds to the salary (in thousands of CHF) that an applicant asks for and $v[a]$ corresponds to the estimated value (in thousands of CHF) that the applicant will bring to the company if hired. There is a fixed budget of B thousand CHF that can be spent on salaries for the new employees. The goal of the company is to determine the maximum total value that can be achieved by hiring the appropriate applicants while spending at most B thousand CHF on salaries for the new employees. Note that an applicant either gets the whole salary s/he asks for or is not hired.

Example: Assume that there are 3 applicants and the available budget for salaries of new employees is $B = 8$. The estimated values of the applicants and the salaries they ask for are available in the arrays $v[] = \{10, 7, 4\}$ and $s[] = \{8, 3, 5\}$, respectively. The maximum total value that can be achieved is 11 thousand CHF, by hiring the second and the third applicant, who ask a total salary of $3 + 5 = 8$ thousand CHF.

4.1 [4 points] Assume $P(k, b)$ is the maximum total value achieved if the first k applicants are taken into consideration and b thousand CHF are available as a budget for new salaries. Give a recursive definition of $P(k, b)$.

$$P(k,b) = \begin{cases} 0 & \text{if } k = 0 \text{ or } b = 0 \\ P(k-1,b) & \text{if } s[k] > b \\ max(P(k-1,b), P(k-1,b-s[k]) + v[k] & else \end{cases}$$

4.2 [5 points] Assume that there are 4 applicants and the available budget for new salaries is $B = 8$. The estimated values of the applicants and the salaries they ask for are available in the arrays $v[] = \{6, 5, 3, 7\}$ and $s[] = \{4, 2, 5, 3\}$, respectively. Complete the following table `P` that is used to compute the maximum total value achieved using dynamic programming.

Note: `P[k, b]` is the maximum total value achieved when the first k applicants are taken into consideration and budget b is available for new salaries.

k\b	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	6	6	6	6	6
2	0	0	5	5	6	6	11	11	11
3	0	0	5	5	6	6	11	11	11
4	0	0	5	7	7	12	12	13	13

4.3 [6 points] Given `n` applicants, the arrays `s[n]` and `v[n]` and the available budget `B` for new salaries, describe an algorithm that uses dynamic programming to compute the maximum total value that the company can achieve by hiring the appropriate applicants. You can use either C or pseudocode for your solution.

```
1 void printP(int *P[], int n, int B) {
2   int k, b;
3   printf(" | 0 1 2 3 4 5 6 7 8\n");
4   for(k=0; k≤n; k++) {
5     printf("%d |", k);
6     for (b=0; b≤B; b++) {
7       printf(" %02d", P[k][b]);
8       }
9       printf("\n");
10    }
11 }
12
13 int hire_dp(int s[], int v[], int n, int B) {
14   int k, b;
15   int P[n+1][B+1];
```

../code/ex4.c