FREIE UNIVERSITÄT BOZEN
LIBERA UNIVERSITÀ DI BOLZANO
FREE UNIVERSITY OF BOZEN · BOLZANO

Fakultät für Informatik | Facoltà di Scienze e tecnologie informatiche | Faculty of Computer Science

Advanced Algorithms

Exam Simulation

17th January 2014

FIRST NAME		LAST NAME	
STUDENT NUMBER		SIGNATURE	

Instructions for students

Write First Name, Last Name, Student Number and Signature where indicated. If not, the examination cannot be marked.

Use a pen, not a pencil.

Write neatly and clearly.

Student Code Ethics

Students are expected to maintain the highest standards of academic integrity. Work that is not of the student's own creation will receive no credit. Remember that you cannot give or receiving unauthorized aid on any assignment, quiz, or exam. A student cannot use the ideas of another and declares it as his or her own. Students are required to properly cite the original source of the ideas and information used in his or her work.

Exercise 1 (3 point)
Prove that in any base b≥2, the sum of any three single-digit numbers is at most two digits long.

Solution
A single digit number is at most b−1, therefore the sum of any three such numbers is at most 3(b−1)=3b-3. On the other hand, a two-digit number can be as large as b^2 −1. It is enough to show that b^2−1 ≥ 3b−3. Indeed, b^2 − 1 − 3b + 3 = (b − 1) · (b − 2), which is ≥ 0 for b ≥ 2.

Exercise 2 (4 point)
The RSA cryptosystem is described in the following:

Bob chooses his public and secret keys.

- He starts by picking two large (n-bit) random primes p and q.
- His public key is (N, e) where $N = pq$ and e is a $2n$-bit number relatively prime to $(p-1)(q-1)$.
- His secret key is d, the inverse of e modulo $(p-1)(q-1)$, computed using the extended Euclid algorithm.

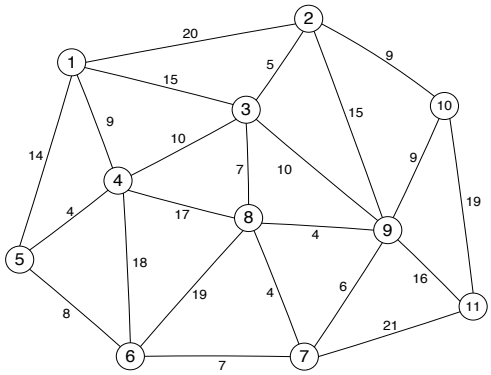
Alice wishes to send message x to Bob.

- She looks up his public key (N, e) and sends him $y = (x^e \bmod N)$, computed using an efficient modular exponentiation algorithm.
- He decodes the message by computing $y^d \bmod N$.

Suppose we have another cryptosystem called WEAK that instead of using a composite *N* = *pq*, simply use a prime *p* so that the public key is *(p,e)* where *e* is an encryption exponent. So, WEAK would encrypt a message *x* into x^e *mod p.*
Prove that WEAK is not secure, by giving an efficient algorithm to decrypt: that is, an algorithm that given *p*, *e*, and x^e *mod p* as input, efficiently computes *x* (the original message).
Justify the correctness and analyze the running time of your decryption algorithm.

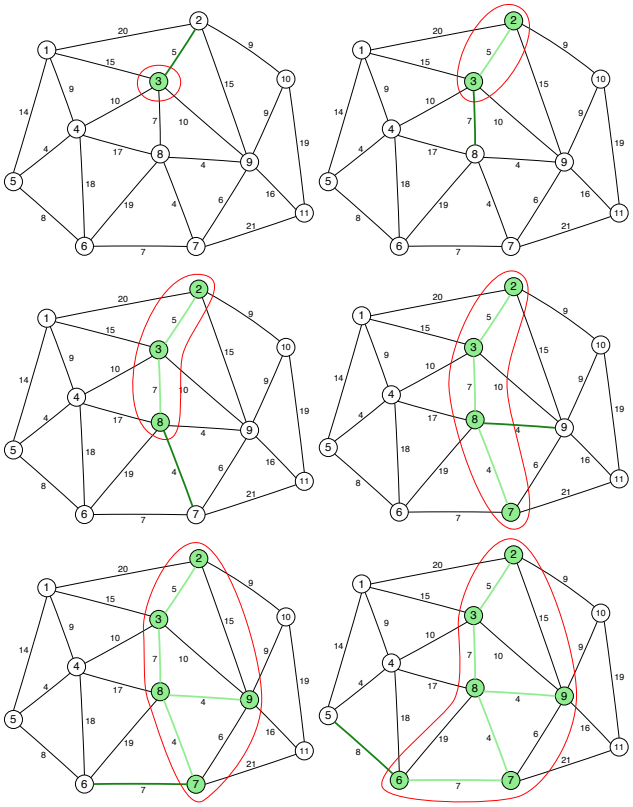
Solution
We just need to calculate the inverse of *e mod (p − 1)*. But this we can do efficiently by using the Extended Euclid's algorithm, whose complexity is $O(n^3)$.

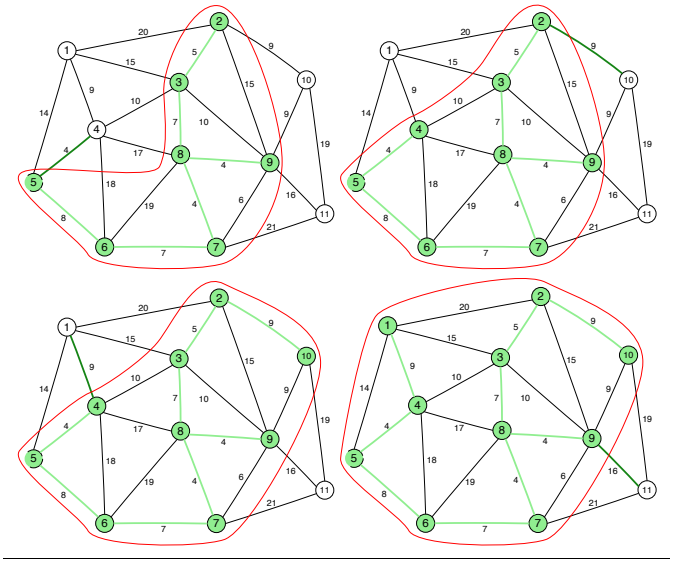
Exercise 3 (3 point)
The Prim's algorithm can find a minimum spanning tree on a graph starting from any node of the graph and repeatedly adding to the partial solution the minimum-weight edge to a node that is not included yet in the partial solution.
Find a minimum spanning tree in the graph G = (V,E) given in the figure, using Prim's algorithm, with 3 as the initial node.



Solution

We apply Prim's algorithm on G, starting at node 3. The set S of the nodes in which the edges selected so far are incident is circled in red. The partial MST spanning tree is highlighted in light green. The edge, among those of minimum cost, that is going to be added to the partial MST is highlighted in dark green.





The edges are added in the following order

reached nodes	edge	cost	iteration
3	(2,3)	5	1
2,3	(3,8)	7	2
2,3,8	(7,8)	4	3
2,3,7,8	(8,9)	4	4
2,3,7,8,9	(6,7)	7	5
2,3,6,7,8,9	(5,6)	8	6
2,3,5,6,7,8,9	(4,5)	4	7
2,3,4,5,6,7,8,9	(2,10)	9	8
2,3,4,5,6,7,8,9,10	(1,4)	9	9
1,2,3,4,5,6,7,8,9,10	(9,11)	16	10=n-1 $\rightarrow$ HALT

The algorithm halts after 10 iteration, when |S| = |V|. The found MST has cost=5+7+4+4+7+8+4+9+9+16=73

Exercise 4 (3 points)

A pig breeder wants to decide what to feed his pigs. He is considering using a combination of pig feeds available from local suppliers. He would like to feed the pigs at minimum cost while also making sure each pig receives an adequate supply of calories and vitamins. The cost, calorie content, and vitamin content of each feed are shown in the table below.

Contents	**Feed Type A**	**Feed Type B**
Calories (per pound)	800	1,000
Vitamins (per pound)	140 units	70 units
Cost (per pound)	$0.40	$0.80

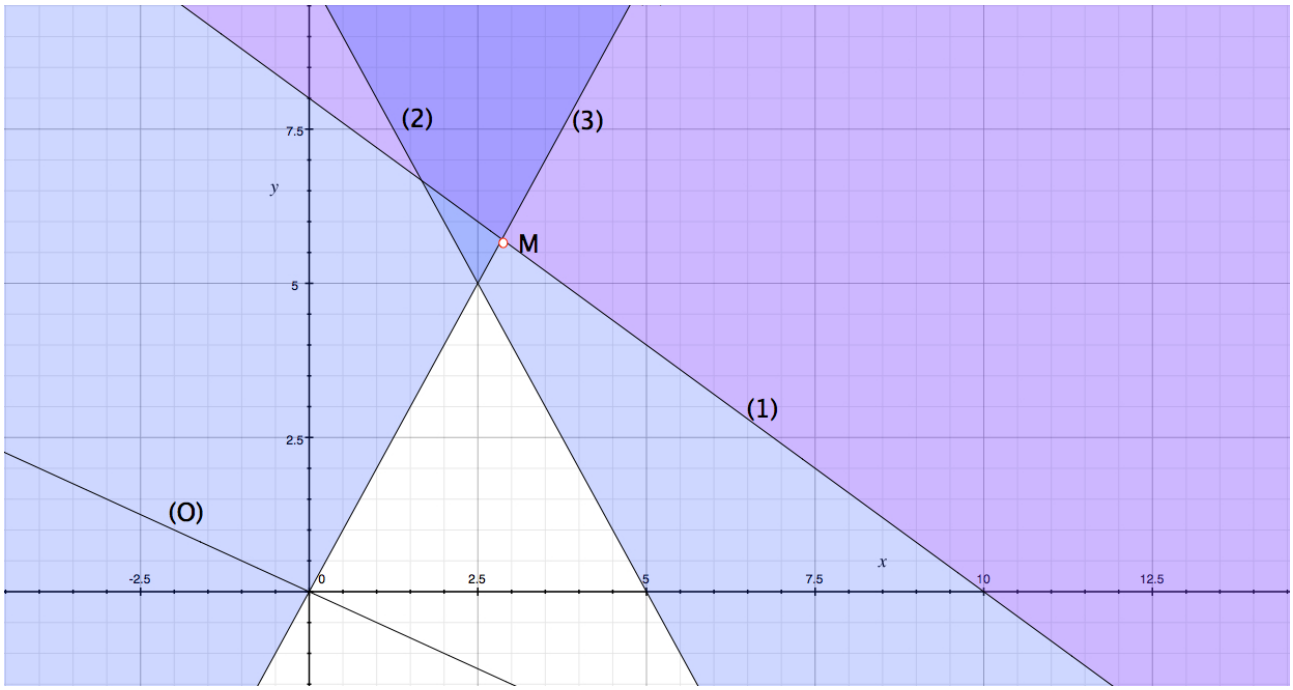
Each pig requires at least 8,000 calories per day and at least 700 units of vitamins per day. In addition, no more than one-third of the diet (by weight) can consist of Feed Type A, since it contains an ingredient, which is toxic if consumed in too large a quantity.

1. Formulate a linear programming model for this problem.
2. Use the graphical method to solve this model. What is the resulting daily cost per pig?

Solution
The LP model for the problem is the following, where A is the amount of Feed Type A and B is the amount of Feed Type B per day.

Minimize 0.4A+0.8B (O)
800A+1000B >=8000 (1)
140A+70B>=700 (2)
A<(A+B)/3 (3)
A>=0
B>=0

The graphical representation of this model is the following. The darkest blue area is the feasible region (trivial constraints are not shown). The line corresponding to all the pairs (A,B) for which the objective is equal to 0 is denoted by O. Given the slope of this line, we can visually determine that the minimum of the objective function is the vertex of the feasible region indicated by M. The coordinates of M can be found calculating the intersection of lines (1) and (3). M=(2.86, 5.71). The value of the objective function in M is about 5.71.



Exercise 5 (3 points)
In a particular network *G=(V, E)*, whose edges have integer capacities c_e, we have already found the maximum flow *f* from node *s* (source) to node *t* (sink). However, we now find out that one of the capacity values we used was wrong: for edge *(u,v)* we used c_{uv} whereas it should have been $c_{uv}-1$. This is unfortunate because the flow *f* uses that particular edge at full capacity: $f_{uv} = c_{uv}$. We could redo the flow computation from scratch, but there's a faster way. Show how a new optimal flow can be computed in $O(|V| + |E|)$ time.

Solution

Algorithm for finding the new optimal flow:

1. Let E' be the edges e ∈ E for which f(e)>0, and let G'= (V,E'). Find in G' a path P_1 from s to u and a path P_2 from v to t.
2. [Special case: If P_1 and P_2 have some edge e in common, then $P_1 \cup \{(u,v)\} \cup P_2$ has a directed cycle containing (u,v). In this case, the flow along this cycle can be reduced by one unit without changing the size of the overall flow. Return the resulting flow.]

3. Reduce flow by one unit along $P_1 \cup \{(u,v)\} \cup P_2$.
4. Run Ford-Fulkerson with this starting flow.

Justification and running time:

Say the original flow has size F. Let's ignore the special case (2). After step (3) of the algorithm, we have a legal flow that satisfies the new capacity constraint and has size F−1. Step (4), Ford-Fulkerson, then gives us the optimal flow under the new capacity constraint. However, we know this flow is at most F, and thus Ford-Fulkerson runs for just one iteration. Since each of the steps is linear, the total running time is linear, that is, O(|V| + |E|).

Exercise 6 (1 point)
Describe the steps and the genetic operators of the Simple Genetic Algorithm.

Solution
The schema of the Simple Genetic Algorithm is described in Lecture 10, page 12. The main genetic operators of the Simple Genetic Algorithm are selection, crossover, and mutation. See Lecture 10 for details on these operators.

Exercise 7 (1 point)
Make an example of a problem to which machine learning can be profitably applied. Informally describe in few lines in English:

- Which is the learning task T;
- Which is the performance measure *P*;
- Which is the training experience *E*.

Solution
See the examples in Lecture 11, pages 4-5.

Exercise 8 (2 points)
Briefly describe the gradient descent algorithm. Is the gradient descent algorithm always assured to find the global minimum of a function? Which is the function that gradient descent optimizes in case of linear regression?

Solution
The gradient descent algorithm is described in Lecture 11, pages 18-20. No, gradient descent is not assured to find the global minimum of a function; it can also converge to a local minimum depending on the point of the function where it starts (see Lecture 11, page 19). In case of linear regression gradient descent is used to optimize the cost function

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$ (see Lecture 11 for details about the cost function)

This cost function is convex and gradient descent is in this case assured to find the global minimum in case of convergence.