TU Wien
Institute of Logic and Computation
Algorithms and Complexity Group

186.814 Algorithmics VU 6.0

2nd Exam (winter term 2021/22)

March 18, 2022

Insert the following information in clearly readable block letters:

First name: | Last name:

Student ID: | Signature:

Please put your student ID on the desk in front of you.

You may write the solutions only to the specification sheets provided by the supervision. You are not allowed to use your own paper. Please use indelible pens (no pencils!).

Question	Points	(max)	Question	Points	(max)
1		(5)	4		(5)
2		(10)	5		(10)
3		(10)	6		(10)

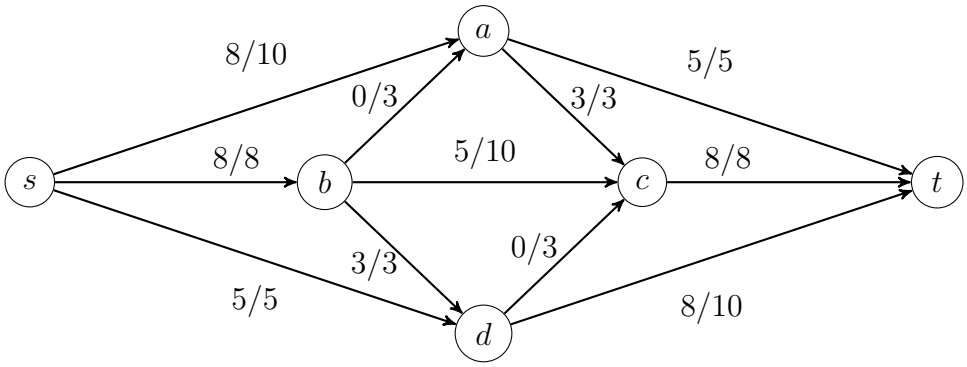
Total: (50)

Good luck!

Question 1 **(5 Points)**

a) (5 Points)

Consider the following flow network $N = (V, E, s, t, c)$ together with a flow f, where each edge e is labeled by $f(e)/c(e)$.



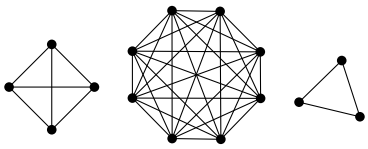
- Draw the residual graph of f. (3 points)

- Is f a maximum flow? Justify your answer. (2 points)

Question 2: Parameterized Algorithms **(10 Points)**

Let P_3 be the path with three vertices. We say a graph G is a *union of disjoint cliques* if the vertex set of G can be partitioned into cliques with no edges between different cliques.

The graph P_3



A disjoint union of cliques

By solving the following two subtasks, we will show that it is fixed parameter tractable to decide whether one can remove k vertices from a graph such that the resulting graph is a union of disjoint cliques.

a) (3 Points) We say a graph G contains P_3 *as induced subgraph* if there are distinct vertices $a, b, c \in V(G)$ such that $ab, bc \in E(G)$ and $ac \notin E(G)$. Show that a graph G is a union of disjoint cliques if and only if G does not contain P_3 as induced subgraph.

union of disjoint cliques $\Rightarrow$ no induced P_3

no induced P_3 $\Rightarrow$ union of disjoint cliques

b) (7 Points)

P_3-Removal
Instance: A graph G and an integer k. *Parameter*: k. *Question*: Does there exist a set X of at most k vertices in G such that deleting X from G results in a graph with no P_3 as induced subgraph?

Design and describe a fixed-parameter algorithm for P_3-Removal. You are allowed to either give a sufficiently detailed description of the algorithm or provide a high-level pseudocode. You may use a subroutine `hasP3(`H`)`, running in time $O(n^3)$, that either yields three vertices inducing a P_3 in H or states that P_3 does not occur as induced subgraph in H. Additionally, give an explicit upper bound on the runtime of your algorithm.

Hint: Try using the bounded search tree technique.

Question 3: Randomized Algorithms **(10 Points)**

Randomized counting. You are given a bag with an unknown number n of balls in it and want to estimate this number. You are only allowed to iteratively select a ball uniformly at random and return it. In an initial step, you select a ball and mark it in some way before returning it.

a) (2 points)

Let X be the random variable corresponding to the number of iterations until the marked ball is re-selected the first time. What is the expected value of this number of iterations in dependence of n?

b) (2 point)

In reverse, when you re-select the marked ball the first time in iteration k, what is your estimate for the total number of balls n?

c) (3 points)

More specifically, what is the probability to re-select the marked ball the first time in the k-th iteration in dependence of n?

d) (3 points)

To speed up the process, you decide to mark *every* ball you select in each iteration. What is now the probability to re-select any formerly marked ball in the k-th selection step the first time in dependence of n? You do not need to come up with a simplified formula, instead briefly argue your answer.

Just for your information if you are curious: Further calculation and simplification, which we omit here, yields an estimate of $\tilde{n} = 2k^2/\pi$ balls for an observed first re-selection of a marked ball in iteration k.

Question 4: Structural Decompositions and Algorithms (5 Points)

The *closed neighborhood* $N[v]$ of a vertex v in a graph $G = (V, E)$ consists of v and its neighbors, formally $N[v] = \{w \in V : w = v \text{ or } vw \in E\}$. A *2-dominating set* of a graph is a subset $S \subseteq V$ of vertices such that, for each vertex $v \in V$, $|S \cap N[v]| \geq 2$. The Minimum 2-Dominating Set problem asks for the cardinality of a minimum-size 2-dominating set in a given graph G.

Prove that Minimum 2-Dominating Set is fixed-parameter tractable when parameterized by the treewidth of the input graph. To do this, write down an MSO formula $\phi(X)$ expressing that X is a 2-dominating set.

Question 5: Linear Programming **(10 Points)**

Consider a constrained version of the *shortest path problem*. Given is a weighted directed simple graph $G = (V, A)$ with nodes $V = \{1, \ldots, n\}$, arcs $A \subseteq V \times V$ and costs $c_{ij} > 0$ associated with each arc $(i, j) \in A$. Furthermore, a *source node* $s \in V$ and a *target node* $t \in V$ $(s \neq t)$ are given. The goal is to find a shortest path from s to t such that several constraints are fulfilled.

The following mixed integer linear program (MILP) is based on the sequential formulation and models the *unconstrained shortest path problem*. Note that this is not an efficient approach for the *unconstrained shortest path problem*, but the model provides a helpful basis for the subsequent tasks of considering further constraints.

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

$$y_s = 1 \tag{2}$$

$$y_t = 1 \tag{3}$$

$$m = \sum_{(i,j)\in A} x_{ij} \tag{4}$$

$$\sum_{i\in V} y_i = m + 1 \tag{5}$$

$$u_s = 0 \tag{6}$$

$$u_t = m \tag{7}$$

$$u_i \leq (n-1) \cdot y_i \qquad \forall i \in V \tag{8}$$

$$u_i + x_{ij} \leq u_j + (n-1) \cdot (1 - x_{ij}) \qquad \forall i, j \in V : (i, j) \in A \tag{9}$$

$$u_i \leq m \qquad \forall i \in V \tag{10}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{11}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{12}$$

$$u_i \geq 0 \qquad \forall i \in V \tag{13}$$

$$m \geq 0 \tag{14}$$

The following decision variables are used:

- $y_i \in \{0, 1\}$, $i \in V$: Indicates if node i is visited $(y_i = 1)$ or not $(y_i = 0)$.
- $x_{ij} \in \{0, 1\}$, $(i, j) \in A$: Indicates if arc (i, j) is used $(x_{ij} = 1)$ or not $(x_{ij} = 0)$.
- $u_i \geq 0$, $i \in V$: Indicates the order in which node i is visited. If $i = s$ or if node i is not visited, then $u_i = 0$ as guaranteed by the constraints (6) and (8).
- $m \geq 0$: Number of arcs in the resulting path.

The model decides what nodes and arcs are used in the path, and the nodes, arcs and order variables u_i are appropriately linked. Note that m may **not** be used as (tight) upper bound in constraints (8) and (9) because $m \cdot y_i$ and $m \cdot (1 - x_{ij})$ would multiply decision variables and, thus, these inequalities would not be linear anymore.

In the following tasks, extend the above MILP by means of formulating additional constraints such that the corresponding requirements are satisfied. If you also introduce new variables, clearly explain their meaning and define their domains.

a) (2 Points) Given a subset of arcs $S_A \subseteq A$, at most three of them may be used.

b) (2 Points) Each node $i \in V$ has a certain amount $k_i \geq 0$ of a given resource. When a node is visited on the resulting path all the available resources of that node are collected. While following the path from s to t at least K units of the resource have to be collected.

c) (3 Points) Given a subset of nodes $S_V \subseteq V$, either none of them may be visited or at least five of them must be visited.

d) (3 Points) If node $k \in V$ is visited, then node $l \in V$ may not be visited anytime before k. Note that l may be visited anytime after k, or may be visited anytime if k is not visited.

Question 6: Geometric Algorithms **(10 Points)**

a) (6 Points)

Let P be a set of n points in the plane $\mathbb{R}^2$ and let T be a Range Tree for P. Using T, design an algorithm for the following type of query, explain why it is correct, and provide an analysis of its asymptotic running time.

INPUT: point set P, axis-aligned rectangle $R = [x_l, x_r] \times [y_b, y_t]$ ($x_l < x_r$, $y_b < y_t$)

OUTPUT: list S of all points in P that are at L_1 distance[1] at most 1 from the boundary of R

Describe your algorithm in high-level pseudocode or as sufficiently detailed plain text. Your algorithm should have an output sensitive running time $O(k + \log^2 n)$, where k is the number of output points.

[1]The L_1 (or Manhattan) distance $d(p, q)$ of two points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ is defined as $d(p, q) = |p_x - q_x| + |p_y - q_y|$.

b) (4 Points)

Consider the randomized algorithm for computing the smallest enclosing disk (SED) D of a set P of $n > 4$ points in the plane. Assume you need to maintain the SED for a dynamic scenario, where new points can be added to P or points from P can be deleted. Here we are interested in the deletions.

- What is the tight upper bound on the probability that the smallest enclosing disk needs to be updated when removing a point $q \in P$, i.e., the probability of $SED(P) \neq SED(P \setminus \{q\})$? Explain your answer.

- Re-using the algorithm from the lecture, compute the expected asymptotic running time for the update procedure when deleting one point from P.

- Draw an example of at least 6 points and their SED D, in which any single point can safely be removed without the need to update D.