Final Exam, Algorithms 2015-2016

- You are only allowed to have a handwritten A4 page written on both sides.
- Communication, calculators, cell phones, computers, etc... are not allowed.
- Your explanations should be clear enough and in sufficient detail that a fellow student can understand them. In particular, do not only give pseudocode without explanations. A good guideline is that a description of an algorithm should be such that a fellow student can easily implement the algorithm following the description.
- **Do not touch until the start of the exam.**

Good luck!

Name: ______________________ **N° Sciper:** ______________

Problem 1 / 23 points	Problem 2 / 10 points	Problem 3 / 15 points	Problem 4 / 16 points	Problem 5 / 17 points	Problem 6 / 19 points

Total / **100**

1 *(23 pts)* **Basic questions.** This problem consists of four subproblems (1a-1d) for which you do *not* need to motivate your answers.

1a *(6 pts)* Let $A[1 \ldots 9] =$

3	9	5	4	8	6	11	7	2

be an array consisting of 9 numbers. Illustrate how A looks like after executing the code

$\textsc{Max-Heapify}(A, 4, 9)$, $\textsc{Max-Heapify}(A, 3, 9)$, $\textsc{Max-Heapify}(A, 2, 9)$, $\textsc{Max-Heapify}(A, 1, 9)$.

Solution:

Resulting $A =$

1b (6 pts) A brilliant and highly motivated student at EPFL has decided to improve upon Strassen's algorithm for matrix multiplication. He strongly believes that after partitioning the matrices into 16 submatrices each of dimension $n/4 \times n/4$ — similarly to Strassen's algorithm (recall that Strassen's algorithm partitions the matrices into 4 submatrices each of dimension $n/2 \times n/2$) — one can obtain the final answer by using 32 matrix multiplications (each multiplying two matrices of dimension $n/4 \times n/4$). If the student modifies Strassen's algorithm using this idea, what is the running time obtained for multiplying two matrices of dimension $n \times n$?

Solution:

The running time is __________

1c *(5 pts)* Suppose you have n distinct keys that you wish to store in a hash table. We have six different alternatives for the size of the hash table:

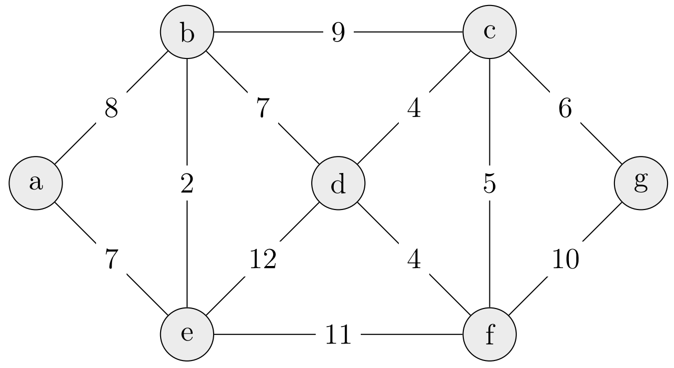
A: n **B:** $n \log n$ **C:** n^2 **D:** $n^2 \log n$ **E:** n^3 **F:** $n^3 \log n$

Assuming simple uniform hashing, what is the smallest size of the hash table (among the above alternatives) such that the expected number of collisions is less than 1?

Solution:

The choice is __________

1d *(6 pts)* Consider the following undirected graph with edge weights:



Write the weights of the edges (in the correct order) added to the shortest path tree by Dijkstra's algorithm starting from a, i.e., with source a.

Solution:

---

2 *(10 pts)* **Probabilistic analysis.** Consider the following algorithm Random-Cut that takes as input an undirected unweighted graph $G = (V, E)$.

Random-Cut(G)

1. Let s and t be any two vertices in G.
2. Let $S = \{s\}$ and $T = \{t\}$.
3. **for each** vertex v in G different from s and t
4. Add v to S or to T with equal probability $1/2$.
5. **return** the cut (S, T)

2a *(6 pts)* Recall that an edge crosses a cut (S, T) if one of its endpoints is in S and the other one is in T. **Prove** that in expectation the number of edges crossing the cut returned by Random-Cut is at least $|E|/2$, i.e., at least half of the edges cross the cut in expectation.

Solution:

Continuation of the solution to 2a:

2b *(4 pts)* **Make a small modification** to the algorithm RANDOM-CUT so that *strictly more* than half of the edges cross the returned cut in expectation. Here, we assume that the input graph contains at least one edge.

Solution:

Continuation of the solution to 2b:

3 *(15 pts)* **Job assignments.** Consider the following job assignment problem:

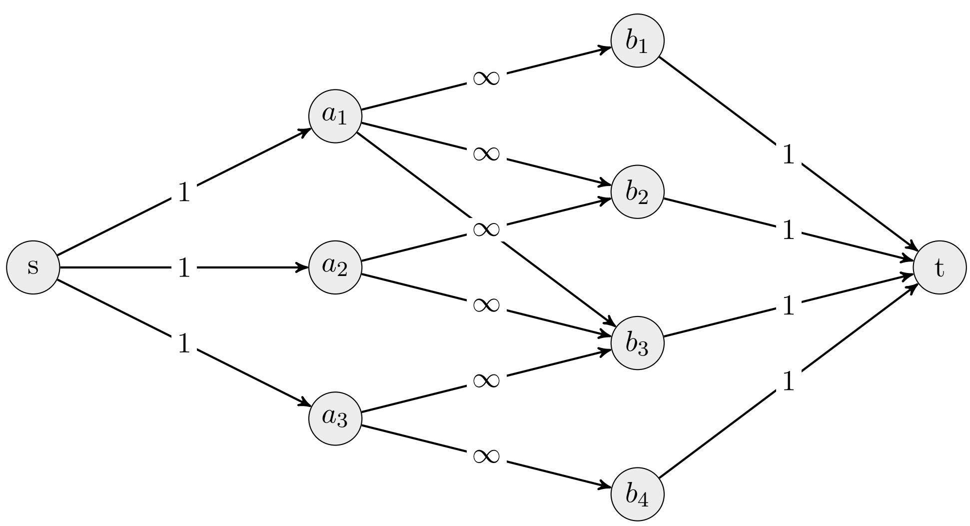
> There is a set $A = \{a_1, a_2, \ldots, a_n\}$ of n students and a set $B = \{b_1, b_2, \ldots, b_m\}$ of m jobs. Each student $a_i \in A$ is interested in a subset $N(a_i) \subseteq B$ of the jobs. The goal is to find (if possible) an assignment of jobs to students so that each student $a_i \in A$ gets exactly one job $b_j \in N(a_i)$ and each job is assigned to at most one student.

In class, we saw how to formulate this problem as a max flow problem in a flow network G so that the maximum flow value is equal to n if and only if there exists an assignment of jobs to the students. The flow network G is defined as follows (see also example below):

- The vertices consist of a source s, a sink t, a vertex $a_i \in A$ for each student, and a vertex $b_j \in B$ for each job.
- There is an arc of capacity 1 from the source s to a_i for each student $a_i \in A$. Similarly, there is an arc of capacity 1 from b_j to the sink t for each job $b_j \in B$.
- Finally, there is an arc of capacity ∞ from a_i to b_j if and only if $b_j \in N(a_i)$, i.e., if student a_i is interested in the job b_j.

The task, in the following two subproblems, is to analyze for which instances there is a job assignment (or equivalently, for which instances the flow network has a max flow value of n).

Example of flow network: Consider the instance when $S = \{a_1, a_2, a_3\}$, $J = \{b_1, b_2, b_3, b_4\}$, and $N(a_1) = \{b_1, b_2, b_3\}$, $N(a_2) = \{b_2, b_3\}$ and $N(a_3) = \{b_3, b_4\}$. The corresponding flow network can then be depicted as follows:



3a *(5 pts)* Suppose there is a subset $A' \subseteq A$ of students so that $|A'| > |N(A')|$, where $N(A') = \bigcup_{a_i \in A'} N(a_i)$. Then there cannot be an assignment of jobs to the students because the total number of jobs that interest the students in A' is less than the number of students in A'.

In this case, which of the following cuts is guaranteed to have capacity less than n?

A: The cut defined by $S = \{s\}$ and $T = \{t\} \cup A \cup B$.

B: The cut defined by $S = \{s\} \cup A \cup B$ and $T = \{t\}$.

C: The cut defined by $S = \{s\} \cup A'$ and $T = \{t\} \cup (A \setminus A') \cup B$.

D: The cut defined by $S = \{s\} \cup N(A')$ and $T = \{t\} \cup A \cup (B \setminus N(A'))$.

E: The cut defined by $S = \{s\} \cup A' \cup N(A')$ and $T = \{t\} \cup (A \setminus A') \cup (B \setminus N(A'))$.

F: The cut defined by $S = \{s\} \cup A' \cup (B \setminus N(A'))$ and $T = \{t\} \cup (A \setminus A') \cup N(A')$.

Solution:

The cut which is guaranteed to have capacity less than n is __________

(You are *not* required to motivate your answer in this subproblem.)

3b *(10 pts)* **Prove** that if every $A' \subseteq A$ satisfies

$$|A'| \leq |N(A')| \qquad \text{(where again } N(A') = \bigcup_{a_i \in A'} N(a_i)\text{)}$$

then there is a job assignment, i.e., the max flow has value n.

Solution:

Continuation of the solution to 3b:

4 *(16 pts)* **Shortest paths.**

4a *(3 pts)* Consider a directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$ and a vertex/source $s \in V$. We assume that G does not contain any negative cycles. In addition, we have the following facts:

- The edges $(u_1, v), (u_2, v) \in E$ are both part of shortest paths from s to v and $w(u_1, v) = 10, w(u_2, v) = 20$.
- The length of a shortest path from s to v is 1981.

What are the lengths of shortest paths from s to u_1 and from s to u_2?

Solution:

The length of a shortest path from s to u_1 is __________

The length of a shortest path from s to u_2 is __________

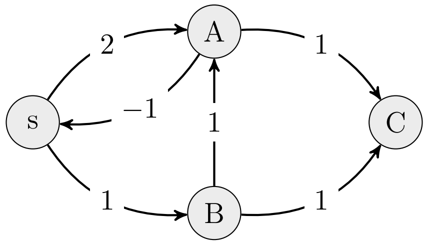
(You are *not* required to motivate your answers in this subproblem.)

4b *(13 pts)* **Design and analyze** an efficient algorithm for the following problem:

> Given a directed graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$ and a vertex/source $s \in V$, return all edges that are part of a shortest path starting at s.

You may assume that the graph only contains non-negative cycles but it may contain negative edge weights.

Example: Consider the following input:



Then the shortest paths from s are $s \to A$, $s \to B$, $s \to B \to A$, $s \to B \to C$. Therefore the correct output of the algorithm would be the edges $(s, A), (s, B), (B, A), (B, C)$.

Solution:

Continuation of the solution to 4b:

5 *(17 pts)* **The funny array game. Design and analyze** an efficient algorithm for the following problem of optimally playing a game on an array:

> You are given a sequence of n positive numbers $a_1, a_2, ..., a_n$. Initially, they are all colored black. In one move, you can choose a black number a_k and color it and its immediate neighbors red (the immediate neighbors are the elements a_{k-1}, a_{k+1}, if they exist). You get a_k points for this move. What is the maximum total number of points that you can get during the game?

For full credit, your algorithm should run in time $O(n)$.

Example: Given a sequence `1 2 6 4 7`, you can start by choosing `6`; when you do, you get 6 points and have to color `2 6 4` red. Next, you can choose `7`, which gives you `7` points; you color `7` red (its neighbor `4` was already red). Finally, you choose `1` (the only remaining black number), get 1 point and color it red. Now all numbers are red and you cannot make any more moves. You obtain $6 + 7 + 1 = 14$ points, which is the maximum possible score.

Solution:

Continuation of the solution to 5:

6 *(19 pts)* **Median of two sorted arrays.**

Given two *sorted* arrays A and B that both contain n integers, **design and analyze** an efficient algorithm that returns the median of the array resulting from merging A and B. For full credit, your algorithm should run in time $O(\log n)$.

(Recall that the median of k numbers $a_1 \leq a_2 \leq \cdots \leq a_k$ is $a_{\lceil k/2 \rceil}$ if k is odd and $\frac{a_{(k/2)}+a_{(k/2+1)}}{2}$ if k is even.)

Solution:

Continuation of the solution to 6: