Institute for Software Technology
Yannic Maus, Birgit Vogtenhuber, Manuel Jakob

January 15, 2024

Design and Analysis of Algorithms WS 2023/24 Additional Questions

Self Test

The following list of questions/statements should be used for personal reflection. The list does not attempt to cover all material relevant for the partial exam, but it covers many important aspects that will likely be relevant for the partial exam. Feel free to ask any questions on these questions/statements/exercises on discord.

Please report typos inaccuracies on discord.

1. I understand the definition of NP (decision problems!) and I know how to show that a decision problem is in NP.
2. I understand in principle how to prove that a problem is NP-hard (not the core content of the exam).
3. I understand the difference between the Ford-Fulkerson method and the Edmonds Karp Algorithm.
4. I can execute the Ford-Fulkerson method on small examples, including being able to draw the residual network, backwards edges, etc.
5. I understand the *Min-st-cut Max Flow Theorem* and can use the lecture material to determine a Min-st-cut in a directed network. I also understand how to use the theorem to show that a given flow is a maximum flow.
6. I understand the difference between an Min-st-cut and a Min-cut.
7. I understand the definition of an approximation factor/ratios and why sometimes approximation ratios are larger than 1 and sometimes they are smaller than one.
8. While this depends a lot on the problem/algorithm at hand, I have some ideas what I need to show that a certain algorithm has a certain approximation guarantee.
9. I understand what a randomized algorithm is and understand the difference between a Las Vegas and a Monte Carlo algorithm.
10. I understand how to amplify the success probability of a Monte Carlo algorithm.
11. I understand why the (small) success probability of $1/(n(n-1))$ for one iteration of Karger's algorithm is helpful.
12. I can execute Karger's algorithm on small examples.
13. I understand the proof of Karger's algorithm to the extent that I understand the conditional probabilities appearing in the proof.
14. I can apply Markov's inequality and understand how it can be useful when analyzing a randomized algorithm.

15. I understand how to use *linearity of expectation*.

16. I can compute simple probabilities.

17. I understand what it means that an algorithm is correct *with high probability*.

18. I understand the LOCAL model and can design simple LOCAL model algorithms.

19. I understand how to compute an MIS in the distributed setting, and I can also compute an MIS when given an input coloring.

20. I understand why iteratively coloring uncolored nodes that are have a smaller ID than all their uncolored neighbors is a slow distributed algorithm.

21. I understand the SimpleColorReduction algorithm and why it works.

22. I understand how/when one can improve the SimpleColorReduction algorithm to reduce the number of colors in a coloring with $C \gg \Delta + 1$ colors much faster. (this was an exercise)

23. I understand how to compute a maximal matching in a bipartite graph with a distributed proposal algorithm (this was a bonus question in the exercises).

24. I understand the Max-Cut algorithm from the lecture and what I need to change to compute a Max-Cut (multi-cut) into more than 2 parts (e.g., into 4 parts as in the exercise session) with the objective of maximizing the number of bichromatic edges (without guaranteeing an optimal solution).

25. We have seen many different types of 'graph colorings'. I understand why some of them are proper colorings (adjacent vertices cannot receive the same color) while some may not (e.g., the ones in the max cut problem).

26. I understand the definition of all problems mentioned in the lectures (e.g., set cover, vertex cover, MIS, C-coloring, $\Delta + 1$-coloring, Max Flow, Min-st-cut, Min Cut, Max Cut, ...)

27. I understand the solutions to all homework assignments and can solve them myself.

1 Las Vegas - Expected runtime is not enough

Assume a randomized Las Vegas algorithm $\mathcal{A}$ for an optimization problem Π with expected runtime upper bounded by $5n$ on instances of size n. Show that with high probability its runtime is $O(n^2)$.

2 It's a Match(ing)

Consider the general LOCAL model where each vertex is equipped with a unique ID from an ID space of size N.

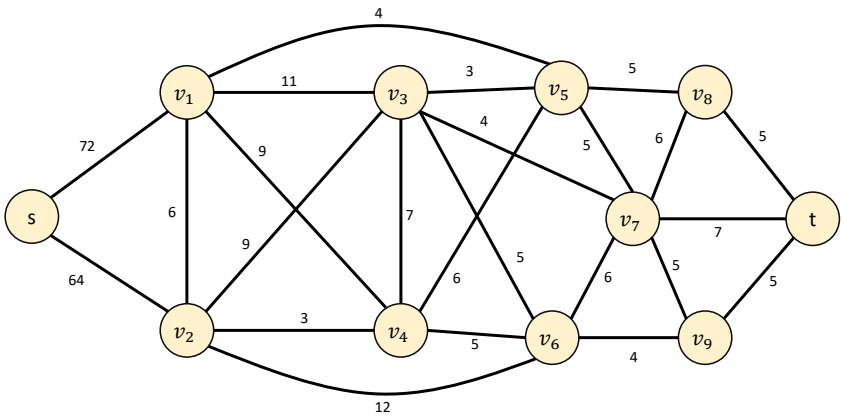
1. (3%) Describe an efficient distributed algorithm that computes a maximal matching for *path graphs, that is, your algorithm only needs to work correctly when the input graph is a path and all nodes know that the input graph is a path. Argue about the correctness and round complexity of your algorithm. Hint: You may use the ideas from the last exercise sheet. It is known that there is a lower bound of* $\Omega(\log^* N)$ *for this problem (you do not need to prove this lower bound).*

2. *(2%) Prove that the computed maximal matching in task 1 is a constant factor approximation for the maximum matching problem on path graphs, that is, it is a* ρ*-approximation for some constant* $0 < \rho < 1$.

3. *Design a centralized algorithm for the problem. What's your algorithm's asymptotic runtime? What is its approximation guarantee?*

3 The 'Min-Cut = Max-Flow Theorem'

Consider the following undirected weighted graph.



The *directed version* of the graph is obtained by replacing each undirected edge $\{u, v\}$ with weight w with two directed edges (u, v) and (v, u) that both have weight w.
Find a minimum weighted s-t-cut in the directed version of the graph and use the **'min-cut = max-flow theorem'** to prove that your cut is optimal. For that you will have to compute a maximum flow. While you do not need to provide any intermediate steps on how you obtained your flow, you need to reason why your flow

is a feasible flow. You can omit all (directed) edges on which your flow has value zero. What is the value of your cut?

4 The LOCAL model

Design a distributed algorithm that solves the minimum vertex cover problem in $O(\mathsf{diameter})$ rounds despite the problem being NP-complete.

5 Max Flow and Backward Edges

Design a Max-flow network, and a submaximal flow for the network, such that the Ford-Fulkerson method needs to use backwards edges to increase the flow.
Remark: This question has not been covered in the discussion session but the lecture contains an example that solves the exercise. We recommend to construct your own example as it will provide you with a good understanding on backward edges.

6 Minimum Dominating Set

A *dominating set* of a graph $G = (V, E)$ is a subset $S \subseteq V$ of the nodes such that every node $v \in V$ is contained in S or has a neighbor in S. A *minimum dominating set* is a dominating set with minimum cardinality among all dominating sets. It is well known that the MDS problem is NP-complete.

- Design a deterministic approximation algorithm for the MDS problem with approximation factor $O(\log n)$ where n is the number of nodes of the graph.

Hint: This was discussed in the discussion session. The solution was based on a reduction to the set-cover problem.

7 Karger's algorithm

Consider the following graph.

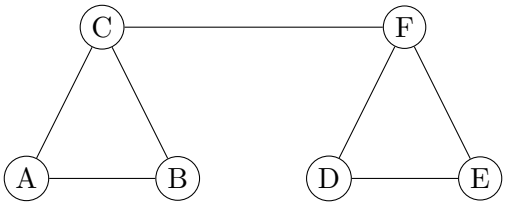


***Fig. 1:* Graph for Karger's algorithm.**

1. What is the global min-cut of the graph? Is it unique?
2. What is the probability that Karger's algorithm picks an edge of the min-cut in the first iteration?
3. Execute one iteration of Karger's algorithm and assume that the first contracted edge is the edge $\{A, C\}$.

8 Rounding Fractional Set Cover

Definition 1 *In probability theory for two events A and B the event $Pr(B \cup B)$ states that either event A* ***or*** *event B occurs. The* ***union bound*** *provides a general upper bound that at least one of the events happens. Formally, for a set of events $A_1, A_2, ..., A_n$ we have*

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} \Pr\left[A_i\right].$$

The set cover problem from the lecture can be extended to a fractional version. In the fractional set cover, it is also allowed to select a fraction $f_i \in [0, 1]$ of a set S_i. Finally, for each element x_j of the universe X the sum of fractions of sets that contain x_j is at least 1. For example, consider the universe $X = \{1, 2, 3\}$ and the collection of sets $S = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$. The fractional set cover size is 1.5, in which a 0.5 fraction of each set is taken.

- Assume we are given a fractional solution $(f_i)_{i\in\{1,...,m\}}$ to a set cover instance $(X, \mathcal{S} = \{S_1, \ldots, S_m\})$ of size $\alpha = \sum_{i\in\{1,...,m\}} f_i$. Design a randomized algorithm that solves the integral version of the set cover and outputs a solution that is of expected size of at most $\alpha \cdot 2 \log |X|$ and covers each element of X with high probability (in the size of the universe $|X|$).

  Hint: Randomized algorithms are often simple. What's the simplest rounding procedure that you can come up with? What's the probability that an element remains uncovered? Adjust your rounding procedure if necessary.

9 Optimizing a production line.

Remark: This question was a homework assignment in a previous year and has not been discussed this year.

Assume that you are the production engineer in some company. As the factory that you are responsible for performs much better than other factories of the company, the CEO asks for your help. The CEO has a mathematics and business administration background, but has not undergone a computer science degree. You are approached with the following situation:
One of our production lines performs poorly. It consists of 23 *steps that need to be performed sequentially. In the i-th step, we have s_i machines, and the j-th machine of step i can process p_{ij} items per day. We have connected the machines of two consecutive steps with a (huge) network of conveyor belts. As you can see we have belts with different capacities and many of our machines are connected with several belts to the machines of preceding/proceeding steps.*
Please help us with the following questions.
A: How can we optimize our productivity, that is, the number of goods that we output after the last step? This year, we do not want to buy new machines and also we do not want to re-connect the conveyor belts. So, what can we do and what's an optimal setup?
B: Next year, we want to invest in this factory in order to increase our production capacity. Can you help us to find the bottleneck of our production line so we can optimally use the available funds?

1. (1%) Explain which choices the production engineer has to increase the productivity in the setting of question A.

2. (5%) Use the lecture material to design an algorithm to help your company to solve part A. While you can use every result and algorithm presented in the lecture in a black-box manner, do not forget to reason why your algorithm solves A and how the runtime of your algorithm depends on the number of machines in the factory.

3. (2%) Modify your algorithm from the previous part (or design a new algorithm) to help the CEO with question B. Again, do not forget to reason why your algorithm is helpful for solving question B.