Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Mohsen Ghaffari, Angelika Steger, David Steurer, Emo Welzl, Peter Widmayer

Algorithms, Probability, and Computing — Exercises KW50 — HS17

The following exercises will be discussed in the exercise class on December 13, 2017. Please hand in your solutions not later than December 12.

Exercise 1: Cover free families

Here, we use the concept of cover free families, as defined in Definition 8.17, to obtain an encoding that allows us to recover information after superimposition. That is, we will be able to decode even if k of the codewords are *superimposed* and we only have the resulting bit-wise OR.

More concretely, we want a function $Enc : \{0,1\}^{\log n} \to \{0,1\}^m$ — that encodes n possibilities using m-bit strings for $m \geq \log_2 n$ — such that the following property is satisfied: $\forall S, S' \subseteq \{1, ..., n\}, S \neq S'$ such that $|S| \leq k$ and $|S'| \leq k$, we have that $\vee_{i \in S} Enc(i) \neq \vee_{i \in S'} Enc(i)$. Here $\vee$ denotes the bit-wise OR operation.

Present such an encoding function, with a small m, that depends on n and k. What is the best m that you can achieve?

Exercise 2: More Algorithms for Color Reduction

(a) Design a single-round algorithm that transforms any given k-coloring of a graph with maximum degree Δ into a k'-coloring for $k' = k - \lfloor \frac{k}{\Delta+2} \rfloor$, assuming $k \geq \Delta + 2$.

(b) Use repetitions of this single-round algorithm, in combination with the $O(\log^* n)$-round $O(\Delta^2 \log \Delta)$-coloring of Theorem 8.12, to obtain an $O(\Delta \log \Delta + \log^* n)$-round $(\Delta + 1)$-coloring algorithm.

Exercise 3: Deterministic coloring

Here, we see yet another deterministic method for computing a $(\Delta + 1)$-coloring in $O(\Delta \log \Delta + \log^* n)$ rounds. First, using Theorem 8.16, we compute an $O(\Delta^2 \log \Delta)$-coloring ϕ_{old} in $O(\log^* n)$ rounds. What remains is to transform this into a $(\Delta + 1)$-coloring, in $O(\Delta \log \Delta)$ additional rounds.

The current $O(\Delta^2 \log \Delta)$-coloring ϕ_{old} can be written using $C \log \Delta$ bits, assuming a sufficiently large constant C. This bit complexity will be the parameter of our recursion. Partition G into two vertex-disjoint subgraphs G_0 and G_1, based on the most significant bit in the coloring ϕ_{old}. Notice that each of G_0 and G_1 inherits a coloring with $C \log \Delta - 1$

bits. Solve the $\Delta + 1$ coloring problem in each of these independently and recursively. Then, we need to merge these colors, into a $\Delta + 1$ coloring for the whole graph.

(A) Explain an $O(\Delta)$-round algorithm, as well its correctness proof, that once the independent $(\Delta + 1)$-colorings of G_0 and G_1 are finished, updates only the colors of G_1 vertices to ensure that the overall coloring is a proper $(\Delta + 1)$-coloring of $G = G_0 \cup G_1$.

(B) Provide a recursive time-complexity analysis that proves that overall, the recursive method takes $O(\Delta \log \Delta)$ rounds.