Antonios Antoniadis and Marvin Künnemann **Winter 2018/19**

Exercises for Randomized and Approximation Algorithms

www.mpi-inf.mpg.de/departments/algorithms-complexity/teaching/winter18/rand-apx-algo/

Exercise Sheet 2: Basics of Randomized Algorithms

To be handed in by **October 30rd, 2018, 2pm**, via e-mail to André Nusser
(CC to Antonios Antoniadis and Marvin Künnemann)

Exercise 1 *(5 Points)* Consider the (unweighted) Max-k-SAT problem: Given a Boolean formula ϕ in conjunctive normal form such that each clause in ϕ contains **exactly k literals**, find an assignment that maximizes the number of satisfied clauses.

Adapt the randomized Max-SAT algorithm from the lecture to this problem. What (expected) approximation ratio do you obtain in terms of k?

Exercise 2 *(5 Points)* Consider a coin that will turn up heads with probability p when flipped. We repeatedly flip the coin until it turns up heads. Prove that the expected number of coin flips is $\frac{1}{p}$.

Exercise 3 *(5 Points)* Assume you are given 50 red and 50 blue balls. You are free to distribute them freely into two urns. Afterwards, the Joker will choose an urn uniformly at random. You then select a ball uniformly at random from the chosen urn. The Joker wins if this balls is red, while you will win if it is blue. What is the maximum probability with which you can win the game?

Exercise 4 *(7 points)* Consider the weighted Max-Cut problem: Given an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{N}$, the task is to determine a *cut* $C \subseteq V$ such that the weight $\sum_{\substack{e=\{i,j\}\in E\\ i\in C, j\notin C}} w(e)$ of the cut edges is maximized. Give a sampling-based algorithm that achieves an expected approximation ratio of $1/2$.

Exercise 5 *(18 Points)* Recall the unweighted version of VertexCover: given a graph $G = (V, E)$, determine a set $C \subseteq V$ of minimum cardinality such that each edge $\{i, j\} \in E$ is *covered* by C, i.e., we have $i \in C$ or $j \in C$.

We define two different greedy algorithms, *randomized greedy* and *bad-decision greedy*: Both algorithms receive as input the graph G represented by a list of its edges $e_1 = \{i_1, j_1\}, \dots, e_m = \{i_m, j_m\}$ – note that the order of endpoints of an edge (i.e., whether we represent an edge as

$\{i, j\}$ or $\{j, i\}$) is given by the input (i.e., chosen by an adversary). The pseudocode of both algorithms is given below.

Algorithm: randomized greedy

```
C := ∅
for k = 1, ..., m do
   consider the k-th edge e_k = {i, j}
   if e_k is not covered by C then
     choose r u. a. r.[a] from {i, j}
     C ← C ∪ {r}
return C
```

[a] uniformly at random

Algorithm: bad-decision greedy

```
C := ∅
for k = 1, ..., m do
   consider the k-th edge e_k = {i, j}
   if e_k is not covered by C then
     C ← C ∪ {i}
return C
```

a) *(6 Points)* Analyze how randomized greedy performs on the n-star graph. (The n-star-graph is the graph consisting of a center vertex c and $n-1$ leaf vertices, where each leaf vertex has a single adjacent edge to c.) What is the expected size of C?
Hint: You may make use of Exercise 2.

b) *(7 Points)* Prove that randomized greedy has an expected approximation ratio of at most 2 on general graphs.
Hint: Use a). In particular, consider an optimal vertex cover of size OPT and view G as a union of OPT-many star graphs.

c) *(5 Points)* Determine the approximation ratio of bad-decision greedy.
*Note: Don't forget to prove both upper **and** lower bound on the approximation ratio!*