Advanced Algorithms 2020 10/29, 2020

Exercise 07

Lecturer: Mohsen Ghaffari *Teaching Assistang: Vaclav Rozhon*

1 Minimum bisection cut (leftover)

A *bisection cut* is a cut (S, S') such that $|S| = |S'| = n/2$. An *r-balanced cut* is a cut where $r \cdot n \leq |S| \leq (1-r) \cdot n$. A size of a cut is the number of edges that go across the cut.

Give a polynomial-time algorithm that, given a graph G as input, outputs a 1/3-balanced cut whose size is $\mathcal{O}(\log n)$ factor from the size of the smallest-size bisection cut of G.

Hint:

Find a black box reduction to the result you saw in the lecture via a greedy algorithm.

2 Minimum bisection cut on a tree [Recommended]

You are given a weighted tree T on n vertices. Assume that n is even. Your task is to give a polynomial time algorithm that finds the smallest bisection cut of T. That is, find a set of vertices $S \subseteq V(T)$ with $|S| = n/2$ such that the weight of the edges between S and $V(T) \setminus S$ is smallest possible.

Recall that you have seen in the lecture that this implies a polynomial time algorithm for $O(\log n)$ approximation of the bisection cut in general graphs.

3 One tree is not enough

Find an unweighted graph G with the following property. For any subgraph $T \subseteq G$ which is also a tree, one can find a cut $S \subseteq V(G)$ such that the number of edges of G that goes across the cut is at least $\Omega(n)$ times more than the number of edges of T that goes across the cut.