1 Background

Definition 1. *An* α-approximation *algorithm for a minimization problem is a polynomial time algorithm which finds a solution of cost* $\leq \alpha|OPT|$ *for each instance of the problem.*

2 Approximation: Problems and Exercises

2.1 Vertex Cover

Definition 2. *A* vertex cover *of a graph* $G = (V, E)$ *is a set of vertices* $X \subseteq V$ *such that for every edge* $e \in E$, $e = \{u, v\}$, *at least one endpoint of* e *lies in* X *(that is,* $u \in X$ *or* $v \in X$*).*

The natural associated optimization problem is defined as follows.

Vertex Cover
Input: Graph $G = (V, E)$
Question: Find a minimum vertex cover of G

Given in class: Vertex Cover has a 2-approximation.

Problem 1. *Consider the greedy algorithm, which in every step includes a vertex of largest degree from the graph. Show an instance where this algorithm can be worse than a 2-approximation. (In fact, there are instances where the greedy algorithm is only a* $\mathcal{O}(\log n)$*-approximation.)*

2.2 Set Cover

The Set Cover problem is a generalization of Vertex Cover, defined as follows.

Set Cover
Input: A set system: A set U, and a set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of U.
Question: Find a minimum set cover of $\mathcal{S}$: a minimum subset X of U such that $X \cap S_i \neq \emptyset$ for every $S_i \in \mathcal{S}$.

Given in class: Set Cover has an $(\ln m)$-approximation.

Problem 2. *Find a hard instance for the greedy algorithm (that is, an instance where the resulting approximation ratio is close to* $\log n$*).*

Problem 3. *Show that the algorithm performs better when OPT is large: Let* $|OPT| = k$*. Show that the greedy algorithm finds a solution of size* $k \ln \frac{m}{k} + k$*. (*Hint: *What happens when there are only* k *uncovered sets left in* $\mathcal{S}$*?)*

2.3 Feedback Vertex Set

Definition 3. *A* feedback vertex set *in a graph* $G = (V, E)$ *is a set of vertices* X *which intersects every cycle in* G *(i.e., so that* $G - X$ *is acyclic).*

Feedback Vertex Set
Input: A graph $G = (V, E)$
Question: Find a minimum feedback vertex set in G.

Definition 4. *A* tournament *is a directed graph* $G = (V, E)$ *where for every pair of vertices* $u, v \in V$ *either* $(u, v) \in E$ *or* $(v, u) \in E$ *(but not both).*

Problem 4. *Show a 3-approximation for* Feedback Vertex Set in Tournaments.

Problem 5. *Show a* $\mathcal{O}(\log n)$*-approximation for* Feedback Vertex Set. Hint*: Try to reach a case where you can show that there is a short cycle.*

Remark 1. *The* Feedback Vertex Set *problem has a 2-approximation in general graphs (in fact, even in directed graphs).*

2.4 Traveling Salesman (TSP)

The most general variant of TSP is as follows.

Traveling Salesman
Input: A complete graph $G = (V, E)$ with a length $\ell(u, v)$ associated with every edge.
Question: Find a cycle through all vertices V of minimum total length.

Given in class: This variant of TSP has no approximation ratio $f(n)$ for any function of $n = |V|$.

To make progress, we need to make an assumption on the length function ℓ. A *metric* on V is a function $d(u, v)$ satisfying (i) $d(u, v) \geq 0$ for all $u, v \in V$; (ii) $d(u, v) = 0$ if and only if $u = v$; and (iii) the *triangle inequality*: $d(u, w) \leq d(u, v) + d(v, w)$ for all $u, v, w \in V$.

We have the following variant.

Metric TSP
Input: A complete graph $G = (V, E)$ with a length $\ell(u, v)$ associated with every edge, where ℓ is a metric and $\ell(u, v) = \ell(v, u)$ for all u, v.
Question: Find a cycle through all vertices V of minimum total length.

Given in class: Metric TSP has a 2-approximation (using a doubled minimum spanning tree).

Problem 6. *Try to improve the above; the goal is a* 1.5*-approximation.* Hint: *Is the "doubling" step really the best way to ensure a solution?*

You will need to know about Eulerian graphs for this one. The solution will be given in class.

Asymmetric TSP
Input: A complete graph $G = (V, E)$ with lengths $\ell(u, v)$ and $\ell(v, u)$ associated with every edge, where ℓ is a metric.
Question: Find a cycle through all vertices V of minimum total length.

Given in class: Asymmetric TSP has a $\log_2 n$-approximation. It uses the following:

Problem 7. *A* cycle cover *of a directed graph G is a collection of vertex-disjoint directed cycles (which are subgraphs of G) which together cover every vertex of G. (Note that cycles of just two edges are also allowed.) The* cost *of a cycle cover is the total length of its edges. Show that a min-cost cycle cover can be found in polynomial time.* Hint: *Try to use the (polynomial-time) algorithm for weighted bipartite matching.*

Remark 2. *For* Metric TSP*, the 1.5-approximation is called* Christofides' algorithm*; although it is from 1976, it is still the best result known. (A* 4/3*-approximation has been conjectured, but not proven.) For* Asymmetric TSP*, a slight improvement is known: a $\mathcal{O}(\log n / \log \log n)$-approximation. It is generally believed that a constant factor approximation should exist.*

2.5 Binpacking / Knapsack

Knapsack
Input: A collection of items I where every item $x \in I$ has a *weight* $w(x)$ and a *value* $f(x)$.
Question: Find a set of items of total weight at most one, with maximum total value.

Problem 8. *A seemingly reasonable approach would be to try to pack the "most profitable" objects first - for example the ones with the highest "unit profit", that is the ratio of value to weight. Show that the greedy algorithm, which always tries to take the most profitable object which still fits can be arbitrarily bad, i.e. it can be arbitrarily far away from the optimal solution.*

Problem 9. *There is a slightly silly remedy for our worst-case example in the previous problem: select either the set $a_1, ..., a_k$ of the most profitable objects filling the backpack, or a_{k+1}, that is the first item which did not fit. Show that this is actually a 2-approximation.*

Problem 10. *Can you find a polynomial algorithm assuming that the weights and the values of all items are small, that is polynomial in n?*

Remark 3. *If our problem instance contains some numerical values, like costs, lengths of the edges etc. we usually assume that they are written in binary and we expect a polynomial time algorithm to run in time polynomial in the lengths of these binary representations, and not in the actual values. An algorithm running in time polynomial in terms of the values, that is in time exponential in terms of the lengths of the binary representations, is called* pseudopolynomial.

Problem 11. *Our goal now is obtaining a* $(1+\epsilon)$*-approximation algorithm for* Knapsack *running in time polynomial in* n *and* $\frac{1}{\epsilon}$ *(such algorithms are called* FPTAS: fully polynomial time approximation schemes*).* Hint: *You can try to reduce the precision of the values in our instance (i.e. ignore less important bits) and somehow use the previous algorithm.*