Problem Set for Advanced Algorithms - Part 2

(Vijay Vazirani “Approximation Algorithms”)

1.1 Give a factor $1/2$ algorithm for the following.

Problem 1.9 (Acyclic subgraph) Given a directed graph $G = (V, E)$, pick a maximum cardinality, set of edges from E so that the resulting subgraph is acyclic.

Hint: Arbitrarily number the vertices and pick the bigger of the two sets, the forward-going edges and the back-going edges. What scheme are you using for upper bounding OPT?

Solution:

Algorithm: Arbitrarily enumerate the vertices. Let $S_{forward-going}$ the subset of edges v_iv_j s.t. $i < j$. Let $S_{back-going}$ the subset of edges v_iv_j s.t. $i > j$. Let $S_sol = S_{forward-going}$. Return $S_{forward-going}$ if $|S_{forward-going}| > |S_{back-going}|$, otherwise return $S_{back-going}$.

Clearly neither $S_{forward-going}$ nor $S_{back-going}$ has a cycle because it is ordered ascending and descending respectively. Then, the algorithms returns an acyclic subgraph.

Any edge v_iv_j in the graph are in one of the sets, because $i > j$ or $i < j$. Then the greatest set must have cardinality grater or equal $|G|/2$, otherwise its is not the greatest set.

Moreover, clearly $|G| \geq |OPT|$.

Let S be the solution returned by the algorithm, then

$$|S| \geq \frac{1}{2}|G| \geq \frac{1}{2}|OPT|$$

.

In conclusion, the algorithm works and has factor $1/2$. ■

1.2 Design a factor 2 approximation algorithm for the problem of finding a minimum cardinality maximal matching in an undirected graph. **Hint:** Use the fact that any maximal matching is at least half the maximum matching.

Solution:

Algorithm: let S and empty set. For each edge v_iv_j in the graph, if there is no edge in S containing v_i nor v_j, add v_iv_j to $|S|$. Return S.

Clearly the algorithm returns a maximal matching.

Now, lets prove that it is a factor 2 approximation algorithm.

Let M being a maximum matching, clearly $|S| \leq |M|$, otherwise M is not maximum.

Let OPT being the optimal solution. Since any maximal matching is at least half the maximum matching then we have that $|OPT| \geq \frac{1}{2}|M| \Rightarrow 2|OPT| \geq |M|$.

Putting everything together e have $|S| \leq |M| \leq 2|OPT|$.

In conclusion, the algorithm works and has a factor 2. ■

1.3 (R. Bar-Yehuda) Consider the following factor 2 approximation algorithm for the cardinality vertex cover problem. Find a depth first search tree in the given graph, G, and output the set, say S, of all the nonleaf vertices of this tree. Show that S is indeed a vertex cover for G and $|S| \leq 2 \cdot OPT$. **Hint:** Show that G has a matching of size $\lceil |S|/2 \rceil$.

Solution:

A DFS tree contain all vertices, then clearly it is a vertex cover. Now let's prove that if we remove the leaves of the tree, it is still a vertex cover, which means that the solution of the provided algorithm is a vertex cover.

Let T is the DFS tree constructed by the algorithm. Let S being the solution returned by the algorithm and suppose bwoc that it is not a vertex cover. Then there is a vertex, say v, s.t. an edge vu is not covered.

Clearly u cannot be an nonleaf vertex of T, because the nonleaves vertices are in S and in this way vu is covered by u.

If u is also a leaf in the tree, then, since T is a DFS, or v would be a connected to u in the tree or u would be a connected to v. Then u cannot be an leaf vertex.

Then u is not a vertex in G and such an edge vu cannot exist. $\rightarrow\leftarrow$.

We proved that the algorithm provides a correct answer S, now let's prove $|S| \leq |OPT|$.

Construct a matching M in this way: enumerate the layers of the tree from 1 to n. Let S_{odd} being the non leaves vertices in odd layers and S_{even} being the non leaves vertices in even layers. For each vertex v in S_{odd} we add to M an edge vu where u is a child of v. In this way the number of edges in the matching is equal to $|S_{odd}|$. Since our layers enumeration starts with an odd number, then excluding the leaves, the number of vertices in odd layers are grater or equal than the number of vertices of even layers, in other words $|S_{odd}| \geq |S_{even}|$. Then it follows that $|M| \geq \frac{1}{2}|S|$.

For each edge in M we must have a vertex in OPT, otherwise this edge is not covered by OPT. Moreover, since M is a matching, all theses vertices are different. Then we have that $|OPT| \geq |M|$.

Putting everything together we have that $|OPT| \geq |M| \geq \frac{1}{2}|S| \Rightarrow |S| \leq 2|OPT|$.

In conclusion, the algorithm works and has a factor 2. ■

2.1 Given an undirected graph $G = (V, E)$, the *cardinality maximum cut problem* asks for a partition of V into sets S and $\bar{S}$ so that the number of edges running between these sets is maximized. Consider the following greedy algorithm for this problem. Here v_1 and v_2 are arbitrary vertices in G, and for $A \subset V$, $d(v, A)$ denotes the number of edges running between vertex v and set A.

Algorithm
1. Initialization:
   $A \leftarrow \{v_1\}$
   $B \leftarrow \{v_2\}$
2. For $v \in V - \{v_1, v_2\}$ do:
   if $d(v, A) \geq d(v, B)$ **then** $B \leftarrow B \cup \{v\}$,
   else $A \leftarrow A \cup \{v\}$
3. Output A and B

Show that this is a factor $1/2$ approximation algorithm and give a tight example. What is the upper bound on OPT that you are using? Give examples of graphs for which this upper bound is as bad as twice OPT. Generalize the problem and the algorithm to weighted graphs.

Solution:

Let E_A being the edges between vertices of A, E_B being the edges between vertices of B and E_{AB} being the edges with one end in A and other end in B.

Every time the algorithm add a vertex in A it add a certain number of edges n in E_A, that can be zero, and a number m of edges in E_{AB}. By construction $m > n$.

Every time the algorithm add a vertex in B it add a certain number of edges n in E_B, that can be zero, and a number m of edges in E_{AB}. By construction $m \geq n$.

Putting everything together for each edge we add in $E_A \cup E_B$ we for sure have an edge in E_{AB}. Then $|E_A| + |E_B| \leq |E_{AB}|$.

Since $|E| = |E_A| + |E_B| + |E_{AB}|$, then we have that $|E| \leq 2|E_{AB}| \Rightarrow |E_{AB}| \geq \frac{1}{2}|E|$.

Clearly $OPT \leq |E|$ then we can say that this is a factor $1/2$ approximation algorithm.

Can it be better? Consider the graph $G = (\{v_1, v_2, v_3\}, \{v_1v_3, v_2v_3\})$. Then the algorithm will return 1 while the OPT is 2. So, it cannot be better.

A generalization of the algorithm to work with weighted graphs is to consider $d(v, S)$ as the **sum of weights of edges** running between vertex v and set S instead of the **number of edges** running between vertex v and set S. ■

2.2 Consider the following algorithm for the maximum cut problem, based on the technique of *local search*. Given a partition of V into sets, the basic step of the algorithm, called *flip*, is that of moving a vertex from one side *solution* under the flip operation, i.e., a solution which cannot be improved by a single flip.

The algorithm starts with an arbitrary partition partition of V. While there is a vertex such that flipping it increases the size of the cut, the algorithm flips such a vertex. (Observe that a vertex qualifies for a flip if it has more neighbors in its own partition than in the other side.) The algorithm terminates when no vertex qualifies for a flip. Show that this algorithm terminates in polynomial time, and achieves an approximation guarantee of $1/2$.

Solution:

Since the flip operation is only performed if it increases the size o the cut and the size of the cut is bounded by |E|, then the amount of times a flip operation can be performed is at most |E|. So, the algorithms terminates in a polynomial time.

Take an edge e outside the cut, then both of its ends are in the same partition, say A. Claim that both of ends of this edges must have at least one neighbor in the other partition, say B.

Suppose bwoc that v is an end of e with no neighbor in B, then we flip v. The size of the cut increased by $|S|$ with S the set of neighbors of v that are in A. Since $e \in S$, then $|S| \geq 1$. Then we flip v and increase the cut. But since the algorithm terminates, there is no such vertex as V. $\rightarrow\leftarrow$

Let A and B being the partitions after the algorithm terminates s.t. $|E_A| \geq |E_B|$, with $|E_A|$ and $|E_B|$ the edges with both ends at A and at B respectively. Then $|E_B| \leq |E_A|$.

Let S the edges in the cut. By the claim, for each edge in $|E_A|$ we must have at least two edges in S. In other words $2|E_A| \leq |S|$.

Putting everything together we have that $|E| = |E_A| + |E_B| + |S| \leq 2|E_A| + |S| \leq 2|S| \Rightarrow |S| \geq \frac{1}{2}|E|$.

Clearly $OPT \leq |E|$ then we can say that this is a factor $1/2$ approximation algorithm. ■

2.3 Consider the following generalization of the maximum cut problem.

Problem 2.14 (MAX k-CUT) Given an undirected graph $G = (V, E)$ with nonnegative edge costs, and an integer k, find a partition of V into sets $S_1, \ldots, S_k$ so that the total cost of edges running between these sets is maximized.

Give a greedy algorithm for this problem that achieves a factor of $\left(1 - \frac{1}{k}\right)$. Is the analysis of your algorithm tight?

Solution:

Algorithm
1. Initialization:
   $S_1, S_2, \ldots, S_k \leftarrow \{v_1\}, \{v_2\}, \ldots, \{v_k\}$.
2. For each $v \in V - \{v_1, v_2, \ldots, v_k\}$
   Select S_i s.t. it maximizes the number of edges from v to vertices in $\bigcup_{\substack{j=0 \\ j \neq i}}^{k} S_j$.
   $S_i = S_i \cup v$.
3. Output $S_1, S_2, \ldots, S_k$.

Now let's prove this algorithm achieves a factor of $\left(1 - \frac{1}{k}\right)$.

For each vertex v we add in S_i, we have a certain number n of edges between v and other vertices of S_i, n can be zero, and a number m of edges between v and vertices in S_j, with $j \neq i$. By construction, $n \geq m$, otherwise v would be added to S_j instead. Then for each edge with both ends in the same partition we must have at least $k-1$ edges in the cut.

Let S being the edges with endpoints in different partitions and E_i the edges with both endpoints in partition S_i. Then for each edge e in E_i, $i = 1, 2, \ldots, k$, we must have at least $k-1$ edges in S, as we showed above. In other words $(k-1)|E_1| + (k-1)|E_2| + \ldots + (k-1)|E_k| \leq |S| \Rightarrow |E_1| + |E_2| + \ldots + |E_k| \leq \frac{1}{k-1}|S|$.

Finally, we have the following

$$
\begin{aligned}
|E| = |E_1| + |E_2| + \ldots + |E_k| + |S| &\leq \frac{1}{k-1}|S| + |S| = \frac{k}{k-1}|S| \\
&\Rightarrow |E| \leq \frac{k}{k-1}|S| \\
&\Rightarrow |S| \geq \left(1 - \frac{1}{k}\right)|E|.
\end{aligned}
$$

Since $OPT \leq |E|$, the algorithm achieves a factor of $\left(1 - \frac{1}{k}\right)$.

Can it be better? Consider the graph $G = (\{v_1, v_2, v_3\}, \{v_1v_3, v_2v_3\})$ and let $k = 2$. Then the algorithm will return 1 while the OPT is 2. In this case $\left(1 - \frac{1}{k}\right) OPT = \left(1 - \frac{1}{2}\right) 2 = 1$, then it cannot be better. Then the analysis is tight. ■

2.14 A *tournament* is a directed graph $G = (V, E)$, such that for each pair of vertices, $u, v \in V$, exactly one of (u, v) and (v, u) is in E. A *feedback vertex set* for G is a subset of the vertices of G whose removal leaves an acyclic graph. Give a factor 3 algorithm for the problem of finding a minimum feedback vertex set in a directed graph. **Hint:** Show that it is sufficient to "kill" all length 3 cycles. Use the factor f set cover algorithm.

Solution:

Algorithm(G=(V,E))
1. Inicialization: $E' = E$.
2. For each u in V
  For each v in V
    For each w in V
      if u, v, w forms a length 3 cycle C then
        $E' = E' - E(C)$.
3. Output $G' = (V, E')$.

Claim: G' has no cycles.

We prove by induction.

Induction step: if G' has a cycle of size n then G has a cycle of size $n-1$. Suppose G' has a cycle of size n: $v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1$. We know that (v_3,v_1) or (v_1,v_3) were in the original graph. In the first case v_1v_2, v_2v_3, v_3v_1 was a cycle of length 3 (not possible since we killed all the length 3 vertex). Then $v_1v_3, v_3v_4, v_4v_5, \ldots, v_{n-1}v_n, v_nv_1$ was a cycle of length $n-1$.

Last step: if G' has a cycle of size 4 then G has a cycle of size 3.

Then G' has no cycle at all and the algorithm outputs a correct answer.

Now let's prove the answer is at most $3 \cdot OPT$.

Each time we find a length 3 cycle we delete all the edges of this cycle. Then for sure the next length 3 cycle we find will not contain any of these edges. Then if running the algorithm we find n length 3 cycles with no edges in common, we deleted $3n$ edges. Moreover, since the cycles are not adjacent, in the optimal solution we must remove at least one edge for each one of these length 3 cycles, then it follows that $OPT \geq n$. Putting everything together we have that $3OPT \geq$ the number of edges we deleted.

In conclusion the algorithm works and has factor 3. ■

3.4 (Hoogeveen [137]) Consider variants on the metric TSP problem in which the object is to find a simple path containing all the vertices of the graph. Three different problems arise, depending on the number (0, 1 or 2) of endpoints of the path that are specified. Obtain the following approximation algorithms.

- If zero or one endpoints are specified, obtain a $3/2$ factor algorithm.
- If both endpoints are specified, obtain a $5/3$ factor algorithm.

Solution: