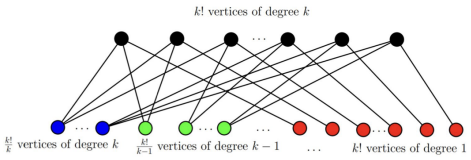
School of Computer Science and Technology
University of Science and Technology of China

Exercise Sheet 3 for
Design and Analysis of Algorithms
Autumn 2022

Due 28 Oct 2022 at 16:59

Exercise 1 *30*

Consider the algorithm AnotherGreedyVC given in Lecture 8 for the minimum vertex cover problem. Use the following example to show that the approximation ratio of AnotherGreedyVC is $\omega_n(1)$, where n is the number of vertices in the input graph.



Solution 1:

Proof:

We firstly aim to prove that in this specific example, the algorithm **'ANOTHERGREEDYVC'** would select $k!$ vertices in the best case, and $k!(1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k})$ vertices in the worst case correspondingly.

Proof of best case: Note that there are in total $k!\cdot k$ edges in this example, and the max-degree of each vertice is at most k, which means the total-number of edge would decrease at most k if we remove a single vertex from the graph. So the number of vertexs selected by the algorithm $\geq \frac{k\cdot k!}{k} = k!$.
Here we provide a specific solution $V_{sel} = \{$**all of the black vertices**$\}$ whose size is $k!$.

Proof of worst case: Note that there are in total $k!\cdot k$ edges in this example. And once we remove a single vertices v from the original graph, the total-number of edges in the graph will decrease $degree(v)$.
Notice the fact that
the algorithm would never select a specific black vertice and its corresponding red one(direct conjunct with it through an edge.). So if there is a vertices-set outputed by the algorithm containing some black vertices, we can always know that there are the same number of red vertives not appeared in the outputed solution.
So we can suppose that **the worst case happens at a situation when the algorithm not selected anyone of the black vertices**.
And this suggests that the number of vertexs selected by the algorithm $\leq \Sigma_{i=1}^{k} i\cdot\frac{k!}{i} = k!\cdot k$.
Here we provide a specific solution which is also a possible result outputed by the algorithm : V_sel = $\{$**blue vertices**, **green vertices**, ... , **red vertices**$\}$ and whose size equals $k!(1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k})$.

Proof of the approximation-rate:

All we need to do is to calculate $\frac{size(\textbf{worst solution})}{size(\textbf{best solution})} = \frac{k!(1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k})}{k!} = 1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k}$
First we need to clarify the relationship between n and k, and let H_k represent the sum of first k term.

$$n = (1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k}+1)\cdot k! = (H_k+1)\cdot k!$$
$$logk+2)\cdot k! < (k+1)\cdot k! = (k+1)! < (k+1)^{k+1}$$
$$\Rightarrow logn < (k+1)log(k+1) < (k+1)^2$$
$$\Rightarrow k > \sqrt{logn} - 1 \Rightarrow k = \Omega(logn)$$

We know the approximation ratio of this algorithm 'ANOTHERGREEDYVC' is
$\frac{size(\textbf{worst solution})}{size(\textbf{best solution})} = \frac{k!(1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k})}{k!} = 1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k} = H_k > logk = \Omega(loglogn) = \omega_n(1)$
And this is the end of the proof.

Exercise 2 *30*

Consider the set cover problem. Let U be a set of n elements. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a collection of subsets of U such that $\cup_{i=1}^{m} S_i = U$. Our goal is to select as few subsets as possible from $\mathcal{S}$ such that their union covers U.
Consider the following algorithm SetCover for this problem. The algorithm takes as input U and $\mathcal{S}$, and does the following:

(a) Initialize $C = \emptyset$.

(b) While U contains elements not covered by C:

(i) Find the set S_i containing the greatest number of uncovered elements

(ii) Add S_i to C.

To analyze the above algorithm, let $k = \text{OPT}$ be the number of sets in the optimal solution. Let $E_0 = U$ and let E_t be the set of elements not yet covered after step t.

(a) Show that $|E_{t+1}| \leq |E_t| - |E_t|/k$.

(b) Show that the algorithm SetCover is a $(\ln n)$-approximation algorithm for the set cover problem.

Hint: Show that the algorithm SetCover finishes within $\text{OPT} \cdot \ln n$ steps.
Solution 2:
Proof of (a):
Since the best solution **OPT** = k, there must exist $S_{i_1}, S_{i_2}, ..., S_{i_k}$ so as to $S_{i_1} \cup S_{i_2} \cup ... \cup S_{i_k} = U$, where each element in the $\{i_1, i_2, ..., i_k\}$ is not equal with each other and each element in them $\in \{1, 2, ..., m\}$.
Now we are considering E_t(which is the set of elements not yet covered after step t):
(1):$|\mathbf{E_t}| = 0$. This means the algorithm is terminated right after step t. for there is no point to keep on considering $|E_{t+1}|$, we can regard this situation satisfies the conclusion we are proving.
(2):$|\mathbf{E_t}| > 0$. There must exist some set $S_{i_j}, 1 \leq j \leq k$, which have not selected by algorithm **SETCOVER**, for E_t is not yet covered all of the elements $\in U$.
Define $G = \{g | g \in \{S_{i_1}, S_{i_2}, ..., S_{i_k}\}$ *and g has not been selected by* **SETCOVER** *after step t*$\}$.
It is obvious that $G \subseteq \{S_{i_1}, S_{i_2}, ..., S_{i_k}\}$, so we have that $|G| \triangleq k' \leq |\{S_{i_1}, S_{i_2}, ..., S_{i_k}\}| = k$.(we denote that k' represent the $|G|$, so $G = \{g_1, g_2, ..., g_{k'}\}$.)
According to the definition of G, we know that $(g_1 \cap E_t) \cup (g_2 \cap E_t) \cup ... \cup (g_{k'} \cap E_t) = E_t$. So $\Sigma_{i=1}^{k'} |(g_i \cap E_t)| \geq |\cup_{i=1}^{k'} (g_i \cap E_t)| = |E_t|$
According to the **pigeonhole principle**, we know that exist m, $1 \leq m \leq k'$, so as to $|g_m \cap E_t| \geq \frac{|E_t|}{k'} \geq \frac{|E_t|}{k}$.
Suppose that at step t+1, we select set x, which has not selected before. According to the description of algorithm **SETCOVER**, $|x \cap E_t| \geq |g_m \cap E_t| \geq \frac{|E_t|}{k}$.
So we know that: $|E_{t+1}| = |E_t| - |x \cap E_t| \leq |E_t| - \frac{|E_t|}{k}$. And this is the end of the proof.
Proof of (b):
According to the conclusion of (a), we have that $|E_{t+1}| \leq (1 - \frac{1}{k}|E_t|) = (1 - \frac{1}{k}|E_t|)$.
We aim to prove that the algorithm **SETCOVER** terminates within $\leq k \cdot ln(n)$ steps.
Let $M = \lceil k \cdot ln(n) \rceil$. Suppose that the algorithm execute for M steps while it still do not terminate.
After M steps, the number of uncovered elements $|E_{M+1}|$ meets that:

$$|E_{M+1}| \leq (1 - \frac{1}{k})|E_M| \leq ... \leq (1 - \frac{1}{k})^{M+1}|E_0| = (1 - \frac{1}{k})^{M+1}|U| = (1 - \frac{1}{k})^{M+1} \cdot n$$

For $M + 1 > k \cdot ln(n)$, we have that the upperbound $(1 - \frac{1}{k})^{M+1} \cdot n \leq (1 - \frac{1}{k})^{k \cdot ln(n)} \cdot n$
$= (\frac{1}{(1+\frac{1}{k-1})^k})^{lnn} \cdot n$**(*)**
Note that $f(k) = (1 + \frac{1}{k-1})^k$ is monotone decreasing, So we have that $(1 + \frac{1}{k-1})^k > \lim_{k \to \infty}(1 + \frac{1}{k-1})^k = e$.
So, previous **(*)** meets that : **(*)**$< (\frac{1}{e})^{lnn} \cdot n = 1$.

This means that $|E_{M+1}| \leq (1-\frac{1}{k})^{M+1} \cdot n < 1$. So $|E_{M+1}| = 0$.
And this contradicts the supposed "the algorithm execute for M steps while it still do not terminate."
So we successfully prove that the algorithm **SETCOVER** terminates within $\leq k \cdot ln(n)$ steps.

Exercise 3 *40*
Consider the max cut problem. Given an undirected n-vertex graph $G = (V, E)$ with positive integer edge weights w_e for each $e \in E$, find a vertex partition $(A, \bar{A})$ such that the total weight of edges crossing the cut is maximized, where $\bar{A} = V \setminus A$ and the weight of $(A, \bar{A})$ is defined to be $w(A, \bar{A}) := \sum_{u \in A, v \in \bar{A}} w_{uv}$.

Consider the following algorithm MaxCut.

(a) Start with an arbitrary partition of V.

(b) Pick a vertex $v \in V$ such that moving it across the partition would yield a greater cut value.

(c) Repeat step (b) until no such v exists.

Now analyze the performance guarantee of the algorithm.

(a) Suppose that the maximum edge weight is $\lceil n^{10} \rceil$. Show that the algorithm runs in polynomial time.

(b) Let $(S, \bar{S})$ be partition output by the algorithm MaxCut. Show that for any vertex $v \in S$, it holds that

$$\sum_{u \in \bar{S}, (u,v) \in E} w_{u,v} \geq \frac{1}{2} \sum_{u:(u,v) \in E} w_{u,v}$$

(c) Show that the algorithm is MaxCut is a 1/2-approximation algorithm for the max cut problem.

Hint: Use the fact that $\sum_{e \in E} w_e \geq$ OPT, where OPT is the total weight of the optimal solution.
Solution 3:
Proof of (a):
Note that this algorithm 'MAXCUT' is a local-search algorithm, making the cut value is strictly increasing monotonically with the iteration.
Consider the following set: $S = \{|\Sigma_{m \in B(v)} \omega_{mv} - \Sigma_{n \in B'(v)} \omega_{nv}| | v \in V;$
$C(v)$ *means the set of vertices which connected with* v; $B(v) \subseteq V \backslash \{v\};$
$B'(v) = V \backslash \{v \cup B(v)\}\}$
It is obvious that the number of element in S is finite. We denote that the minimum positive number in S is δ.
So we know that the minimum update step of cut-value is at least δ (constant). Then we only need to prove that the maximum possible cut-value is a polynomial of n.
Suppose that the graph G is a n vertices fully-connected graph, which means there is an edge between any two vertices. And we further suppose that the weight of each edge is the maximum edge weight $\lceil n^{10} \rceil$.
Suppose that the maxcut of G is $(A, \bar{A})$, and there are x vertices in the A and $n - x$ vertices in the A.
The corresponding cut-value is $x(n-x) \cdot \lceil n^{10} \rceil$, which is at most $\frac{n^2}{4} \cdot \lceil n^{10} \rceil$, when $x = n - x = \frac{n}{2}$
So the result outputted by the algorithm is at most $\frac{n^2}{4} \cdot \lceil n^{10} \rceil$, and this suggests that the largest running time of this algorithm is $\frac{\frac{n^2}{4} \cdot \lceil n^{10} \rceil}{\delta} = O(n^{12})$.
This is the end of the prove.

Proof of (b):

$$\sum_{u \in \bar{S}, (u,v) \in E} w_{u,v} \geq \frac{1}{2} \sum_{u:(u,v) \in E} w_{u,v}$$
$$\Leftrightarrow \sum_{u \in \bar{S}, (u,v) \in E} w_{u,v} \geq \sum_{u \in S, (u,v) \in E} w_{u,v} (*)$$

Let us denote the current cut-value $= X$, which is a local-minima outputted by the algorithm 'MAX-CUT'. We make the assumption that the inequality (*) does not exists, we have that $\sum_{u\in\bar{S},(u,v)\in E} w_{u,v} < \sum_{u\in S,(u,v)\in E} w_{u,v}$.
we can pick v out and move it across the cut to $\bar{(S)}$,
then the cut-value $= X - \sum_{u\in\bar{S},(u,v)\in E} w_{u,v} + \sum_{u\in S,(u,v)\in E} w_{u,v} > X$.
And this is contradicting the fact that X is a local-minima.
So the assumption we made does not exists $\Rightarrow \sum_{u\in\bar{S},(u,v)\in E} w_{u,v} \geq \sum_{u\in S,(u,v)\in E} w_{u,v}$
$\Leftrightarrow \sum_{u\in\bar{S},(u,v)\in E} w_{u,v} \geq \frac{1}{2}\sum_{u:(u,v)\in E} w_{u,v}$. And this is the end of the prove.

Proof of (c):

Let us denote that $(S, \bar{(S)})$ is a partition output by the algorithm 'MAXCUT'.

According to the conclusion of (b) :

for $\forall v \in S$, *it holds that* :

$$\sum_{u\in\bar{S},(u,v)\in E} w_{u,v} \geq \frac{1}{2}\sum_{u:(u,v)\in E} w_{u,v}$$

Similarly, for $\forall v \in \bar{S}$, *it holds that* :

$$\sum_{u\in S,(u,v)\in E} w_{u,v} \geq \frac{1}{2}\sum_{u:(u,v)\in E} w_{u,v} ...(2)$$

Considering all of the v *and do summation, we have that* :

$$\sum_{v\in S}\sum_{u\in\bar{S},(u,v)\in E} w_{u,v} \geq \sum_{v\in S}\frac{1}{2}\sum_{u:(u,v)\in E} w_{u,v} ...(1)$$

$$\sum_{v\in\bar{S}}\sum_{u\in S,(u,v)\in E} w_{u,v} \geq \sum_{v\in\bar{S}}\frac{1}{2}\sum_{u:(u,v)\in E} w_{u,v} ...(2)$$

$(1) + (2)$, *we have that* :

$$2\cdot\sum_{v\in\bar{S}}\sum_{u\in S,(u,v)\in E} w_{u,v} \geq \frac{1}{2}\cdot\sum_{v\in V}\sum_{u:(u,v)\in E} w_{u,v} = \sum_{e\in E} w_e$$

$$\Rightarrow \sum_{v\in\bar{S}}\sum_{u\in S,(u,v)\in E} w_{u,v} \geq \frac{1}{2}\cdot\sum_{e\in E} w_e \geq \frac{1}{2}\cdot OPT$$

This is the end of the proof!
