Stav Yosef, 316298876

Advanced Algorithms - Homework 2

Question 1

First, we need to find the biggest size of vertices forming a maximal independent set (MIS), note this size by **k**.
In order to find k we can perform a binary search in the range [1,n] $(n = |V|)$ using the function A_d that runs in polynomial time.
Note - The first TRUE that A_d returns is not enough, we need to keep searching for the biggest.
Runtime to find the best k: $\log_2(n) \cdot T(A_d)$

A_o:
After we know the value of k we can start the second part of the solution, run n iterations and each iteration we'll delete a vertex and check with $A_d(G', k)$ if there is still MIS by the size of k and if the graph has k left vertices we can quit the loop and return the graph, if there is not MIS by the size of k (the decision function returns FALSE) then we need to undo the deletion of the vertex and continue to the next iteration.

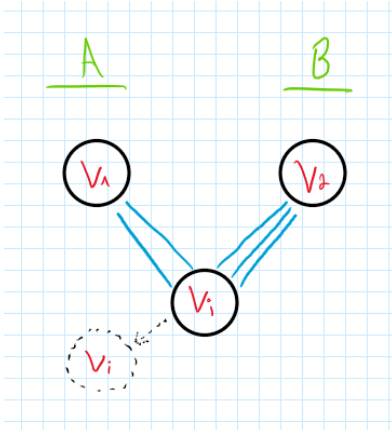
Pseudo:
```
A_o(G)
* check edge cases
k = binary_search(G) # as described above
for v in V:
        G.delete_vertex(v)
        if A_d(G, k):
                if |V| = k:
                        return vertices & edges
        else:
                G.add_vertex(v)
```

Runtime of $A_o(G)$: $|V| \cdot T(A_d) = n \cdot \text{polynomial-time}$

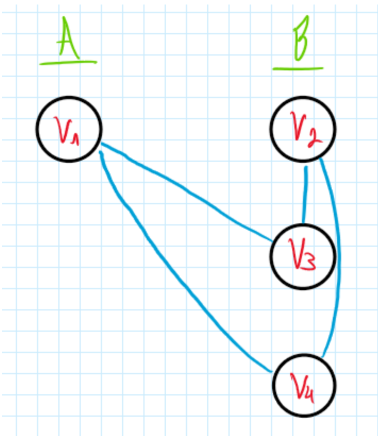
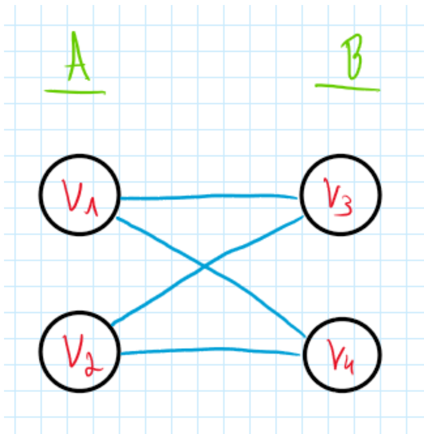
Stav Yosef, 316298876

Question 2

First, we need to understand how the algorithm works, lets see the example below



As we can see $\mathrm{c}(\mathrm{v_i}, \mathrm{A}) = 2$ and $\mathrm{c}(\mathrm{v_i}, \mathrm{B}) = 3$
The algorithm works in a greedy way → in this example the algorithm will decide to assign $\mathrm{v_i}$ to group A. We can see that the worst scenario is if $\boldsymbol{c(v_i, A) = c(v_i, B)}$, therefore half of the edges will be in the maximum cut and half will be "lost" ($\boldsymbol{r = 2}$)
Note - the worst case for this algorithm is when the equality I just mentioned above repeats itself for each of the vertices and their edges equal distributed to $\mathrm{v_1}, \mathrm{v_2}$ (A, B respectively) without edge between $\mathrm{v_1}$ and $\mathrm{v_2}$

Example for the worst case	The algorithm will produce maximum cut size of **2** but the optimal solution is **4**
	

We can see the simplest example above.

Stav Yosef, 316298876

Question 3

Let G be a **complete** graph with weights that obey the **triangle inequality**, and a subset of vertices R (terminals).

Let OPT be the cost of an optimal solution T^* to Steiner tree problem.

We'll start by doubling each edge to obtain an Euler cycle, with DFS tour we know this cost is $2 \cdot \mathrm{OPT}$.

Making a Hamilton circuit using "short-cutting" Steiner vertices and visited vertices (shortcut = connecting new edge between pair of adjacent terminals in the DFS).

The short-cutting doesn't increase the total of the cost (complete graph & triangle inequality). Deleting one edge from this Hamilton circuit yields a spanning tree of R with cost at most $2 \cdot \mathrm{OPT}$.

So, if this tree is a MST on R we know its cost is less than $2 \cdot \mathrm{OPT}$

To summary:

$2 \cdot \mathrm{OPT}$ = cost of the tour on the terminals in T^*

≥ cost of any spanning tree (the tree after the manipulations, adding edges short-cutting and so on)

≥ cost of the MST of the tree

Stav Yosef, 316298876

Question 4

Let b be the number of the bins after using the algo Next-fit and OPT be the number of bins possible that needed to pack all of the items.
Two things we know:

1. The maximum size of each item is $\frac{1}{5}$
2. Each of the bins full at least $\frac{4}{5}$ of the maximum capacity (max cap. = 1) (from 1.)
3. $\mathrm{bin}_{\mathrm{last}} + \mathrm{bin}_{\mathrm{last\text{-}1}} > 1$

The total size of the items is at least $\frac{4}{5} \cdot (\mathrm{b} - 2) + 1 \rightarrow \mathrm{OPT} > \frac{4}{5} \cdot (\mathrm{b} - 2) + 1 = \frac{4\mathrm{b}}{5} - \frac{3}{5}$
Order the equation will results us that $\frac{5}{4}\mathrm{OPT} + \frac{3}{4} > \mathrm{b}$

Let s be a sequence as the following $\mathrm{s} = \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \varepsilon\right)$ where $\varepsilon \leq \frac{4}{5\mathrm{n}}$.
Consider our instance as $\overbrace{\mathrm{s}, \mathrm{s}, \cdots, \mathrm{s}, \mathrm{s}}^{\text{n times}}, \frac{1}{5}$ therefore we have $4 \cdot \mathrm{n} + 1$ of the items $\frac{1}{5}$ and we have n of the items ε.
The algo Next-fit will pack this instance using $\mathrm{n} + 1$ bins and the OPT will do with $\frac{4\mathrm{n}}{5} + 1$ bins.
$\frac{5}{4} \cdot \mathrm{OPT} - \frac{1}{4} = \frac{5}{4}\left(\frac{4\mathrm{n}}{5} + 1\right) - \frac{1}{4} = \mathrm{n} + 1$ exactly as Next-fit algo.

Note - ε derive from the last bin (in optimal solution), $\frac{1 - \frac{1}{5}}{\mathrm{n}} = \frac{\left(\frac{4}{5}\right)}{\mathrm{n}} = \frac{4}{5\mathrm{n}}$

Stav Yosef, 316298876

Question 5

I read and understood the example and the tightness of the analysis.