- The instructions are the same as in Homework-0, 1, 2.

There are 1 questions for a total of 20 points.

---

1. (20 points) Design an $O(poly(n) \cdot 2^{n/4})$-time backtracking algorithm for the maximum independent set problem for graphs with bounded degree 3 (these are graphs where all vertices have degree at most 3). Give running time analysis and proof of correctness.

Solution:

Note that the following algorithm also works for graphs with vertices with degree more than 3, however the exhibited runtime bound matches with the requested runtime bound only for graphs with bounded degree 3.

We exhibit an algorithm that works in $O((n+m)^2 2^{n'/4})$ time, where m is the number of edges in the graph, and n' is defined as

$$\sum_{i \in \mathbb{N}} i \times \text{number of vertices with degree } (i+2)$$

Clearly for a graph with max degree 3, we have $n' \leq n$, and $m = O(n)$, so the algorithm has runtime in $O(n^2 2^{n/4})$.

Note: The following algorithm can be optimized in a real-world implementation using backtracking and updating the graph instead of creating a new graph each time, and maintaining a heap to find the vertex with the least degree. However, the complexity still remains the same and there is no real complexity gain.

Denote by $N(v)$ the neighbours of a vertex v for the sake of brevity. Edges are denoted as ordered pairs $\{u, v\}$ and not as ordered pairs (u, v) since the graph is undirected.

Algorithm

function MaximumIndependentSet(Graph $G = (V, E)$)
  if G is empty **then**
    return $\{\}$
  end if
  let v be the vertex with the least degree in G
  if $deg(v) = 0$ **then**
    return $\{v\} \cup$ MaximumIndependentSet$(G \setminus \{v\})$
  else if $deg(v) = 1$ **then**
    let u be the (only) neighbour of v
    return $\{v\} \cup$ MaximumIndependentSet$(G \setminus \{v, u\})$
  else if $deg(v) = 2$ **then**
    let u_1, u_2 be the (only) neighbours of v.
    if there is an edge between u_1 and u_2 in G **then**
      return $\{v\} \cup$ MaximumIndependentSet$(G \setminus \{v, u_1, u_2\})$
    else
      let v' be a new vertex not in G.
      let $E_o = \{\{u_1, u\} \mid u \in N(u_1)\} \cup \{\{u, u_2\} \mid u \in N(u_2)\}$ ▷ Edges incident to at least one vertex in the set $\{u_1, u_2\}$. Recall that we represent edges as unordered pairs.
      let $E_n = \{\{v', u\} \mid u \in N(u_1) \cup N(u_2) \setminus \{v\}\}$ ▷ Edges that will join v' to all neighbours of u_1, u_2 except v

```
            let G' = ((V ∪ {v'}) \ {v, u_1, u_2}, (E ∪ E_n) \ E_o) ▷ Merge v, u_1, u_2 into 'super-node' v'
            let S = MaximumIndependentSet(G').
            if v' is in S then
                return {u_1, u_2} ∪ S \ {v'}
            else
                return {v} ∪ S
            end if
        end if
    else                                                    ▷ In this case, deg(v) ≥ 3
        let S_1 = {v} ∪ MaximumIndependentSet(G \ ({v} ∪ N(v)))
        let S_2 = MaximumIndependentSet(G \ {v})
        return the larger of S_1, S_2.
    end if
end function
```

Proof of Correctness We prove the correctness of the algorithm using induction on the number of vertices in the graph.

Inductive hypothesis: MaximumIndependentSet(G) returns a maximum independent set of G for all graphs G with number of vertices less than n for some $n > 0$.

Base Case: In the case the graph is empty, the maximum independent set of the graph is the empty set, and hence the algorithm returns the correct answer in the case that the graph is empty.

Inductive step:

Firstly we claim the following:

Claim 1. *For any graph G, and a vertex v in it, if there is a maximum independent set in G that contains exactly one vertex in $\{v\} \cup N(v)$, then there is a maximum independent set in G containing v.*

Proof. Suppose that there exists a maximum independent set S in G that contains exactly one vertex out of the vertices in $\{v\} \cup N(v)$. If that vertex is v, we are trivially done. Now assume that S contains a vertex in $N(v)$, say u. Consider the set $S \setminus \{u\} \cup \{v\}$. Note that since there is no neighbour of v in this set (which is because u was the only neighbour of v in S), this set is also a valid independent set by the validity of S and this observation. Since the size of this set is the same as that of S, this set is also a maximum independent set in G, and we have shown the claim, as needed. □

Now, we make cases on the vertex v with the least degree in the graph.

1. $deg(v) = 0$.

   In this case, since this vertex is isolated, we claim that this vertex will always be in any maximum independent set of G. Indeed, suppose S is any maximum independent set of G that doesn't contain v. Then for any vertex u in S, since $deg(v) = 0$, u is not adjacent to any vertex in S. Hence we can add u to S to get a bigger independent set, which is a contradiction. By the inductive hypothesis, the recursive call gives the rest of the independent set, as needed.

2. $deg(v) = 1$.

   We claim that there is always at least one maximum independent set of G that contains v. Let u be the only neighbour of v. Consider any maximum independent set S of G.

(a) S contains at least one of u, v.

Note that it can't contain both of these vertices. Also, by the claim in the beginning, we have a maximum independent set that contains v.

(b) S contains none of u, v.

By a completely similar analysis, we can show that $S \cup \{v\}$ is also an independent set, which is larger than S, and it contradicts the fact that S is a maximum independent set of G.

Hence, we have shown our claim, so it is always optimal to choose a degree 1 vertex. Since u can't be in such an independent set, the remaining part of the set must come from $G \setminus \{u, v\}$, and must be independent. Note that the maximum independent set of this subgraph is an independent set by definition, and taking the union of this with v yields an independent set of G. It is also the maximum independent set of G, since if it wasn't, then removing the vertex v from the maximum independent set of G which has v would yield a larger independent set of $G \setminus \{u, v\}$. By the induction hypothesis, the recursive call in the function corresponding to this case gives a maximum independent set of this case, and we are done in this case.

3. $deg(v) = 2$.

Let u_1, u_2 be the two neighbours of v. In this case, we make two subcases:

(a) There is an edge between u_1, u_2. In this case, we can choose exactly one of v, u_1, u_2 in the maximum independent set (if we choose none, the maximum property is violated by adding in v, and if we choose two or more, the set is no longer independent). By the claim, it is optimal to choose v, and the remaining set must come from $G \setminus \{v, u_1, u_2\}$. Using a completely analogous argument as in the degree 1 case, we can show that the union of the chosen vertex (which is v) and the answer of the recursive call is a maximum independent set of the graph.

(b) There is no edge between u_1, u_2. In this case, we can choose either one or two vertices from $\{v, u_1, u_2\}$ in the maximum independent set of G. We combine the nodes v, u_1, u_2 into a node v' to get a new graph G'. Note that the recursive call computes a maximum independent set of this new graph since it has two less vertices than G, call that set S. Our claim is the following:

Claim 2. *If S contains v', then it is optimal to choose u_1, u_2 in G, and if S doesn't contain v, then it is optimal to choose v in G.*

Proof. We first show how to construct an independent set of G from S which has exactly one more vertex than S. We make two cases:

i. S contains v'.

Note that since S contains v', no vertex adjacent to v' in G' is in S, and thus no neighbour of u_1, u_2 is in S. Hence we can add u_1, u_2 after removing v' to get an independent set of G with exactly one more vertex than S has.

ii. S doesn't contain v'.

In this case, we can add v to S, and this is an independent set because no neighbour of v is in S. Since we have added one vertex to S, the size of this set is exactly one more than that of S.

Now we show that the size of any independent set S' of G is at most $1 + |S|$. We make two cases again:

i. At least two of v, u_1, u_2 are in S'.

In this case, since v is adjacent to u_1, u_2, so v can't be in S'. Hence u_1, u_2 are both in S. Consider the corresponding independent set in G' (formed by choosing the vertices in S' except u_1, u_2 but possibly also containing v' – the set without v' is independent because of being a subset of S', but as we show in the next sentence, we can choose v' as well and still have this set as independent in G'). Note that since u_1, u_2 have both been chosen, v' has been chosen in the corresponding independent set in G' (since the neighbours of v' are precisely the neighbours of u_1, u_2 except v, and none of them have been chosen in S' because u_1, u_2 have been chosen), so we have an independent set of size $|S'| - 1$ in G'. Now that we know that the maximum size of an independent set in G' is $|S|$, we have $|S'| - 1 \leq |S|$, or $|S'| \leq 1 + |S|$, as needed.

ii. Exactly one of v, u_1, u_2 is in S'.

Again, considering the corresponding independent set in G', either v' is chosen or it is not, and in either case, we have that the size of that set is either $|S'|$ or $|S'| - 1$. Since the size of the corresponding independent set is at most $|S|$ (as S is a maximum independent set of G'), $|S'|$ is at most $\max(|S|, |S| + 1)$, so we have $|S'| \leq |S| + 1$ in this case as well.

From this analysis, we see that any independent set of G must have at most $|S|+1$ vertices. Since we have exhibited an independent set of G with exactly these many vertices, it must be a maximum independent set of G in each case. □

Note that this finishes the proof of the claim, and thus by the claim (and the construction mentioned in the proof), the independent set returned by the algorithm is indeed a maximum independent set of G.

4. $deg(v) \geq 3$.

In this case, either we choose v or not. If we choose v, the answer is the union of v with the maximum independent set of the graph obtained after we remove v and all its three neighbours, and if we don't, then the answer is the union of v with the maximum independent set of the graph obtained after we remove v from G, as done in class. Since both of these graphs have number of vertices strictly less than that of G, using the inductive hypothesis, we are done in this case as well.

By the above case analysis, we can see that the algorithm returns a maximum independent set in all cases, and thus the induction is complete.

Time complexity analysis Let n, m be the number of nodes, edges in the graph, and let n' be as defined. Let the number of constant-time operations needed in this algorithm be denoted by $f(n, m, n')$. Then we claim that $f(n, m, n') \leq c(n + m + 1)^2 2^{n'/4}$ for any non-empty graph where c is a suitably chosen large enough constant, which will be specified later on.

We first begin with two claims:

Claim 3. $n' = \sum_{v \in V(G)} \max(0, deg(v) - 2)$.

Proof. Note that for each vertex with degree i contributes $i - 2$ to the right hand side if $i \geq 3$ and

0 otherwise. Hence by grouping vertices by degree, we have the result. More precisely, we have:

$$\begin{aligned}\sum_{v\in V(G)} \max(0, deg(v)-2) &= \sum_{j\in\mathbb{N}} \sum_{v\in V(G), deg(v)=j} \max(0, j-2)\\ &= \sum_{j\in\mathbb{N}_{\geq 3}} \sum_{v\in V(G), deg(v)=j} (j-2)\\ &= \sum_{i\in\mathbb{N}} \sum_{v\in V(G), deg(v)=i+2} i\\ &= \sum_{i\in\mathbb{N}} i \times |\{v\in V(G) \mid deg(v)=i+2\}|\\ &= n'\end{aligned}$$

as required. □

Claim 4. *Whenever we create a graph G' from G in the algorithm, $n'(G') \leq n'(G)$.*

Proof. Note that in the cases with minimum degree 0, 1, or at least 3, we always remove vertices from the graph, which either reduces the degree of any remaining vertex or keeps it the same, and also removes the non-negative contribution of the removed vertices from the expression of n'. Hence the inequality holds in that case. The same holds in the subcase of the case of minimum degree 2 where we have an edge between u_1, u_2.

The more important case is when the minimum degree is 2 and we replace a group of nodes with a new node.

In the case when the degree is 2, the highest possible degree of the combined node is $deg(u_1) + deg(u_2) - 2$ (this is in the case when $N(u_1), N(u_2)$ intersect in only one vertex v). The contribution of v, u_1, u_2 to $n'(G)$ is $0, \max(deg(u_1)-2, 0), \max(deg(u_2)-2, 0)$ respectively. Also, for any vertex adjacent to both u_1, u_2, its degree in the new graph is one less than its old degree, and this can't increase the difference between $n'(G'), n'(G)$. Hence we have

$$n'(G') - n'(G) \leq \max(deg(u_1) + deg(u_2) - 4, 0) - \max(deg(u_1)-2, 0) - \max(deg(u_2)-2, 0)$$

Note that the real function $f(a) = \max(a, 0)$ is the point-wise maximum of the identity function and the 0 function, both of which are convex, and hence it is also convex. So it satisfies $f(a) + f(b) \geq 2f\left(\frac{a+b}{2}\right)$ (special case of Jensen's inequality), and thus applying this inequality to $(a, b) = (deg(u_1) - 2, deg(u_2) - 2)$, we have that

$$\begin{aligned}n'(G') - n'(G) &\leq \max(deg(u_1) + deg(u_2) - 4, 0) - \max(deg(u_1)-2, 0) - \max(deg(u_2)-2, 0)\\ &= 2\max\left(\frac{deg(u_1) + deg(u_2) - 4}{2}, 0\right) - \max(deg(u_1)-2, 0) - \max(deg(u_2)-2, 0)\\ &\leq 0\end{aligned}$$

This completes the proof. □

Now we move to the main proof of the time complexity.

Note that all the constants c_i mentioned in the below proof shall be such that the mentioned upper bound holds for all n, and not just the particular n considered in the inductive step.

Proof. The proof goes by strong induction on the lexicographical ordering of tuples (n, m, n') of non-negative integers.

For the base case, if the number of operations to check if the graph is empty is c_1, then if we have $c \geq c_1$, the base case holds.

The number of operations to find out the vertex with the least degree is at most linear in $n + m$. Suppose it is upper bounded by $c_2(n + m)$ for all n, m.

If the degree is 0, then removing the vertex from the graph takes at most linear operations (wherever operations are mentioned, we mean constant-time operations unless specified otherwise). Suppose the time for this and taking the union and returning the set is upper bounded by $c_3(n + m)$ for all n, m. So we have

$$\begin{aligned} f(n, m, n') &\leq f(n-1, m, n') + c_3 n + c_2 n \\ &\leq c(n+m)^2 2^{n'/4} + (c_3 + c_2)(n + m) \\ &\leq c(n+m+1)^2 2^{n'/4} \end{aligned}$$

where the last inequality is true if we have $c > c_3 + c_2$.

If the degree is 1, then removing both vertices and returning the set takes linear time, say upper bounded by $c_4(n + m)$ for all n, m. Then we have

$$\begin{aligned} f(n, m, n') &\leq \max_{0 \leq i \leq n'} f(n-2, m - \deg(u), n' - i) + (c_4 + c_2)(n + m) \\ &\leq c(n+m-1)^2 2^{n'/4} + (c_4 + c_2)(n + m) \\ &\leq c(n+m+1)^2 2^{n'/4} \end{aligned}$$

where the last inequality is true if we have $c > c_4 + c_2$.

If the degree is 2, the checking of the two subcases takes at most linear number of operations, upper bounded by, say $c_5(n + m)$ for all n, m.

In the first subcase, the removal of the vertices to get a new graph, taking the union and returning takes at most $c_6(n + m)$ operations for all n, m, for some constant c_6. Then we have

$$\begin{aligned} f(n, m, n') &\leq \max_{0 \leq i \leq n'} f(n-3, m-3-\epsilon, n' - i) + (c_6 n + c_5 + c_2)(n + m) \\ &\leq c(n+m-5)^2 2^{n'/4} + (c_6 n + c_5 + c_2)(n + m) \\ &\leq c(n+m-5)^2 2^{n'/4} + (c_6 n + c_5 + c_2)(n + m) \\ &\leq c(n+m+1)^2 2^{n'/4} \end{aligned}$$

where the last inequality is true if $c > c_5 + c_6 + c_2$.

In the second subcase, the creation of the new graph takes a linear number of operations, and so does taking the union, searching in the returned set, modifying the returned set and returning it. Suppose it is upper bounded by $c_7(n + m)$ for all n, m.

$$\begin{aligned} f(n, m, n') &\leq \max_{0 \leq i \leq n'} f(n-2, m-2-\epsilon, n' - i) + (c_7 + c_5 + c_2)(n + m) \\ &\leq c(n+m-3)^2 2^{n'/4} + (c_7 + c_5 + c_2)(n + m) \\ &\leq c(n+m+1)^2 2^{n'/4} \end{aligned}$$

where the last inequality is true if $c > c_7 + c_5 + c_2$.

If none of the above cases is true, then the minimum degree d in the graph must be at least 3, and as a consequence, all vertices in the graph must have degree at least d. Comparing sizes of sets and returning the larger one of them takes linear number of operations, and so does creating two graphs with one and two vertices removed. Suppose all of this is upper bounded by $c_8(n+m)$ for all n, m.

We compute the difference in n' in this case. When we remove a vertex with degree d, n' decreases by at least $2d-2$, since the contribution of this vertex is $d-2$, and the reduction in the degree of each other vertex is exactly 1. When we remove this vertex and the set of all neighbours of this vertex, the reduction in n' is at least $(d+1)(d-2)$. Let the exact reduction be d'.

We then have:

$$\begin{aligned}
f(n,m,n') &\leq f(n-1, m-d, n'-2d+2) + f(n-d-1, m-d-\epsilon, n'-d') + (c_8+c_2)(n+m) \\
&\leq cn^2 2^{(n'-2d+2)/4} + c(n-d)^2 2^{(n'-d')/4} + c_8 n + c_2 n \\
&\leq c(n+m-d)^2 2^{(n'-2d+2)/4} + c(n+m-2d)^2 2^{(n'-(d+1)(d-2))/4} + (c_8+c_2)(n+m) \\
&\leq c(n+m-3)^2 2^{(n'-4)/4} + c(n+m-6)^2 2^{(n'-4)/4} + (c_8+c_2)(n+m) \\
&= \frac{1}{2}\cdot c(n+m-3)^2 2^{n'/4} + \frac{1}{2}\cdot c(n+m-6)^2 2^{n'/4} + (c_8+c_2)(n+m) \\
&\leq \frac{1}{2}\cdot c(n+m)^2 2^{n'/4} + \frac{1}{2}\cdot c(n+m)^2 2^{n'/4} + (c_8+c_2)(n+m) \\
&= c(n+m)^2 2^{n'/4} + (c_8+c_2)(n+m) \\
&\leq c(n+m+1)^2 2^{n'/4}
\end{aligned}$$

where the last inequality is true if $c > c_8 + c_2$. We have made use of the facts that $d \geq 3$, $d' \geq 4d-8$, and that $(d+1)(d-2)$ is increasing on $[2, \infty)$.

Note that if we choose $c = 1 + c_2 + \max(c_1 - c_2, c_3, c_4, c_5 + c_6, c_5 + c_7, c_8)$, all of these claims will be valid, and thus our proof by induction is complete. □

Since our claim is completed, we have that for $n > 0$, our algorithm takes $O((n+m+1)^2 2^{n'/4})$ time, since it takes the same order of constant time operations. Note that we have $n' \leq n$ as well as $m = O(n)$ for any graph with bounded degree 3, so the runtime of the algorithm is in $O(n^2 2^{n/4})$, as required.