CS 39000-ATA: Homework 0

Due on 2025-01-23 23:59

Prof. Kent Quanrud, Spring 2025

Mukul Agarwal and Kovid Tandon

Additional Collaborators: Aaryan Wadhwani, Saad Sharief, Maanas Karwa, and TA Peter Jin

Problem 2.2

For each of the recursive specifications below, design a recursive algorithm implementing the specification. (No proof or analysis is needed.)

3. max-matching$(G = (V, E))$: Given an undirected graph $G = (V, E)$, return the maximum size of any matching. (A matching is a set of edges $M \subseteq E$ with disjoint endpoints.)

```
max-matching(G = (V, E))
    if |E| = 0 then
        return 0
    end if
    // Inductive case: arbitrary (u, v) ∈ E is either in or not in a max matching
    let (u, v) ∈ E be an arbitrary edge
    // Let E_u be the sets of edges incident to u
    let E_u ← {e ∈ E : u ∈ e}
    // Let E_v be the sets of edges incident to v
    let E_v ← {e ∈ E : v ∈ e}
    // Let G_1 be the graph obtained by removing u, v and all incident edges
    let G_1 ← (V \ {u, v}, E \ (E_u ∪ E_v))
    // Let G_2 be the graph obtained by removing the edge (u, v)
    let G_2 ← (V, E \ (u, v))
    return max(1 + max-matching(G_1), max-matching(G_2))
end algorithm
```

4. <u>max-independent-set</u>$(G = (V, E))$: Given an undirected graph $G = (V, E)$, return the maximum size of any independent set of vertices. (Given an undirected graph $G = (V, E)$, an independent set is a set of vertices $S \subseteq V$ such that no two vertices $u, v \in S$ are connected by an edge.)

```
max-independent-set(G = (V, E))
    // Base case
    if |V| = 0 then
        return 0
    end if
    // Inductive case: arbitrary s ∈ V is either in or not in a m.i.s.
    let s ∈ V be arbitrary
    // Let N_s be the set of all vertices adjacent to s including s
    let N_s ← {v ∈ V : {s, v} ∈ E} ∪ {s}
    // Let E_s be the set of all edges incident to s
    let E_s ← {e ∈ E : s ∈ E_s}
    // Let E_{N_s} be the set of all edges incident to all vertices in N_s
    let E_{N_s} ← {e ∈ E : e ∩ N_s ≠ ∅}
    // Let G_1 be the graph obtained by removing all vertices in N_s, all edges in E_{N_s}
    let G_1 ← (V \ N_s, E \ E_{N_s})
    // Let G_2 be the graph obtained by removing s and all incident edges to s
    let G_2 ← (V \ {s}, E \ E_s)
    return max(1 + max-independent-set(G_1), max-independent-set(G_2))
end algorithm
```

5. <u>longest-increasing-subsequence-including-first</u>$(x_1, \ldots, x_n)$: Given a sequence of numbers $x_1, \ldots, x_n$, return the length of the longest (strictly) increasing subsequence of $x_1, \ldots, x_n$ over all subsequences that include x_1. (You may assume $n \geq 1$).

   A subsequence of $x_1, \ldots, x_n$ is a sequence of the form $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$, where $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. A sequence of numbers $y_1, \ldots, y_\ell$ is strictly increasing if $y_i < y_{i+1}$ for $i = 1, \ldots, \ell - 1$.

```
longest-increasing-subsequence-including-first(x_1, ..., x_n)
    // The trivial subsequence is guaranteed
    let m ← 1
    // No need for explicit base case since [n] \ {1} is empty if n = 1
    for j ∈ [n] \ {1} do
        if x_j > x_1 then
            // Consider the subseq. starting as x_1, x_j, ...
            let m′ ← 1 + longest-increasing-subsequence-including-first(x_j, ..., x_n)
            m ← max(m, m′)
        end if
    end for
    return m
end algorithm
```

6. reachable$(G = (V, E), s, t)$: Given a directed graph G and two vertices $s, t \in V$, return true if s can reach t in G, and false otherwise. (s can reach t if either $s = t$ or there is a sequence of (directed) edges of the form $(s, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k), (v_k, t)$. Note that the endpoint of one edge is the initial point of the next edge. This is also called a (directed) walk in G from s to t.)

```
reachable(G = (V, E), s, t)
    if s = t then
        return true
    end if
    // |V_{G'}| < |V|
    // Let E_s be the set of all directed edges starting at s
    let E_s ← {e ∈ E : ∃s' ∈ V s.t. e = (s, s')}
    // Let G' be the graph obtained by removing s and all directed edges starting at s
    let G' ← (V \ {s}, E \ E_s)
    // for any vertex v ≠ s such that there exists an edge from s to v
    for v in {v ∈ V \ {s} : (s, v) ∈ E} do
        if reachable(G', v, t) then
            // t is reachable from s using v as an intermediate node
            return true
        end if
    end for
    return false
end algorithm
```

7. <u>partition</u>$(x_1, ..., x_n)$: Given n integers $x_1, ..., x_n \in \mathbb{Z}$, returns true if there is a subset of indices $S \subseteq [n]$ such that $\sum_{i \in S} x_i = \sum_{i \in [n] \setminus S} x_i$, and false otherwise.

```
partition(x_1, ..., x_n)
    if n ≤ 1 then
        return (n = 0) ∨ (x_1 = 0)
    end if
    return partition(x_1 + x_2, x_3, ..., x_n) ∨ partition(x_1 − x_2, x_3, ..., x_n)
end algorithm
```